UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione di un sistema di supporto ad un CMMS finalizzato alla creazione e gestione di dati aziendali

Design of a system to support a CMMS for business data creation and management

Relatore

Prof. Domenico Ursino

Candidato

Alessandro Bedetta

Correlatore

Ing. Antonio Spadaccini

Correlatore

Ing. Botond Zalai-Ruzsics

ANNO ACCADEMICO 2022-2023

A tutti coloro che mi hanno permesso
di raggiungere questo traguardo...
Ancona
Dicembre 2023

Sommario

Questa tesi si propone di affrontare la sfida della progettazione e del parziale sviluppo di un sistema di supporto integrato nell'ambito di un contesto aziendale. L'obiettivo primario di questa infrastruttura è migliorare significativamente la gestione dei dati connessi alla logica aziendale. Nel panorama lavorativo, la carenza di una struttura robusta per la rappresentazione di ingenti quantità di informazioni è una problematica diffusa, particolarmente evidente in contesti meno estesi, che spesso conduce a situazioni di errore ed inefficienza.

L'approccio adottato in questa tesi consiste nell'unire la progettazione di strutture informative ed informatiche con la profonda comprensione del dominio aziendale. Tale sinergia mira a sviluppare un sistema capace di offrire un contributo significativo al potenziamento della gestione dei processi chiave dell'azienda. La necessità di superare le sfide legate alla rappresentazione e alla manipolazione di grandi volumi di dati si traduce in un impegno concreto per sviluppare soluzioni innovative e efficaci.

Keywords: Manutenzione, CMMS, Database, Data Exchange, Asset, Piani, Operazioni, Analisi di dati

Indice

Elenco delle figure

Introduzione

Nel contesto aziendale odierno, la gestione efficiente delle attività di manutenzione svolge un ruolo cruciale per garantire la continuità operativa e la durata degli impianti industriali e degli asset aziendali. Le imprese specializzate nella consulenza per la manutenzione affrontano sfide sempre più complesse, che richiedono soluzioni innovative e l'adozione di strumenti avanzati per ottimizzare le operazioni quotidiane.

In questo scenario, l'implementazione di un sistema informativo completo ed integrato emerge come elemento fondamentale per migliorare la gestione, la pianificazione e l'esecuzione delle attività di manutenzione. Un'analisi dell'evoluzione della gestione delle attività di manutenzione nel corso degli anni evidenzia un passato in cui l'intera organizzazione delle operazioni legate ai piani manutentivi avveniva su supporto cartaceo. Gli addetti, manualmente, consultavano registri di manutenzione per individuare le attività associate a ciascuna attrezzatura. Questo approccio è stato radicalmente superato dai Computerized Maintenance Management System (CMMS). In tempi recenti, questi strumenti software hanno trasformato in modo permanente il modo in cui si gestisce la manutenzione, sostituendo l'approccio cartaceo con una piattaforma centralizzata per la pianificazione, la registrazione e il monitoraggio di interventi di manutenzione preventiva e correttiva su macchinari, impianti e asset aziendali.

Uno dei CMMS più diffusi è SAP Plant Maintenance (PM), ampiamente utilizzato dai clienti che affidano la creazione di piani di manutenzione a Pansoinco, l'azienda coinvolta nel progetto descritto in questa tesi. Tuttavia, la mancanza di uno standard interno per la gestione delle informazioni legate ai piani di manutenzione in Pansoinco porta a una personalizzazione delle rappresentazioni dei dati in base alle esigenze specifiche del cliente. Questa mancanza di standardizzazione comporta rischi, tra cui la mancanza di strutture di validazione centralizzate, aumentando il potenziale per errori umani nella creazione dei piani di manutenzione e complicando l'estrazione di conoscenza dai dati aziendali nel corso del tempo.

Affrontare queste problematiche potrebbe conferire all'azienda un notevole vantaggio competitivo. Miglioramenti nell'efficacia e nell'efficienza durante la creazione dei piani di manutenzione, la potenziale capacità di associare automaticamente attrezzature a operazioni specifiche, e un significativo incremento dell'integrità e della consistenza dei dati rappresentano solo alcuni dei benefici che potrebbero derivare dalla risoluzione di tali sfide. In definitiva, ciò contribuirebbe a prevenire anomalie legate ad errori o duplicazione di informazioni, promuovendo una gestione della manutenzione più solida ed efficiente.

La presente tesi si prefigge l'ambizioso obiettivo di esplorare in dettaglio e sviluppare un sistema informativo su misura, mirato a soddisfare le esigenze specifiche dell'azienda.

L'obiettivo centrale di questo sistema è quello di ottimizzare le procedure aziendali, mettendo in campo soluzioni avanzate come l'implementazione di un database centralizzato, un'interfaccia utente basata su operazioni CRUD (Create, Read, Update, Delete) e un sofisticato programma di scambio dei dati. L'intento è agevolare la comunicazione e la condivisione efficiente delle informazioni tra tutti gli attori coinvolti nel processo, contribuendo, così, a un aumento complessivo dell'efficienza operativa.

La creazione di un database robusto rappresenta un passo fondamentale in questo percorso. Questo strumento consentirà di raccogliere, organizzare e archiviare in modo strutturato tutti i dati pertinenti agli impianti, alle attività di manutenzione e alle risorse impiegate. Prima di procedere con la progettazione dello schema del database, è prevista una fase di mappatura della struttura del database interno a SAP PM. Questo approccio strategico permetterà di comprendere a fondo le dinamiche esistenti e di identificare le aree in cui possono essere apportate migliorie significative.

Successivamente, sullo schema di tabelle derivato da SAP PM, verranno eseguiti tagli e aggiunte mirati. Questo processo ha come obiettivo quello di creare lo schema finale del progetto, ottimizzato per le specifiche esigenze dell'azienda. Le modifiche apportate allo schema SAP non saranno solo superficiali, ma strategicamente finalizzate a conferire al database una maggiore malleabilità ed adattabilità. L'obiettivo è allineare il sistema alle particolari esigenze e agli standard di rappresentazione utilizzati da Pansoinco nei vari progetti di manutenzione. Tale adattamento mirato è essenziale per garantire che il sistema informativo non solo risponda alle esigenze attuali, ma sia in grado di evolversi e adattarsi in futuro a nuove sfide e opportunità nel contesto aziendale.

La realizzazione di questo robusto database costituirà le fondamenta su cui poggerà l'innovativa interfaccia CRUD, progettata per offrire agli utenti un accesso intuitivo e rapido alle informazioni. Tale interfaccia consentirà loro di effettuare operazioni di inserimento, consultazione, aggiornamento ed eliminazione dei dati in modo efficace e sicuro. Durante il complesso processo di sviluppo di questa interfaccia, è stato cruciale fare affidamento sul potente strumento SQL Maestro PHP Generator. Tale tool, fornito da Pansoinco, si distingue come uno strumento professionale di generazione di interfacce Web, basato sulla struttura portante del database. La sua implementazione garantisce una sinergia armonica tra la struttura dei dati e l'esperienza utente, creando un ambiente di lavoro che risponde in modo ottimale alle esigenze specifiche dell'azienda.

Parallelamente a questa componente, un programma di data exchange sarà fondamentale per agevolare lo scambio fluido di informazioni tra i sotto-sistemi coinvolti. Tale programma non solo favorirà l'integrazione dei dati, ma garantirà anche coerenza tra le diverse componenti del sistema informativo. La sua versatilità sarà evidenziata dalla capacità di consentire a qualsiasi utente di effettuare operazioni di caricamento massivo di informazioni nel database. Tale processo sarà gestito in modo efficiente, occupandosi della conversione fra formati e segnalando tempestivamente eventuali situazioni di errore. In questo modo, si promuoverà un flusso di dati continuo e affidabile all'interno del sistema informativo.

Attraverso lo sviluppo di questo sistema informativo completo, l'obiettivo ultimo è contribuire in modo significativo al miglioramento della produttività, dell'efficienza operativa e della qualità del servizio offerto dall'azienda. La ricerca e lo sviluppo di soluzioni avanzate in questo settore non sono soltanto una necessità, ma rappresentano un passo fondamentale per affrontare le sfide in continua evoluzione e per mantenere un vantaggio competitivo nell'industria della manutenzione. Questo progetto si propone di essere una risposta innovativa alle esigenze sempre crescenti del settore, posizionando l'azienda in prima linea nel panorama competitivo e aprendo nuove prospettive di successo.

La struttura della presente tesi si articola attraverso sei capitoli, ognuno dei quali rappresenta un tassello essenziale per la comprensione e l'implementazione del sistema informativo

dedicato alla pianificazione della manutenzione industriale.

- *Capitolo 1: Contesto di riferimento*. In questo capitolo , si offre una panoramica dettagliata sull'azienda oggetto di studio e sull'ampio contesto in cui essa opera, focalizzandosi sulla rilevanza della pianificazione della manutenzione industriale. Attraverso un'analisi approfondita, si cercherà di delineare il contesto in cui si inserisce la ricerca, evidenziando l'importanza strategica della manutenzione nelle attività industriali.

- *Capitolo 2: Obiettivi e Requisiti*. Questo capitolo approfondisce gli obiettivi e i requisiti che il sistema informativo intende soddisfare. Si delinea la soluzione proposta, esaminando in dettaglio i principali vantaggi e svantaggi. Questa fase di analisi critica mira a fornire una visione completa delle sfide e delle opportunità che il progetto affronta.

- *Capitolo 3: Attività di Reverse Engineering*. Questo capitolo si concentra sulla fase di reverse engineering, che ha condotto alla completa mappatura del CMMS SAP PM. Questa mappatura costituirà la base per la progettazione del database, con uno sguardo attento alle scelte progettuali adottate e alle loro implicazioni sul sistema informativo.

- *Capitolo 4: Sviluppo del Database*. Questo capitolo analizza in dettaglio le fasi di progettazione e realizzazione della base dati. Verranno esplorate le decisioni di progettazione chiave e le soluzioni tecniche adottate per garantire un sistema informativo robusto e adattabile alle esigenze specifiche della manutenzione industriale.

- *Capitolo 5: Sviluppo dell'interfaccia CRUD*. Questo capitolo si concentrerà sulla descrizione dell'interfaccia CRUD, elemento cruciale per l'accesso e la gestione efficiente dei dati. Si esploreranno gli strumenti utilizzati per agevolarne l'implementazione, sottolineando l'importanza di un'interfaccia utente intuitiva e funzionale.

- *Capitolo 6: Componente di Data Exchange*. Questo capitolo analizzerà lo stato di sviluppo della componente di data exchange, con un focus specifico sulla fase di data loading. Si esploreranno le sfide incontrate e le soluzioni adottate per garantire un flusso coerente e affidabile di informazioni tra i diversi sotto-sistemi del sistema informativo.

CAPITOLO 1

Contesto di riferimento

In questo primo capitolo della trattazione si intende delineare un contesto in grado di fungere da background a tutto quello che poi verrà esposto nelle sezioni a venire. Di conseguenza, verranno rapidamente illustrati alcuni concetti e termini fondamentali al fine di garantire una corretta comprensione. Si partirà riassumendo cosa si intende quando si parla di manutenzione industriale per poi spostare l'attenzione su quella che è l'attività di Pansoinco in questo contesto. A completamento del capitolo si discuterà di come la gestione delle attività manutentive è evoluta nel corso degli anni, passando da metodi e tecniche rudimentali all'utilizzo di strumenti avanzati e moderni in grado di migliorare la gestione e la pianificazione, sia sotto il punto di vista dell'efficienza che dell'efficacia. Fra questi tool utilizzati ai giorni nostri troviamo i CMMS.

1.1 Manutenzione industriale

Nell'attuale panorama industriale, la manutenzione riveste un ruolo cruciale nella gestione e nell'ottimizzazione delle risorse. In un contesto in cui l'efficienza operativa è essenziale per la competitività delle imprese, la manutenzione industriale emerge come un elemento strategico per garantire la continuità delle attività produttive, la sicurezza degli impianti e la massimizzazione della vita utile degli asset.

Il settore manifatturiero, in particolare, è caratterizzato da una complessità sempre crescente, con impianti e macchinari che spesso operano in condizioni estreme e affrontano sfide legate all'usura, alla corrosione e alle sollecitazioni meccaniche.

In questo contesto, la manutenzione assume un ruolo proattivo, non solo intervenendo in caso di guasti o malfunzionamenti, ma pianificando e implementando strategie preventive per evitare potenziali problemi e massimizzare l'efficienza operativa.

Il contesto attuale è caratterizzato da una crescente consapevolezza dell'importanza della manutenzione predittiva, che si basa sull'utilizzo di avanzate tecnologie, come il monitoraggio dello stato degli asset tramite sensori, l'analisi dei dati in tempo reale e l'Intelligenza Artificiale. Queste tecnologie stanno rivoluzionando il modo in cui le imprese affrontano la manutenzione, consentendo interventi tempestivi e mirati per evitare fermi imprevisti e prolungare la vita utile degli impianti.

Un ulteriore aspetto chiave della manutenzione industriale riguarda la gestione delle risorse umane coinvolte in queste attività. La formazione continua e lo sviluppo delle competenze sono fondamentali per garantire che il personale addetto alla manutenzione sia in grado di adottare le nuove tecnologie e di affrontare sfide sempre più complesse. La sicurezza sul luogo di lavoro è un altro elemento critico, considerando l'ambiente spesso pericoloso in cui si svolgono le attività di manutenzione.

La manutenzione industriale, quindi, si presenta come un elemento cruciale per il successo e la sostenibilità delle imprese nel contesto attuale. Pansoinco Srl è un'azienda che da più di 50 anni lavora nel campo della manutenzione di impianti energetici, specializzata nella consulenza e nello sviluppo di soluzioni mirate ad aziende che operano nel settore.

Fondata nel 1970 con il nome Soinco Argentina Engineering, l'azienda ha acquisito, in particolare negli ultimi anni, una reputazione per la grande capacità di soddisfare le necessità dei clienti, migliorandone l'efficienza operativa e la produttività.

1.2 Contesto aziendale

Nel dettaglio i principali settori in cui si muove l'azienda sono principalmente la consulenza tecnica, lo sviluppo di soluzioni ingegneristiche e l'asset management. Con consulenza tecnica si vuole intendere un servizio fornito da esperti nel campo specifico dell'ingegneria, i quali forniscono supporto e consigli basati sulla loro conoscenza ed esperienza professionale. La consulenza tecnica può riguardare una vasta gamma di argomenti, tra cui la progettazione, la sicurezza, le normative, la valutazione di tecnologie e molto altro ancora.

Lo sviluppo di soluzioni ingegneristiche implica la progettazione e l'implementazione di nuove soluzioni, prodotti o sistemi ingegneristici. Questo processo può coinvolgere fasi come la progettazione concettuale, la progettazione dettagliata, la prototipazione, i test e l'implementazione. Gli ingegneri e i professionisti del settore lavorano per sviluppare soluzioni che rispondano a specifiche esigenze o problemi, integrando conoscenze scientifiche e tecnologiche per creare prodotti o processi innovativi e funzionali.

Infine, l'asset management è una pratica che coinvolge la gestione strategica degli asset di un'organizzazione. Gli asset possono includere infrastrutture, impianti, attrezzature, e molto altro ancora. L'obiettivo dell'asset management è quello di massimizzare il valore degli asset attraverso la loro ottimizzazione, manutenzione efficace, il monitoraggio delle prestazioni e la gestione del ciclo di vita. Questa pratica contribuisce a garantire che queste risorse siano utilizzate in modo efficiente per supportare gli obiettivi aziendali, minimizzando i costi e mitigando i rischi associati alla proprietà e all'utilizzo degli asset.

1.3 Background storico sulla manutenzione

La pianificazione della manutenzione degli impianti in passato era spesso basata su strategie di manutenzione preventive e correttive. Gli operatori e i responsabili della manutenzione si basavano sul manuale dell'impianto e sulle specifiche tecniche per comprendere i requisiti di manutenzione consigliati dal costruttore o dal progettista.

L'intera manutenzione consisteva, quindi, nella consultazione di questi documenti i quali fornivano informazioni dettagliate sui componenti, sui tempi di vita previsti e sulle procedure di manutenzione raccomandate. Sulla base delle informazioni fornite dalle specifiche tecniche, si definivano cicli di manutenzione preventiva, i quali potevano includere attività pianificate, come la sostituzione regolare di parti soggette a usura, la pulizia di componenti critici e l'esecuzione di test diagnostici periodici. Si pianificavano ispezioni regolari per valutare lo stato generale dell'impianto; queste ultime potevano essere condotte internamente dal personale di manutenzione o, in alcuni casi, si potevano coinvolgere aziende specializzate. Era pratica comune tenere registri dettagliati di tutte le attività di manutenzione eseguite sull'impianto, includendo informazioni come la data dell'intervento, le attività svolte, le parti sostituite e qualsiasi problema riscontrato.

In caso di guasti improvvisi, si applicava la manutenzione correttiva. Questa forma di manutenzione non era pianificata e si basava sulla risoluzione tempestiva dei problemi per ripristinare il funzionamento dell'impianto. Periodicamente, si effettuavano valutazioni delle condizioni operative dell'impianto per identificare potenziali problemi in anticipo. Ciò poteva coinvolgere analisi delle performance, controllo dei parametri di funzionamento e monitoraggio delle tendenze nel tempo.

1.4 La manutenzione ai giorni nostri - I CMMS

Le pratiche di manutenzione degli impianti hanno subito notevoli cambiamenti nel corso del tempo, principalmente a causa dell'avanzamento tecnologico e dell'adozione di approcci più avanzati.

Con l'avanzare della tecnologia e l'adozione di sistemi di monitoraggio avanzati, la pianificazione della manutenzione è diventata sempre più predittiva. Oggi, i sensori intelligenti e i sistemi di monitoraggio remoto forniscono dati in tempo reale consentendo di anticipare i guasti e di ottimizzare le attività di manutenzione in base alle reali condizioni di funzionamento dell'impianto.

L'uso crescente di tecnologie come l'Internet of Things (IoT), i sensori intelligenti e i dispositivi connessi ha reso possibile il monitoraggio in tempo reale delle condizioni degli impianti. Ciò ha aperto la strada a una transizione verso la manutenzione predittiva, in cui gli interventi vengono programmati in base alle reali condizioni degli asset, riducendo al minimo le interruzioni impreviste e ottimizzando le risorse.

Negli ultimissimi anni vengono impiegate anche tecnologie quali l'AI e il Machine Learning per analizzare grandi quantità di dati provenienti dai sensori e dai sistemi di monitoraggio. Queste tecnologie consentono di identificare modelli, prevedere guasti imminenti e ottimizzare le strategie di manutenzione in modo dinamico.

Tuttavia una delle più significative ed importanti innovazioni nel mondo della pianificazione delle attività manutentive industriali è stata l'introduzione dei sistemi *CMMS*. I Computerized Maintenance Management Systems, sono sistemi informatici progettati per facilitare la gestione delle operazioni di manutenzione all'interno di un'organizzazione. Essi forniscono un'ampia gamma di funzionalità per la pianificazione, l'esecuzione e il monitoraggio delle attività di manutenzione, automatizzando e digitalizzando di fatto gran parte di tutte quelle operazioni che un tempo venivano svolte manualmente dai tecnici negli impianti (Figura 1.1).

Parlando più nello specifico delle funzionalità offerte, in primis troviamo la possibilità di pianificare attività di manutenzione preventive in modo efficiente, con la possibilità di impostare scadenze, pianificando interventi di routine e automatizzando il processo di assegnazione delle attività agli operatori.

Essi forniscono anche una visione dettagliata delle attività in corso, inclusi gli interventi di manutenzione in corso e quelli pianificati. Ciò aiuta a monitorare l'avanzamento delle attività e l'utilizzo delle risorse.

I CMMS agevolano anche la creazione, l'assegnazione e la tracciabilità degli ordini di lavoro permettendo agli operatori di registrare direttamente nel sistema le attività svolte, le risorse impiegate e i materiali utilizzati. Oltre a questo, registrano ed archiviano informazioni dettagliate su tutte le attività di manutenzione passate, inclusi interventi correttivi e preventivi. Questa storicizzazione è utile per l'analisi delle tendenze, la valutazione delle prestazioni e la programmazione futura. I livelli di stock dell'inventario ricambi e dei materiali necessari per le attività di manutenzione vengono costantemente monitorati, facilitandone il riordino e contribuendo ad evitare situazioni di mancanza di materiali critici.

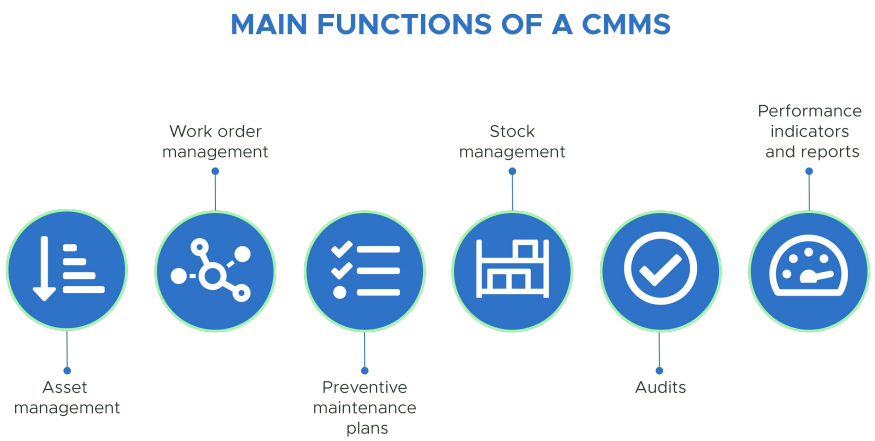


Figura 1.1: Principali funzionalità di un CMMS generico

Solitamente questi sistemi forniscono anche strumenti per generare report dettagliati riguardanti le attività di manutenzione, le prestazioni dell'impianto e altri indicatori chiave di performance (KPI). Questi report sono utili per prendere decisioni informate e ottimizzare le strategie di manutenzione.

Per concludere, un'ultima funzionalità spesso garantita è quella di integrare il CMMS con altri sistemi aziendali, come quelli per la gestione degli asset, la gestione delle risorse umane e la contabilità. Questa integrazione facilita la condivisione di dati critici e garantisce un flusso di informazioni più fluido tra i vari dipartimenti.

Complessivamente, l'implementazione di un CMMS ha il potenziale di migliorare la pianificazione, l'esecuzione e il controllo delle attività di manutenzione, contribuendo a una gestione più efficiente e a una riduzione dei costi nel lungo termine.

CAPITOLO 2

Obiettivi e Requisiti

Questo secondo capitolo della tesi fornisce una panoramica sulle motivazioni alla base del progetto. L'azienda ha sperimentato una crescita significativa in termini di dimensioni e complessità delle operazioni, rendendo evidente la necessità di un approccio più efficiente e centralizzato alla gestione dei dati aziendali. Il progetto mira a risolvere problemi legati alle diverse rappresentazioni dei dati di processo all'interno dell'azienda, distinguendo tra dati di input o gestione e dati di output.

Verrà descritto lo stato dell'arte all'interno di Pansoinco, analizzando le metodologie seguite per la gestione dei dati progettuali. Si evidenziano i passaggi principali che caratterizzano lo svolgimento di un progetto, includendo, anche, le definizioni di Asset e Maintenance Register.

2.1 Motivazioni alla base del progetto

Negli ultimi anni, l'azienda ha sperimentato una crescita significativa sia in termini di dimensioni che di complessità delle sue operazioni. Con un aumento del personale e una diversificazione delle attività, è diventata evidente la necessità di un approccio più efficiente e centralizzato alla gestione dei dati aziendali.

Attualmente, l'azienda sfrutta una serie di sistemi separati e di software per la gestione delle informazioni progettuali, fra i quali spicca per utilizzo Microsoft Excel. Questo approccio frammentato ha portato a una mancanza di coerenza nei dati e ha reso difficile ottenere una visione completa e standardizzata dei dati di processo.

Uno dei problemi che si punta a risolvere con questo progetto è quello legato alle diverse rappresentazioni dei dati di processo che coesistono all'interno dell'azienda.

Parlando più nello specifico delle varie forme con le quali le informazioni vengono rappresentate durante i vari processi interni alla Pansoinco, possiamo fare due macro distinzioni, separando fra quelli che sono dati di input o di gestione e quelli che invece sono i dati in output.

La prima categoria racchiude una serie di standard aziendali che rispecchiano il modo di rappresentare le informazioni da parte del cliente in questione. Ciò vuol dire che, in base al progetto che si sta svolgendo, ma soprattuto per chi lo si sta svolgendo, si avranno delle rappresentazioni parecchio variabili.

Nei fatti i dati che vengono rappresentati condividono dei concetti di base comuni ma che presentano, di volta in volta, una forma che può variare. Questi gradi di etereogeneità possono presentarsi sotto varie forme:

- *Nomenclatura*: spesso possono essere riscontrati nomi diversi per concetti logicamente identici.

- *Struttura:* delle volte ci si imbatte in delle diverse configurazioni dei legami fra le informazioni che però nella pratica rispecchiano situazioni sovrapponibili.
- *Numerosità*: si ha anche una variabilità sotto il punto di vista della mole di queste informazioni, passando da progetti più contenuti, caratterizzati da una numerosità ristretta di dati, a progetti di vasta scala, in cui anche la sola gestione pratica della quantità di informazioni costituisce un problema da non sottovalutare.

Tutti questi aspetti di cui sopra si è discusso rappresenteranno, durante le fasi di sviluppo, uno dei maggiori ostacoli al raggiungimento dell'obiettivo. Non a caso, uno dei punti cardine nello sviluppo del progetto è proprio quello di riuscire a fornire una rappresentazione omogenea delle informazioni, in grado di eliminare le problematiche legate all'etereogeneità.

Per quanto riguarda la seconda tipologia di dati con i quali si avrà a che fare, si parlerà della rappresetazione in SAP.

L'azienda negli anni, oltre alle attività descritte nell'introduzione di questa tesi, si è specializzata nel supporto alla gestione delle operazioni di manutenzione per clienti che sfruttano proprio SAP PM come software gestionale; nei capitoli successivi si illustrerà in maniera approfondita la fase di reverse engineering del modulo Plant Maintenance di SAP.

2.2 Stato dell'arte

Approfondendo ora le metodologie con le quali l'azienda al giorno d'oggi gestisce i dati progettuali, e non potendo fare riferimento ad un unico processo standardizzato, verranno descritti quelli che, nella maggioranza dei casi, sono i procedimenti messi in atto per fornire un piano di manutenzione al cliente finale.

Solitamente quello che succede nelle fasi iniziali di un progetto (dopo il kick-off meeting) è che il cliente fornisce all'azienda una serie di dati di input che rappresentano la definizione del dominio operativo in cui verrà svolto il progetto.

Tutte queste informazioni vengono fornite durante una serie di meeting atti a delineare quello che viene chiamato *Asset Register*.

L'Asset Register è un registro di tutti gli asset fisici posseduti e gestiti da un'organizzazione nell'ambito di un determinato progetto. Gli asset possono includere una vasta gamma di elementi, come proprietà fisiche, attrezzature, veicoli, macchinari, immobili e altri beni di valore. Questo elenco contiene informazioni più o meno dettagliate riguardanti le varie risorse; per ogni asset vengono riportati dati che possono riguardare:

- *La descrizione*: si tratta di una descrizione dettagliata di ogni asset, comprensiva di caratteristiche specifiche, numero di serie, modello, e qualsiasi altra informazione identificativa.
- *Il valore dell'Asset*: si tratta del valore monetario dell'asset, che può includere il costo iniziale, il valore residuo e il valore attuale.
- *La data di Acquisto*: riguarda la data in cui l'asset è stato acquistato o acquisito dall'organizzazione.
- *Localizzazione*: riguarda l'ubicazione fisica dell'asset, che può essere utile per la gestione delle risorse e la manutenzione.
- *Scadenze e Garanzie*: riguardano le date di scadenza per garanzie, manutenzioni preventive pianificate o altri eventi importanti relativi all'asset.

- *Responsabile*: è l'individuo o il dipartimento responsabile della gestione e della manutenzione dell'asset.
- *Numero di Inventario o Identificatore Unico*: si tratta di un numero di inventario o un identificatore unico assegnato all'asset per facilitare la sua identificazione.
- *Storia dell'Utilizzo*: è il registro delle attività legate all'utilizzo dell'asset nel tempo, che può includere trasferimenti, assegnazioni, manutenzioni e altro.
- *Dati Fiscali e Amministrativi*: si tratta di informazioni fiscali e amministrative necessarie per la dichiarazione fiscale e la conformità normativa.

Bisogna tenere in considerazione che, gestendo progetti che possono avere durate molto variabili (anche di svariati anni), questo registro non è praticamente mai un elemento che resta invariato nel tempo, ma subisce, ad ogni revisione, delle operazioni di aggiornamento; ciò avviene in quanto su di esso verrà poi basata tutta la definizione delle attività di manutenzione; di conseguenza, è di fondamentale importanza riuscire ad avere una visione realistica ed esente da errori della situazione degli asset del cliente al quale si sta offrendo un servizio.

Lo step successivo nell'esecuzione di un progetto, una volta definito l'insieme di asset, riguarda la definizione delle operazioni di manutenzione per ognuno di essi. Questi studi vengono svolti dal reparto ingegneristico interno all'azienda e si basano su una serie di analisi riguardanti gli asset. Queste analisi riguardano le caratteristiche, la tipologia e le proprietà di ogni singolo oggetto di interesse.

Due strumenti di fondamentale importanza utilizzati nell'ambito della definizione dei piani di manutenzione sono le analisi *RAM* ed *FMECA*. Queste verranno illustrate in dettaglio nelle prossime sottosezioni.

Analisi RAM

L'analisi RAM, o Reliability, Availability, and Maintainability (Affidabilità, Disponibilità e Manutenibilità), è una metodologia di valutazione e misurazione delle prestazioni di un sistema, attrezzatura o processo. Questa analisi è fondamentale per comprendere il comportamento di un sistema nel tempo e la sua capacità di svolgere le funzioni richieste. Più in dettaglio, l'analisi RAM fornisce una valutazione quantitativa e qualitativa di diversi aspetti chiave riguardanti un asset, tra cui:

- *Reliability (Affidabilità)*: l'affidabilità di un sistema si riferisce alla sua capacità di svolgere una determinata funzione senza guasti o interruzioni per un periodo di tempo specifico. L'analisi RAM esamina la probabilità di guasto del sistema nel tempo, identificando potenziali punti deboli e valutando la resistenza del sistema a malfunzionamenti.
- *Availability (Disponibilità)*: la disponibilità è la misura della capacità di un sistema di essere operativo e funzionante quando richiesto. L'analisi RAM valuta il tempo durante il quale il sistema è effettivamente disponibile per l'utilizzo, tenendo conto dei tempi di inattività pianificati e non pianificati. Ciò include la gestione di guasti e la riparazione del sistema per minimizzare i tempi di fermo.
- *Maintainability (Manutenibilità)*: la manutenibilità si riferisce alla facilità con cui un sistema può essere riparato o mantenuto in modo efficace. L'analisi RAM esamina la complessità delle operazioni di manutenzione, la disponibilità di parti di ricambio e la capacità di riparare il sistema in modo tempestivo. Questo aspetto è cruciale per garantire tempi di ripristino rapidi e una gestione efficiente delle risorse di manutenzione.

L'analisi RAM coinvolge spesso l'utilizzo di dati storici, test di affidabilità e dati di manutenzione per costruire modelli predittivi. Questi modelli possono essere utilizzati per stimare la probabilità di guasto, i tempi di riparazione e altri parametri critici. Inoltre, l'analisi può essere eseguita sia a livello del singolo componente che a livello di sistema completo, consentendo una visione olistica delle prestazioni del sistema.

Analisi FMECA

In aggiunta all'analisi RAM, come accennato in precedenza, gli ingegneri della manutenzione, al fine di definire le attività manutentive per ogni asset, fanno affidamento anche sull'analisi FMECA.

La Failure Modes, Effects, and Criticality Analysis (Analisi delle Modalità di Guasto, degli Effetti e della Criticità) è un approccio metodologico utilizzato per identificare, valutare e comprendere le modalità di guasto di un sistema, gli effetti di tali guasti e la loro criticità per il funzionamento complessivo di un asset.

Questa analisi si basa sulla valutazione sistematica dei possibili guasti che potrebbero verificarsi in un sistema, esaminando come ciascuno di essi potrebbe influenzare il sistema nel suo complesso e valutando il loro impatto sulla sicurezza, l'affidabilità e le prestazioni del sistema.

Durante l'analisi FMECA, vengono esaminate le seguenti fasi:

- *Identificazione delle Modalità di Guasto*: si identificano tutte le possibili modalità di guasto che potrebbero verificarsi all'interno del sistema. Queste modalità possono riguardare componenti specifici, processi o funzionalità del sistema.
- *Analisi degli Effetti*: per ciascuna modalità di guasto identificata, si analizzano gli effetti che essa potrebbe avere sul sistema. Questi effetti possono includere malfunzionamenti, perdite di prestazioni, danni a componenti o, in alcuni casi, conseguenze più gravi, come incidenti o danni alla sicurezza.
- *Determinazione della Criticità*: si valuta la criticità di ciascuna modalità di guasto, considerando la sua probabilità di verificarsi, la sua rilevanza in termini di impatto e la gravità degli effetti associati. La criticità riflette quanto un guasto potrebbe influire negativamente sul sistema e sulle sue prestazioni.
- *Sviluppo di Strategie di Mitigazione*: sulla base dei risultati dell'analisi vengono sviluppate strategie di mitigazione per ridurre al minimo gli effetti critici delle modalità di guasto. Ciò può includere miglioramenti progettuali, l'implementazione di sistemi di backup o di sicurezza oppure la creazione di procedure di intervento in caso di guasto.

Per portare a termine queste analisi viene utilizzato un gran numero di strumenti di supporto; di seguito verrà fornita una rapida panoramica di quelli di più comune e diffuso utilizzo.

Fault Tree Analysis (FTA)

L'analisi dell'albero dei guasti offre la possibilità di esaminare le cause dei guasti, identificando gli eventi di base che possono condurre a un guasto di maggiore entità, noto come "evento principale" (top event). Questi eventi possono essere categorizzati come guasti primari, intermedi o secondari.

Mediante la rappresentazione a forma di albero dell'impianto in questione o di una sua sezione, l'analisi evidenzia le interconnessioni fisiche e logiche tra i vari componenti che costituiscono l'impianto (Figura 2.1). L'obiettivo è individuare le connessioni strutturali che, ad esempio, hanno portato da un guasto di minore entità a uno di maggiore gravità. In

questo modo, siamo in grado di comprendere le dinamiche sottostanti che contribuiscono allo sviluppo e all'escalation dei guasti nell'impianto, fornendo, così, un quadro dettagliato delle relazioni tra gli eventi e le loro conseguenze.

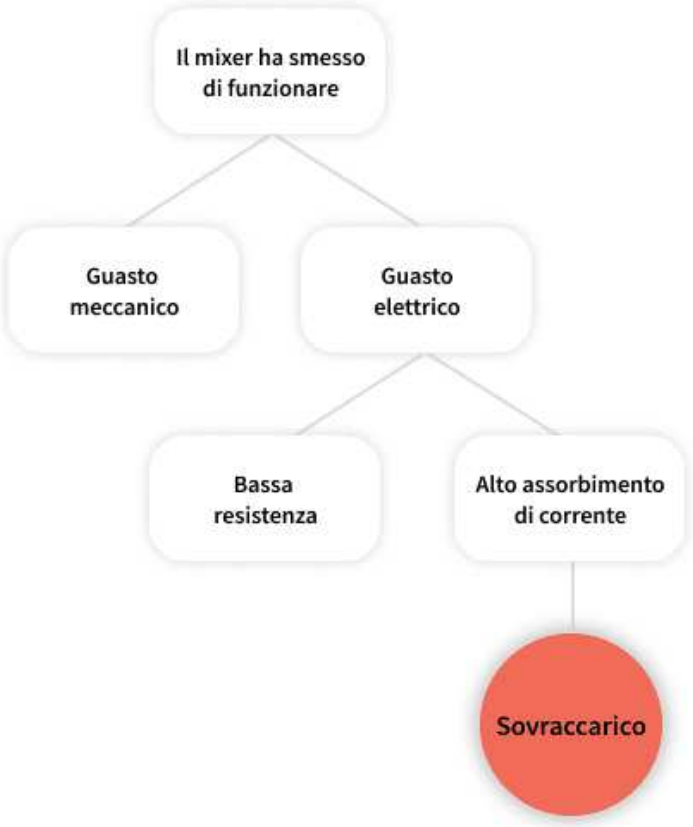


Figura 2.1: Esempio semplificativo di un Fault Tree

Reliability Block Diagram (RBD)

Questa analisi si basa su un diagramma a blocchi che, analogamente alla fault tree analysis, consente di visualizzare un impianto come una complessa rete logica. Tale rete è composta da blocchi interconnessi tramite nodi, dove ogni blocco rappresenta un componente del sistema.

Suddividendo l'impianto in blocchi o sotto-sistemi e assegnando a ognuno di essi un indice di affidabilità, si è in grado di calcolare l'affidabilità complessiva dell'impianto. Questo calcolo si basa sui valori di affidabilità attribuiti ai singoli elementi, fornendo, così, una valutazione dettagliata della sicurezza e dell'affidabilità dell'intero sistema. Nella figura 2.2 viene riportato un esempio di RBD

Risk Priority Number (RPN)

Il Risk Priority Number permette di associare a ciascuna tipologia di guasto un indice basato sulla gravità degli effetti generati sul sistema dal guasto stesso.

Viene largamente utilizzato nell'analisi FMECA, introducendo un metodo per stabilire le priorità tramite la valutazione del rischio. Nella figura 2.3 viene riportato un esempio di matrice usata per attribuire l'RPN.

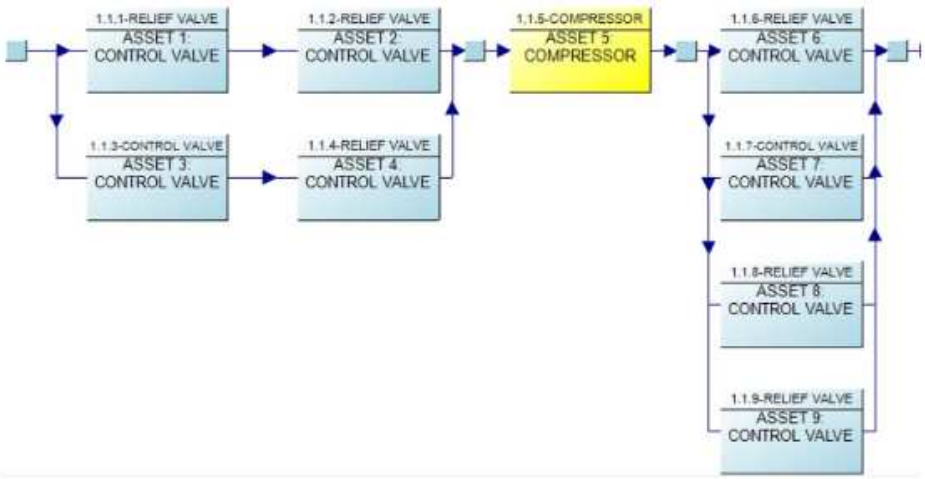


Figura 2.2: Esempio di un RBD

Figura 2.3: Esempio di matrice usata per attribuire l'RPN

Una volta che, grazie all'applicazione di queste tecniche, si riesce ad avere un quadro completo su quelle che saranno le attività di controllo e manutenzione per ognuno degli asset, si passa alla stesura del piano di manutenzione vero e proprio.

A tal proposito è di fondamentale importanza introdurre un elemento, strettamente legato all'asset register, chiamato *Maintenance Register*. Si tratta di uno strumento essenziale per la gestione efficace delle attività di manutenzione all'interno dell'organizzazione. Nella pratica ha la forma di un documento dettagliato e organizzato che funge da registro centrale per tutte le informazioni pertinenti relative agli asset fisici gestiti dall'azienda e alle attività di manutenzione ad essi associate.

Questo registro svolge un ruolo fondamentale nel garantire la corretta manutenzione degli asset, la pianificazione di interventi preventivi e correttivi, nonché la tracciabilità delle attività passate.

Il maintenance register può essere strutturato in modo da includere diverse sezioni chiave per coprire tutti gli aspetti critici delle attività di manutenzione. Una sezione cruciale del maintenance register è dedicata allo storico delle attività di manutenzione, in cui vengono registrati tutti gli interventi eseguiti, inclusi quelli correttivi e preventivi. Ogni voce può contenere informazioni sulla data dell'intervento, la natura del problema riscontrato, le azioni correttive adottate e i tempi di inattività associati. Questo storico consente di avere

una panoramica chiara delle performance passate degli asset.

Nella sezione di pianificazione delle attività, vengono registrate tutte le attività di manutenzione preventive, le quali potrebbero includere cicli di manutenzione regolari, scadenze per la sostituzione di componenti critici e altre attività programmabili. Solitamente nel registro sono anche incluse informazioni relative all'utilizzo degli asset nel tempo, includendo informazioni su trasferimenti, assegnazioni e altri eventi rilevanti. Monitorare i dati di utilizzo e le prestazioni è essenziale per valutare l'efficacia degli asset nel contesto delle attività aziendali.

Un'ulteriore informazione che questo oggetto può includere riguarda la responsabilità della gestione di ciascun asset. Chi è responsabile della manutenzione di un determinato asset, ad esempio un dipartimento o un individuo, può essere registrato per garantire una chiara attribuzione delle responsabilità.

Per concludere, nella maggior parte dei casi viene inclusa anche la documentazione fiscale e amministrativa, al fine di garantire la conformità normativa e una corretta dichiarazione fiscale; questi dati potrebbero riguardare informazioni come la data di acquisto, il valore di acquisizione o altre considerazioni di questo genere.

Si può dire, quindi, che il maintenance register, in maniera analoga al registro degli asset, è un elemento che evolve con il progetto stesso, includendo un numero di informazioni sempre crescente, man mano che gli eventi si susseguono.

Tornando, ora, a discutere del piano di manutenzione, esso rappresenta una sottocomponente del maintenance register; in un certo modo è possibile intenderlo come il blocco di partenza dal quale poi il registro viene costruito ed aggiornato.

La forma che il maintenance plan può assumere varia in base alle necessità e alle specifiche fornite dal cliente finale; infatti, in base allo standard e alla gestione interna adottati dall'organizzazione di turno, il piano potrebbe avere una facciata molto variabile. Come accennato nelle sezioni precedenti, è proprio qui che si trova uno dei problemi più rilevanti di questo progetto, ovvero riuscire a trovare una metodologia di rappresentazione, ovvero una struttura, utilizzabile nel maggior numero di casi possibili.

Per rendere più tangibile il problema sopra esposto verranno ora riportati dei frammenti di due piani di manutenzione. In particolare, si considerino le figure 2.4 e 2.5.

Group	Group Counter	Main Work Center	Operation	Plant	Description	Duration	1st Package	2nd
validation table 15				Validation Table 9			Validation Table 12-14	Validatio
8	2	8	4	4		7	1M	2M
PLNNR	PLNAL	ARBPL	VORNR	WERKS		DAUNO	MARK01	M
TAM07	1	IITETECZ	10	PTSH	3M -CCTV equipr	150	3M	
TQZ07	1	IITETECZ	10	PTSH	1Y-LAMP/RELAY (	36		
TQN07	1	IITETECZ	10	PTSH	3QN51LEPHONE	269	3M	
TQQ07	1	IITETECZ	10	PTSH	6M-VISUAL UDIB	96	3M	
TQZ07	2	IITETECZ	10	PTSH	6M–TELEPHONE	40	3M	
TQU07	1	IITETECZ	10	PTSH	1Y-FUNCTIONAL .	60		
TQU07	1	IITETECZ	20	PTSH	3Y-FUNCTIONAL .	120		
IIR07	E2	IMVMTECZ	10	PTSH	3M-ESDV-FC	8	3M	
IIR07	E2	IMVMTECZ	20	PTSH	6M-PRESSURE CC	8		6M
IIR07	E2	IMVMTECZ	30	PTSH	1Y-ESDV-PNEUM	20		
IIR07	D9	IMVMTECZ	10	PTSH	3M-ESDV-FC	2	3M	
IIR07	D9	IMVMTECZ	20	PTSH	6M-PRESSURE CC	2		6M
IIR07	D9	IMVMTECZ	30	PTSH	1Y-ESDV-PNEUM	5		
IIR07	E3	IMVMTECZ	10	PTSH	3Y-CONTROL VAL	8		
IIR07	E4	IMVMTECZ	10	PTSH	3Y-PRESSURE COI	60		
IIR07	E4	IMVMTECZ	20	PTSH	6M-PRESSURE CC	15		6M

Figura 2.4: Piano di manutenzione: Esempio 1

Come è possibile osservare dalle figure, in entrambi i piani di manutenzione abbiamo informazioni quasi sovrapponibili ma caratterizzate da una forma differente.

Facendo alcuni esempi, nel primo caso i dati seguono quello che viene chiamato *Standard ENI* (basato su un'interpretazione della struttura interna di SAP) in cui le operazioni di manutenzione sono raggruppate in cicli di manutenzione; ogni operazione sarà, quindi,

5	JP Code	JP Description	Instruction Type	Frequency	Main Skill	People	Duration
22	EX002E	A.C. Electric Motor	Functional verification	3 months / 2000 h	ELEC	2	4
23	PSV01E	Pressure Safety	Major intervention	36 months	INST	1	2
24	SDV01E	Shut Down Valve	Major intervention	60 months	INST	2	10
25	SDV01E	Shut Down Valve	Intermediate intervention	12 months	INST	2	2
26	SDV01E	Shut Down Valve	Intermediate intervention	24 months	INST	2	2
27	SDV01E	Shut Down Valve	Functional verification	6 months	INST	2	1
28	VN001E	Knock out drum / G	Major intervention	60 months	INSP	2	6
29	VN001E	Knock out drum / G	Major intervention	60 months	MECH	5	36
30	VN001E	Knock out drum / G	Functional verification	12 months	INSP	1	4
31	XCV01E	Control Valve	Major intervention	36 months	INST	4	8
32	PSV01E	Pressure Safety	Major intervention	36 months	INST	2	4
33	SDV01E	Shut Down Valve	Major intervention	60 months	INST	2	20
34	SDV01E	Shut Down Valve	Intermediate intervention	12 months	INST	2	2
35	SDV01E	Shut Down Valve	Intermediate intervention	24 months	INST	2	2
36	SDV01E	Shut Down Valve	Functional verification	6 months	INST	2	1
37	XS001E	Switch	Major intervention	12 months	INST	2	2
38	EX002E	A.C. Electric Motor	Major Intervention	24 months or 8000 H	ELEC	2	12
39	EX002E	A.C. Electric Motor	Functional verification	3 months	ELEC	2	4
40	VS002E	Gas Separator / G	Major intervention	60 months	INSP	2	6
41	VS002E	Gas Separator / G	Major intervention	60 months	MECH	5	36
42	VS002E	Gas Separator / G	Functional verification	12 months	INSP	1	4
43	RG001E	Manual / Motorized	Major intervention	24 months	ELEC	2	8

Figura 2.5: Piano di manutenzione: Esempio 2

identificata da un ciclo (Group + Group Counter) e da un numero di operazione (Identificatore dell'operazione nel ciclo).

Nel secondo caso, invece viene seguito lo standard *JP*, in cui non si hanno più i cicli a raggruppare le operazioni, ma bensì i cosiddetti jobplan.

Oltre a questo è anche possibile osservare etereogeneità legate alla nomenclatura; infatti, l'informazione che nel primo esempio è riportata sotto la colonna "Main Work Center" è semanticamente sovrapponibile a ciò che nella seconda tabella è chiamata "Main Skill".

Un altro punto di discrepanza fra i due esempi sta nel modo di rappresentare la frequenza con la quale le operazioni di manutenzione dovranno essere svolte.

Nello standard ENI (primo esempio) si ha una rappresentazione per "pacchetti di frequenza" dove, nel secondo caso, si ha una rappresentazione molto più diretta, tramite un campo stringa che a parole descrive ogni quanto bisognerà effettuare l'operazione.

Quanto esposto sopra è solo un esempio di ciò a cui si fa riferimento quando si parla di etereogeneità delle rappresentazioni dei dati; quest'ultima problematica può portare a diverse complicanze all'interno di un azienda.

Non riuscendo ad avere una visione unificata e standardizzata dell'informazione, le diverse interpretazioni dei dati potrebbero causare fraintendimenti ed errori di comunicazione, influendo negativamente sulla coesione e sull'allineamento organizzativo; ancora peggio, avendo a che fare con progetti diversi, la difficoltà che si incontrerebbe nell'integrare questi sistemi potrebbe portare ad una limitazione della capacità dell'organizzazione di sfruttare appieno i vantaggi delle tecnologie moderne, come l'Intelligenza Artificiale, l'analisi dei dati e l'automazione dei processi.

Immaginando un altro scenario, se l'azienda un giorno dovesse decidere di passare a nuovi sistemi o piattaforme, l'eterogeneità dei dati andrebbe a complicare notevolmente il processo di migrazione. La trasformazione e l'adattamento dei dati da un formato all'altro potrebbero richiedere sforzi significativi e comportare il rischio di perdita o corruzione delle informazioni.

Tuttavia, la conseguenza più comune, e per questo impattante, di questa mancanza di standardizzazione è l'aumento del rischio di errori umani durante le operazioni di manipolazione ed interpretazione dei dati. Questi errori possono avere conseguenze significative sulle tempistiche, sulla qualità dell'output e, quindi, più in generale, sulla soddisfazione del

cliente.

2.3 Introduzione alla soluzione proposta

Come accennato nella sezione precedente, le maggiori criticità che la progettazione discussa in questa tesi proverà a risolvere sono in gran parte legate alla diversità introdotta dalle varie rappresentazioni dei dati che coesistono all'interno dell'azienda. In comune abbiamo la sola rappresentazione dei dati SAP che, in molti casi, costituisce anche l'output diretto dei processi di creazione dei piani manutentivi. Fissando questa struttura come punto di partenza, verrà svolta una fase di reverse engineering atta alla mappatura di tutta la complessa gestione dei dati messa in atto da SAP; una volta ottenuta una mappatura ben chiara, si procederà selezionando le componenti di questo schema che rappresentano concetti comuni e riscontrabili anche nella gestione interna dei progetti aziendali.

Ciò, tuttavia, non sarà sufficiente; di fatto non si può dire che la rappresentazione dei dati in SAP è un super-set di tutti quelli che sono gli standard aziendali. Nonostante l'estrema complessità e flessibilità della sua rappresentazione, SAP non presenta al suo interno dei concetti che, invece, possono essere riscontrati in molti dei progetti portati avanti dall'azienda.

Tutto questo per dire che, al fine di ottenere quella che sarà la *struttura informativa* finale, a questo sub-set del database SAP dovranno necessariamente essere aggiunti dei concetti chiave a cui si fa comunemente riferimento nella realtà aziendale.

Per concludere, quindi, non ci si potrà accontentare di sviluppare una base dati che ricalca un'ombra di SAP, ma, partendo da esso, dovrà essere messo in atto, dopo la fase di mappatura, un processo di estensione finalizzato al raggiungimento della completa rappresentatività e sovrapponibilità del modello ricavato ai dati progettuali su cui l'azienda basa il proprio lavoro.

I punti di discrepanza fra le due macro-rappresentazioni a cui precedentemente si faceva riferimento porteranno alla necessità di progettare una *componente di data-exchange* in grado di controllare, adattare ed indirizzare i dati in transito da e verso la base dati ottenuta come output della prima parte del progetto.

Una volta stabilita questa rappresentazione portante, che costituirà le fondamenta di tutto il sistema, si svilupperà un modulo di *interfaccia* in grado di fornire una chiara vista sulle informazioni contenute in tale schema. Si sta parlando, nei fatti, di un sistema CRUD; questo acronimo sta ad indicare le operazioni fondamentali che un'interfaccia software avente a che fare con dei dati deve svolgere. Esse sono:

- *Create (Creazione)*: consente di inserire nuovi dati nel sistema; corrisponde alla creazione di un nuovo record del database.
- *Read (Lettura)*: consente di leggere o recuperare informazioni esistenti dal sistema.
- *Update (Aggiornamento)*: consente di modificare i dati esistenti nel sistema.
- *Delete (Cancellazione)*: consente di rimuovere dati dal sistema.

Riassumendo questo breve excursus introduttivo riguardante il lavoro svolto ed illustrato in questa trattazione, pè possibile identificare in linea di massima, tre macro-step che porteranno al raggiungimento della soluzione finale:

- *Fase di reverse engineering* , atta alla mappatura di SAP e all'estensione di tale struttura per garantire una rappresentazione pienamente coerente con gli standard aziendali riguardanti la rappresentazione dell'informazione.

- *Progettazione di un modulo di data exchange*, mirato alla gestione dei dati in ingresso/uscita dal nostro sistema ed incaricato dell'automatizzazione dei processi di controllo ed adattamento.
- *Progettazione di un sistema CRUD*, in grado di fungere da interfaccia diretta ai dati presenti nel sistema.

Parlando ora dei *pro e dei contro* a cui una soluzione complessiva di questo tipo potrebbe portare, bisogna porla a confronto con quello che è lo stato dell'arte odierno in azienda. L'uso di un supporto CMMS invece che Excel (il quale rappresenta il software di più diffuso utilizzo) comporta vantaggi e svantaggi, e il peso di ognuno di essi varia in base a quelle che sono le esigenze aziendali (spesso legate al livello di strutturazione e dimensione). Alcuni di questi pro e contro riguardanti l'utilizzo di Excel potrebbero essere:

Vantaggi:

- *Flessibilità*: Excel è noto per la sua flessibilità e adattabilità. Può essere utilizzato per una vasta gamma di scopi, inclusa la gestione di dati di manutenzione.
- *Accessibilità*: la maggior parte delle persone è familiare con Excel, quindi l'apprendimento e l'utilizzo sono spesso intuitivi. Excel non richiede necessariamente una formazione specifica.
- *Personalizzazione*: gli utenti possono progettare fogli di calcolo personalizzati per soddisfare le loro specifiche esigenze di gestione delle attività di manutenzione.
- *Costi Iniziali*: non ci sono costi di licenza iniziali associati all'utilizzo di Excel per la gestione della manutenzione.

Svantaggi:

- *Limiti nella Scalabilità*: Excel può diventare inefficace quando si tratta di gestire grandi quantità di dati o un gran numero di attività di manutenzione.
- *Collaborazione Complessa*: la collaborazione in tempo reale su fogli di calcolo può essere complessa. Le versioni multiple possono causare confusione e errori.
- *Sicurezza*: la sicurezza dei dati in Excel è limitata rispetto ai sistemi CMMS, e la protezione delle informazioni sensibili potrebbe essere problematica.

D'altra parte, per quanto riguarda i pro e i contro nell'utilizzo di un sistema di supporto ad un CMMS, questi potrebbero essere come di seguito specificato:

Vantaggi:

- *Gestione Centralizzata*: un CMMS consente di gestire tutte le attività di manutenzione in un unico sistema centralizzato, migliorando l'efficienza e la visibilità.
- *Automazione dei Processi*: i CMMS offrono automazione per molte attività, come la pianificazione delle manutenzioni preventive, la gestione delle risorse e la registrazione automatica dei dati.
- *Reporting e Analisi*: i CMMS forniscono robusti strumenti di reporting e analisi, consentendo un monitoraggio più efficace delle prestazioni e la presa di decisioni informate.
- *Collaborazione Facilitata*: la collaborazione tra membri del team è semplificata, condividendo facilmente informazioni e aggiornamenti in tempo reale.

Svantaggi:

- *Costi Iniziali e Implementazione*: l'implementazione di un CMMS può richiedere un investimento significativo iniziale in termini di tempo e risorse finanziarie.
- *Apprendimento Iniziale*: potrebbe essere necessario un periodo di apprendimento per gli utenti prima che questi diventino completamente proficienti nell'utilizzo di un nuovo sistema CMMS.
- *Personalizzazione Limitata*: in alcuni casi, la personalizzazione avanzata potrebbe richiedere competenze tecniche specifiche e potrebbe non essere così flessibile come Excel per alcune esigenze specifiche.

In conclusione, la scelta tra Excel e un CMMS dipende dalle dimensioni dell'organizzazione, dalla complessità delle attività di manutenzione e dalle risorse disponibili. Se, da una parte, Excel può essere un'opzione più immediata e adatta per piccole operazioni, un CMMS offre vantaggi decisamente più significativi in termini di organizzazione, automazione e analisi per organizzazioni più complesse e con una maggiore quantità di dati e attività da gestire.

CAPITOLO 3

Attività di Reverse Engineering

In questo terzo capitolo verranno illustrate le fasi di ricerca che hanno fornito la base di partenza per la definizione del database centrale. Ci si concentrerà unicamente sulla fase di reverse engineering del database interno a SAP PM, il quale ha consentito di avere un riferimento chiaro durante lo sviluppo. Illustrando le tabelle di maggiore interesse, divise per Asset e Maintenance Register, verranno introdotti i concetti basilari della gestione SAP per quanto riguarda il mondo della Plant Maintenance. Questi elementi saranno poi utilizzati durante la fase di sviluppo come blocchi di partenza, la cui combinazione porterà alla definizione della base di dati centrale del progetto.

3.1 Introduzione al reverse engineering

Il reverse engineering è un processo complesso e intrigante che si propone di esaminare attentamente, scomporre e comprendere un oggetto, un sistema o un software già esistente. Contrariamente allo sviluppo tradizionale, dove l'attenzione è posta sulla creazione di qualcosa di nuovo, il reverse engineering si concentra sulla destrutturazione di ciò che esiste per analizzarne il funzionamento interno. Questa pratica può essere applicata a una vasta gamma di campi, tra cui l'ingegneria del software, l'elettronica, la meccanica, la sicurezza informatica, e molto altro.

Nel contesto dell'ingegneria del software, il reverse engineering coinvolge l'analisi di un programma o di un'applicazione per comprendere il suo codice sorgente, la sua struttura interna e la sua logica di funzionamento.

Questo può essere utile per vari scopi, come la manutenzione di software legacy, l'indagine forense digitale, l'identificazione di vulnerabilità di sicurezza e la comprensione di come funzionano applicazioni o dispositivi per la creazione di alternative o miglioramenti.

Uno degli aspetti centrali del reverse engineering è la disassemblazione del codice binario, che è il linguaggio di basso livello che un computer comprende direttamente. Questo processo è cruciale per ottenere una visione dettagliata di come un software interagisce con il sistema sottostante.

Attraverso la disassemblazione, gli ingegneri possono esaminare le istruzioni del codice macchina, identificare algoritmi implementati e capire come i dati vengono elaborati.

Il reverse engineering non è limitato solo al mondo del software. Nell'ambito dell'ingegneria elettronica, ad esempio, può essere applicato per analizzare dispositivi hardware e circuiti. In meccanica, può essere utilizzato per comprendere la struttura interna di un oggetto fisico, come, ad esempio, una macchina o un veicolo.

Tuttavia, il reverse engineering solleva questioni etiche e legali, poiché può essere utilizzato anche per violare la proprietà intellettuale o la sicurezza di un sistema. Ciò ha portato a una continua riflessione su come regolamentare questa pratica in diversi settori e giurisdizioni.

Un altro aspetto fondamentale del reverse engineering è la sua connessione con l'innovazione. Analizzando e comprendendo i prodotti esistenti, gli ingegneri possono ideare nuove soluzioni, miglioramenti o applicazioni alternative. Questo processo può stimolare l'innovazione e contribuire al progresso tecnologico, creando un ciclo di apprendimento continuo.

In conclusione, il reverse engineering rappresenta un potente strumento che offre un'opportunità unica di esplorare e comprendere il mondo tecnologico esistente. È un processo che richiede competenze approfondite, integrità etica e una visione critica, e il suo impatto si estende a diversi settori, contribuendo a plasmare il futuro delle tecnologie che ci circondano.

3.2 SAP

SAP, acronimo di Systems, Applications, and Products in Data Processing, è una delle più grandi e influenti aziende al mondo specializzate nello sviluppo di soluzioni software per il business.

Fondata in Germania nel 1972 da cinque ex dipendenti di IBM, l'azienda ha rapidamente assunto un ruolo di primo piano nel settore della tecnologia dell'informazione, fornendo strumenti e piattaforme software avanzate per migliorare l'efficienza e l'efficacia delle operazioni aziendali.

Al cuore della vasta gamma di prodotti offerti da SAP si trova *SAP ERP*, una suite di applicazioni software integrata progettata per supportare e ottimizzare una vasta gamma di processi aziendali. Questo sistema è costruito su un'architettura modulare che consente alle aziende di selezionare e implementare solo i moduli necessari alle proprie esigenze specifiche. La modularità è una caratteristica chiave di SAP ERP, consentendo una flessibilità significativa nelle implementazioni e negli adattamenti ai cambiamenti nelle operazioni aziendali.

Un elemento fondamentale di SAP ERP è il modulo Finanze (SAP FI), che gestisce tutte le transazioni finanziarie, la contabilità e i processi di reporting. Questo modulo fornisce un quadro completo e accurato delle attività finanziarie dell'azienda, consentendo una migliore pianificazione e gestione delle risorse finanziarie.

Inoltre, SAP ERP offre un modulo dedicato alla gestione delle risorse umane (SAP HR/HCM), che copre aspetti chiave come il reclutamento, lo sviluppo e l'amministrazione del personale, contribuendo a ottimizzare la gestione delle risorse umane.

Il modulo di Produzione (SAP PP) è progettato per coordinare e ottimizzare i processi di produzione, dalla pianificazione alla gestione delle risorse. Questo modulo consente alle aziende di raggiungere una produzione più efficiente e di adattarsi rapidamente alle variazioni nella domanda o nelle risorse disponibili.

L'integrazione con il modulo di Gestione della Catena di Approvvigionamento (SAP SCM) estende questa ottimizzazione alla gestione end-to-end della catena di approvvigionamento, migliorando la visibilità e la pianificazione.

Il modulo di Vendite e Distribuzione (SAP SD) facilita le attività di vendita, dalla gestione dei contatti con i clienti alla consegna dei prodotti. Questo modulo contribuisce a migliorare l'efficienza del processo di vendita, ottimizzando la gestione dell'inventario e la distribuzione dei prodotti.

Inoltre, SAP ERP fornisce moduli dedicati alla gestione di magazzino e logistica (SAP WM), garantendo un controllo completo e ottimizzato delle attività di gestione di magazzino.

La piattaforma SAP ERP si estende anche alla gestione dei progetti con il modulo SAP PS, che supporta la pianificazione, l'esecuzione e il monitoraggio dei progetti aziendali. Questo modulo è essenziale per le aziende coinvolte in progetti complessi e richiede una visione chiara delle risorse e delle attività.

Un elemento distintivo di SAP ERP è la sua capacità di offrire una visione integrata delle operazioni aziendali. L'integrazione tra i vari moduli consente la condivisione di dati in tempo reale, eliminando la necessità di procedure manuali e riducendo il rischio di errori. Questa visibilità end-to-end è fondamentale per la presa di decisioni informate e la gestione strategica delle risorse aziendali.

SAP ERP può essere implementato sia on-premise che su cloud, offrendo alle aziende la flessibilità di scegliere la modalità di implementazione che meglio si adatta alle proprie esigenze e alle condizioni di business in evoluzione. L'implementazione su cloud consente, inoltre, un accesso più flessibile e scalabile alle risorse e ai dati aziendali. Nel corso degli anni, SAP ha continuato a innovare e adattarsi alle nuove esigenze del mercato. Ha ampliato la sua offerta con soluzioni avanzate di Business Intelligence (SAP BI/BW) per l'analisi e la visualizzazione dei dati aziendali, contribuendo alla presa di decisioni basata sui dati.

Si può dire, quindi, che SAP ERP rappresenta un pilastro fondamentale per molte aziende in tutto il mondo, offrendo una suite di applicazioni integrata e flessibile che supporta una vasta gamma di processi aziendali. La sua presenza capillare in diverse industrie è testimone della sua efficacia nel migliorare l'efficienza operativa e consentire alle aziende di adattarsi dinamicamente a un ambiente aziendale in continua evoluzione.

Il principale strumento utilizzato per lo sviluppo all'interno dell'ambiente SAP è *ABAP* (Advanced Business Application Programming); si tratta di un linguaggio di programmazione sviluppato per la personalizzazione e l'estensione delle applicazioni SAP ed è ampiamente utilizzato per la creazione di report, formulari, interfaccia utente, workflow e altre funzionalità personalizzate all'interno del sistema ERP.

ABAP viene utilizzato per personalizzare ed estendere le funzionalità delle applicazioni SAP in base alle esigenze specifiche di un'organizzazione. Questo può includere la creazione di campi aggiuntivi, la modifica di layout di schermate, la definizione di nuovi report, etc.

ABAP contempla anche la scrittura di programmi che estraggono, analizzano e presentano dati da diverse parti del sistema SAP. Ciò consente agli utenti di generare report personalizzati e analizzare le informazioni aziendali in modi specifici. Questo linguaggio è anche coinvolto nella creazione di schermate e formulari utente all'interno dell'ambiente SAP. Ciò include la progettazione di layout, la gestione degli eventi utente e l'interazione con gli elementi dell'interfaccia utente. ABAP, per di più, facilita l'integrazione di sistemi SAP con altre applicazioni o sistemi esterni. Ciò consente un flusso continuo di dati e informazioni tra diversi sistemi aziendali.

Gli sviluppatori ABAP sono spesso coinvolti nella manutenzione e nel supporto delle applicazioni SAP, garantendo che le personalizzazioni e le estensioni continuino a funzionare correttamente con le nuove versioni del software.

3.3 SAP PM - Mappatura delle tabelle e relazioni

Fra tutti i moduli offerti da SAP quello sul quale è stata focalizzata la fase di reverse engineering di questo progetto è stato *SAP Plant Maintenance*.

Questo modulo consente alle aziende di pianificare, eseguire e monitorare tutte le attività legate alla manutenzione degli impianti, delle attrezzature e delle risorse. SAP PM fornisce strumenti per la gestione dei cicli di vita degli asset, la pianificazione delle attività di manu-

tenzione preventiva e correttiva, la registrazione delle attività di manutenzione e l'analisi delle prestazioni degli asset. Integrato con altri moduli SAP ERP, contribuisce a mantenere gli impianti operativi in modo efficiente e affidabile.

Entrando ora più nello specifico ed iniziando a discutere della fase di studio, avente come obiettivo la piena comprensione della struttura interna di SAP PM, si comincerà fornendo una panoramica sulla metodologia generale seguita per portare a termine questo compito.

Prima di tutto, c'é da dire che la licenza SAP PM in possesso alla sede aziendale di Ancona in cui è stato svolto il progetto permette l'accesso alla GUI SAP da un unico portatile sempre presente negli uffici dell'organizzazione.

Il primo step di questa fase di reverse engineering è stato proprio quello di studiare le caratteristiche generali della base di dati interna a SAP, chiamata *SAP HANA*.

SAP HANA, acronimo di High-performance ANalytic Appliance, costituisce un database multimodello che si distingue per la memorizzazione dei dati direttamente nella sua memoria anziché su un tradizionale disco. La sua architettura in-memory, focalizzata sulle colonne, consente l'esecuzione di analisi avanzate e transazioni ad alta velocità all'interno di un unico sistema. Questa caratteristica riveste un'importanza fondamentale poiché consente alle aziende di elaborare considerevoli volumi di dati con una latenza prossima allo zero, di effettuare interrogazioni istantanee e di adottare un approccio veramente basato sui dati.

Lanciato nel 2010, SAP HANA non è soltanto un DBMS, ma è una soluzione completa e consolidata. Oltre al suo ruolo di server di database per l'archiviazione e il recupero dei dati richiesti dalle applicazioni, SAP HANA offre avanzate funzionalità di ricerca, analisi e integrazione dei dati, abbracciando tutte le tipologie di dati, sia strutturati che non strutturati. Inoltre, agisce come server di applicazioni, supportando le aziende nella creazione di applicazioni intelligenti basate su informazioni approfondite, sfruttando le potenzialità dell'in-memory computing e della tecnologia di machine learning. Queste caratteristiche sono accessibili sia in modalità cloud che on-premise.

Un elemento distintivo di SAP HANA è la sua capacità di combinare diverse funzionalità di gestione di dati, rendendo immediatamente accessibili tutti i tipi di dati da un'unica piattaforma. Tale integrazione semplifica notevolmente l'IT, agevolando le iniziative di innovazione aziendale e superando le sfide legate alla trasformazione digitale. Con decine di migliaia di clienti in tutto il mondo, SAP HANA si afferma come una soluzione moderna e consolidata, giocando un ruolo chiave nell'accelerare il progresso tecnologico e la competitività aziendale.

Progettata per eseguire query rapidamente e facilitare transazioni ad alta velocità, l'architettura in-memory di SAP HANA è orientata alle colonne e abbraccia non solo la gestione dei database, ma anche lo sviluppo di applicazioni, l'analisi avanzata dei dati e la virtualizzazione flessibile degli stessi.

Un ulteriore approfondimento degno di nota riguarda le diverse tipologie di tabelle in SAP HANA; infatti, non si ha una sola tipologia generica in grado di contenere dati, ma, a livello più alto, si ha una divisione in *strutture* e *tabelle*. Una tabella costituisce un insieme di dati ordinati in cui ciascuna riga rappresenta un record, ossia un dato memorizzato in SAP. Ogni riga è formata da campi, corrispondenti alle singole "caselle", ciascuna contenente un dato specifico.

Diversamente, la struttura conserva informazioni solo durante l'esecuzione del programma con i dati, ovvero durante il "runtime" dell'elaborazione. Ci sono principalmente tre tipi di tabelle e tre tipi di strutture in SAP. Questi sono:

- *Tabella trasparente (Transparent table)*: c'è una tabella fisica nel database per ogni tabella trasparente. I nomi delle tabelle fisiche corrispondono alla definizione logica della tabella nel dizionario ABAP. In esse vengono memorizzati tutti i dati aziendali e i dati dell'applicazione.

- *Tabella raggruppata (Pooled Table)*: le tabelle raggruppate possono essere utilizzate per memorizzare dati di controllo (ad esempio, sequenze di schermate, parametri di programma o dati temporanei). Diverse tabelle raggruppate possono essere combinate per formare pool di tabelle. Questi ultimi corrispondono a delle tabelle fisiche presenti nel database in cui sono memorizzati tutti i record delle tabelle raggruppate allocate. Ciò è finalizzato a scopi interni, come la memorizzazione di dati di controllo o testi di aggiornamento, etc.

- *Tabella di cluster*: le tabelle di cluster contengono testo continuo, ad esempio documentazione. Diverse tabelle di cluster possono essere combinate per formare un cluster di tabelle. Diverse linee logiche di diverse tabelle vengono combinate per formare un record fisico in questo tipo di struttura dati. Ciò consente la memorizzazione o l'accesso oggetto per oggetto. Per combinare tabelle in cluster, almeno parte delle chiavi deve essere in comune. Diverse tabelle di cluster sono memorizzate in una tabella corrispondente nel database.

- *Struttura*: una struttura (tipo strutturato) è composta da componenti (campi) i cui tipi sono definiti. Un componente può avere un tipo elementare, un tipo strutturato, un tipo di tabella o un tipo di riferimento. Le strutture sono utilizzate in modo particolare per definire i dati di interfaccia e per definire i tipi dei parametri dei moduli di funzione. Le strutture utilizzate più frequentemente possono essere modificate tutte insieme purchè siano state definite centralmente. Il dizionario ABAP attivo apporta questa modifica in tutti i luoghi rilevanti. I programmi ABAP o i modelli di schermate che utilizzano una struttura vengono aggiornati automaticamente quando la struttura cambia.

- *Struttura di appendice*: una struttura di appendice definisce un insieme di campi che appartengono a un'altra tabella o struttura ma sono trattati nella gestione delle correzioni come un oggetto separato. Le strutture di appendice sono utilizzate per supportare le modifiche.

- *Struttura di vista generata (Generated View Structure)*: in fase di attivazione, viene generata una struttura per una vista. Questa struttura funge da interfaccia per l'ambiente di runtime. In genere, essa non appare nel dizionario ABAP.

Chiarite le caratteristiche generali riguardanti il database interno a SAP, il focus dello studio si è spostato verso l'analisi della Graphic User Interface del modulo Plant Maintenance in possesso dell'azienda. Di seguito verranno riportate alcune illustrazioni (Figure 3.1, 3.2, 3.3, 3.4 e 3.5) di diverse schermate della UI in questione.

Il processo di approfondimento inizialmente consisteva nell'esplorazione dei progetti aziendali caricati nel software, cercando di trovare pattern, associazioni e concetti ricorrenti che potessero essere in qualche modo riconducibili ad una struttura relazionale.

Un primo strumento che ha velocizzato enormemente lo studio è stata la pagina web *www.se80.co.uk* (la cui home page è visibile in Figura 3.6); in essa è contenuta una vastissima quantità di informazioni riguardnti ogni singola tabella presente nel database HANA (in Figura 3.7 è possibile osservare una di queste pagine informative). Grazie ad essa è stato possibile iniziare ad osservare direttamente tutti i campi caratterizzati delle tabelle SAP e a fare chiarezza sull'intricata rete di riferimenti fra queste ultime.

Per ogni tabella vengono riportati i tutti i campi, identificativi e non, accompagnati da una serie di informazioni relative alla natura del dato; fra queste ultime, quelle di maggior rilevanza allo scopo dello studio sono:

- *Field*: nome dell'attributo.

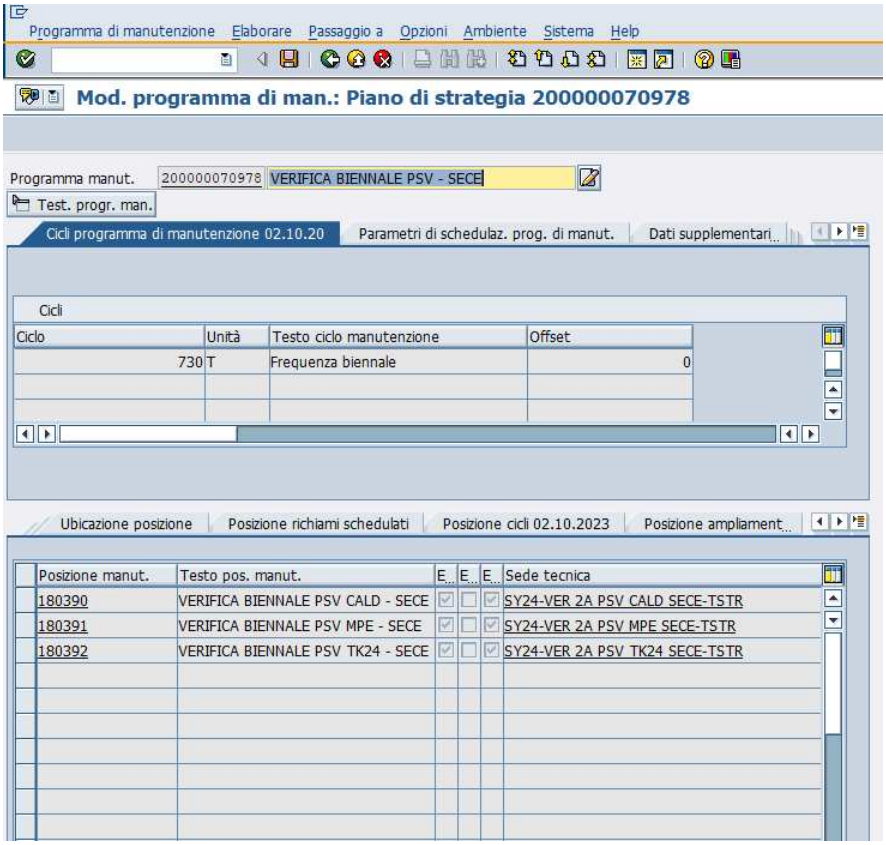


Figura 3.1: Schermata SAP relativa al Piano di manutenzione

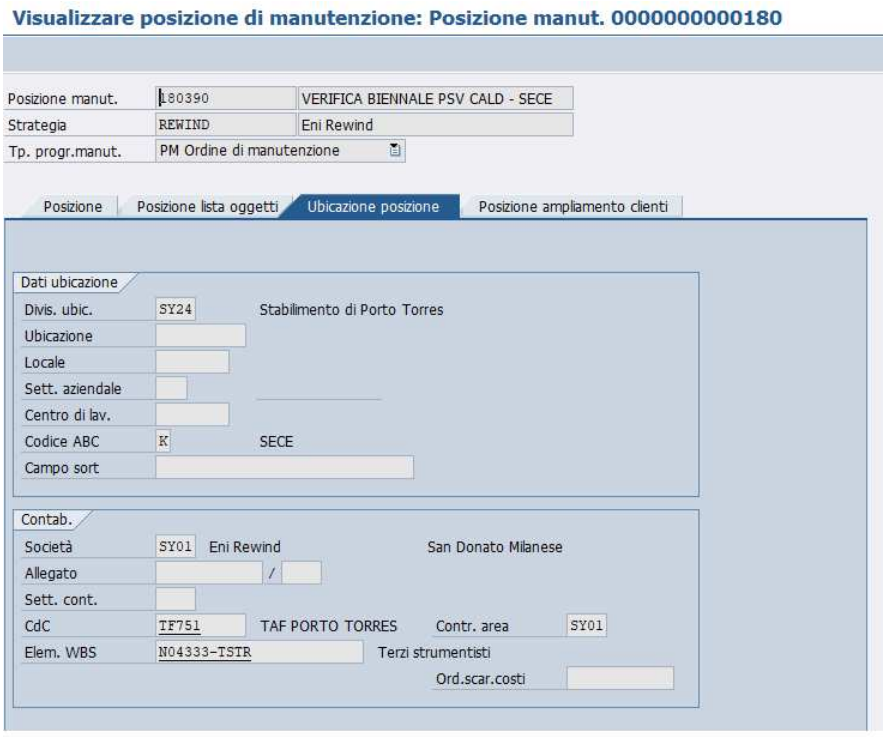


Figura 3.2: Schermata SAP relativa alla Posizione di manutenzione

- *Description*: descrizione sommaria del significato pratico del campo.

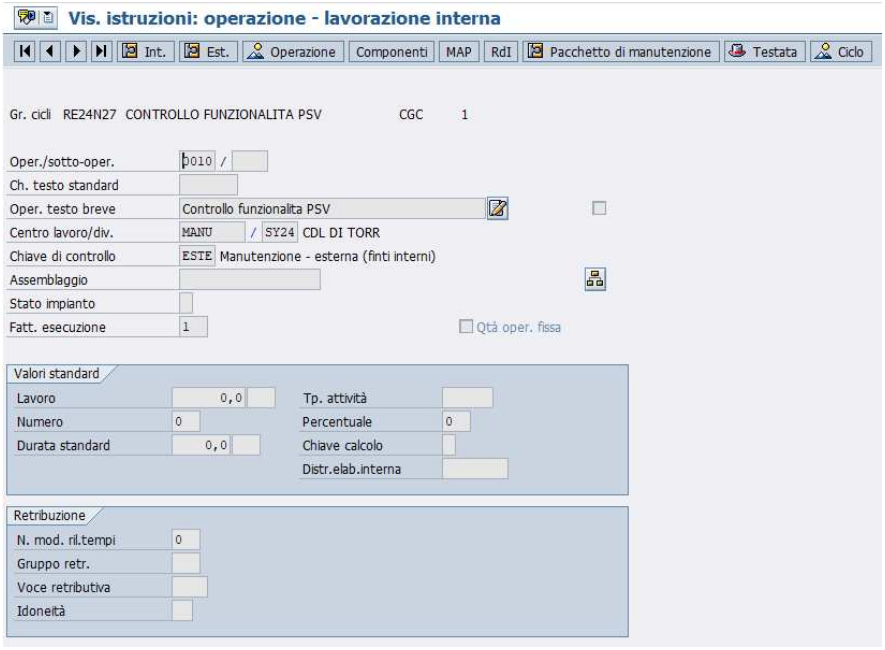


Figura 3.3: Schermata SAP relativa alle Operationi

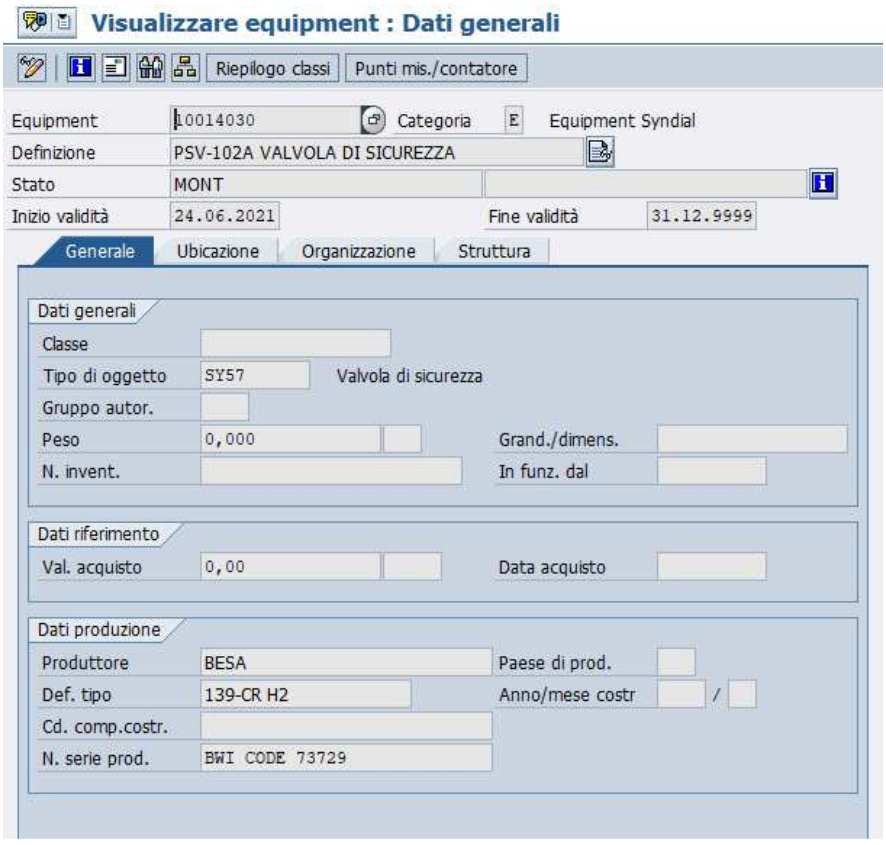


Figura 3.4: Schermata SAP relativa all'Equipment

- *Data Element*: tipo di dato legato al campo; oltre ai tipi classici (CHAR, NUMC etc.) SAP supporta anche la definizione di user-defined-data-types.
- *Domain name*: questa informazione ci permette di identificare quali campi corrispondono

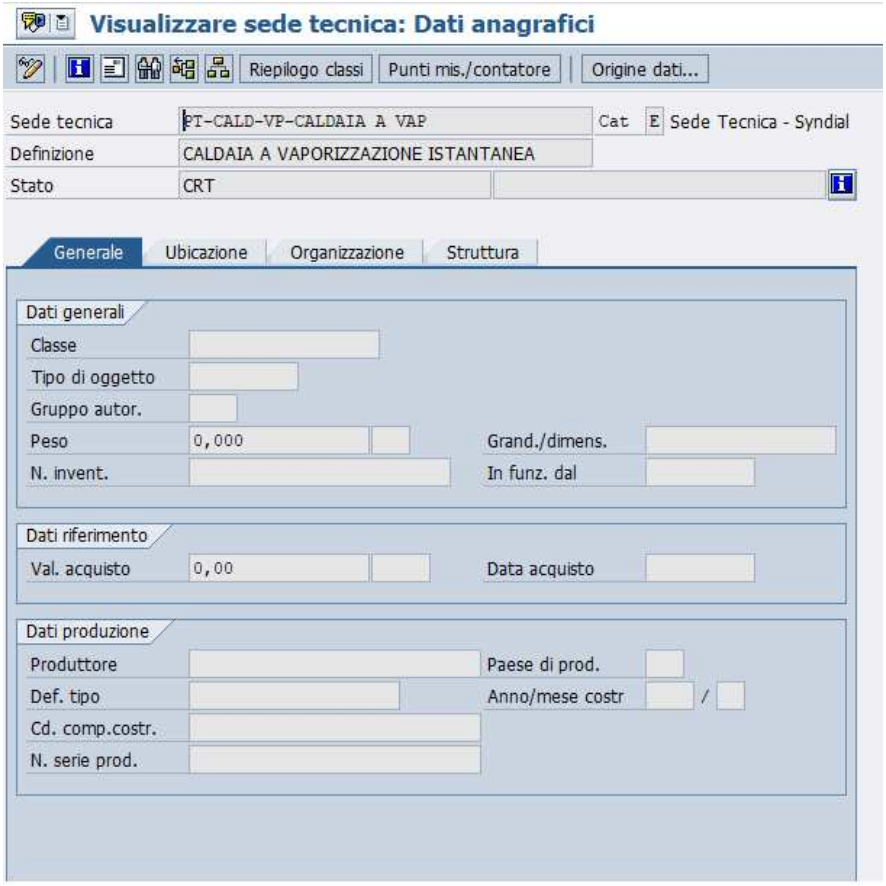


Figura 3.5: Schermata SAP relativa alla Sede tecnica

a record presenti in tabelle esterne. Di fatto, specificando una tabella esterna, questa informazione ci permette di capire quali sono i valori accettati dall'attributo (appunto il suo dominio)

Una prima osservazione da fare in merito alle tabelle su cui SAP PM basa il suo funzionamento è la presenza orizzontale (in tutte le strutture dati) di un attributo, riscontrato sempre fra i campi identificativi (e, quindi, parte della chiave primaria), chiamato *MANDT*. Questa informazione rappresenta il cliente per il quale si sta sviluppando un progetto. Essenzialmente si tratta di una divisione verticale in grado di partizionare l'insieme dei dati progettuali aziendali sulla base del cliente per il quale si sta lavorando.

Tornando ora a parlare del procedimento seguito per la mappatura, un ulteriore tool che ha enormemente facilitato il rintracciamento delle varie tabelle, accorciando di molto le tempistiche, è stata la finestra delle informazioni tecniche legate alla SAP GUI (Figura 3.8). Essenzialmente, per ogni campo presente nell'interfaccia, si è stati in grado di osservare il nome e la tabella di appartenenza, insieme ad altre informazioni di contorno.

Questo significa che si è smesso di navigare alla cieca cercando di trovare i legami fra cosa veniva osservato nell'interfaccia SAP e le corrispondenti informazioni nel database, passando ad una modalità operativa più veloce ed efficace.

Andando avanti in questo modo, procedendo tabella dopo tabella, si è delineata la struttura completa portante di SAP PM, la quale ha rappresentato un importante checkpoint nello stato di avanzamento del progetto.

Nelle prossime pagine seguiranno delle illustrazioni della struttura completa (Figure 3.9 e 3.10), frutto della fase di mappatura (per ragioni legate alla leggibilità, il grafico completo

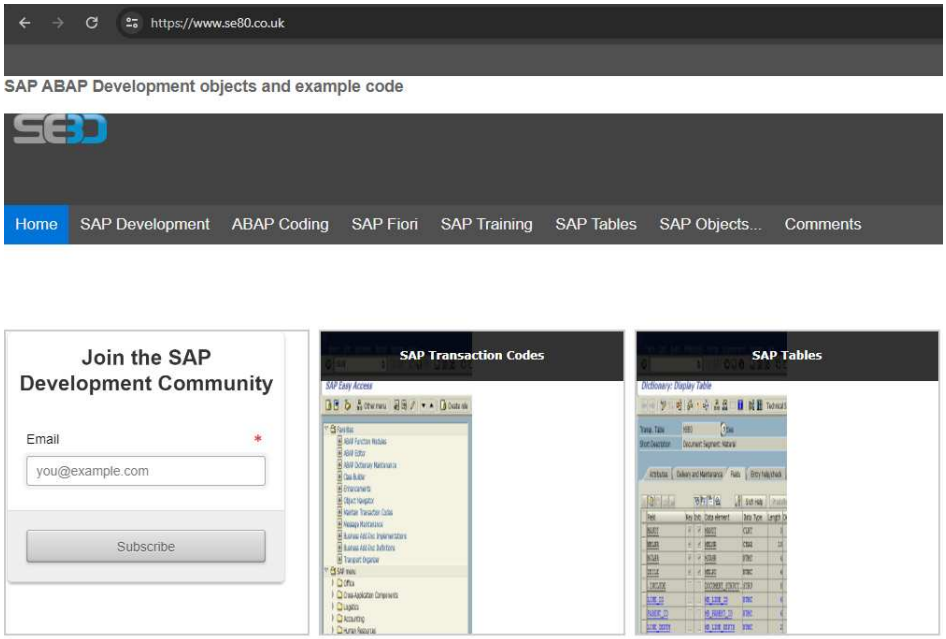


Figura 3.6: Home Page di se80

Company Codes data

T001 is a standard SAP Table which is used to store Company Codes data and is available within R/3 SAP systems depending on the version and release level.

The T001 table consists of various fields, each holding specific information or linking keys about Company Codes data available in SAP. These include BUKRS (Company Code), BUTXT (Name of Company Code or Company), ORT01 (City), LAND1 (Country Key).. See below for full list along with technical details, documentation, text table, check tables, foreign key relationships, conversion routines, relevant tcodes and example ABAP select code etc. .

Delivery Class: C - Customising table, maintenance only by cust
Display/Maintenance via tcode SM30: Display/Maintenance Allowed but with Restrictions
SAP enhancement categories: Can be enhanced (character-type or numeric)

T001 KEY fields - List of fields that make up the full SQL / SAP table key

Field	Description	Data Element	Data Type	length (Dec)	Check table	Conversion Routine	Domain Name	MemoryID	SHLP
MANDT	Client	MANDT	CLNT	3	T000		MANDT		
BUKRS	Company Code	BUKRS	CHAR	4			BUKRS	BUK	C_T001

Field	Description	Data Element	Data Type	length (Dec)	Check table	Conversion Routine	Domain Name	MemoryID	SHLP
BUTXT	Name of Company Code or Company	BUTXT	CHAR	25			TEXT25		
ORT01	City	ORT01	CHAR	25			TEXT25		
LAND1	Country Key	LAND1	CHAR	3	T005		LAND1	LND	
WAERS	Currency Key	WAERS	CUKY	5	TCURC		WAERS	FWS	

Figura 3.7: Esempio di sezione descrittiva relativa alla tabella in se80

sarà riportato su più pagine, collegando i punti di taglio con delle lettere).

Struttura completa di SAP PM

Si procederà ora, nel corso delle prossime pagine, a fornire una descrizione generale dei concetti fondamentali che possono essere riscontrati all'interno dello schema appena fornito;

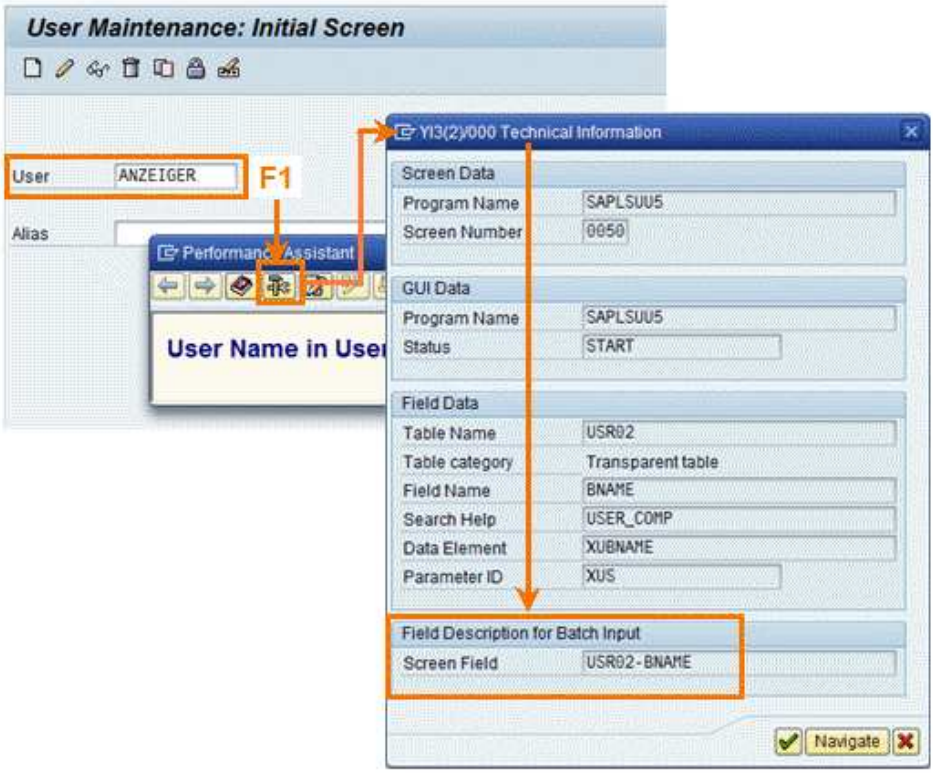


Figura 3.8: Tool di ricerca delle informazioni grafiche

per le tabelle di interesse ai nostri scopi verranno fornite una descrizione semantica e una descrizione di carattere più tecnico legata ai campi caratterizzanti.

Si vedrà, anche, come ciscuno di questi concetti si posiziona all'interno di asset e maintenance register e come sono fra loro interconnessi.

3.4 Concetti di base

I concetti che verranno approfonditi nel continuo di questa tesi non sono solo da intendere come tabelle presenti nel database SAP, ma come elementi portanti di quella che sarà, poi, la struttura finale che verrà realizzata. Il significato, le modalità di rappresentazione ed i legami fra questi concetti forniranno quello che sarà lo standard di cui tanto si è discusso nelle sezioni precedenti.

Queste informazioni relative agli attributi di ogni entità verranno riprese dalla pagina se80.co.uk (utilizzata in fase di definizione della mappatura).

È anche importante specificare che la maggior parte dei concetti che verranno illustrati corrispondono nello schema ad un numero variabile di tabelle; alcuni concetti sono riconducibili ad una singola tabella, molti altri no.

Detto ciò ora si proseguirà illustrando la forma data da SAP a questi elementi portanti; le diverse tabelle illustrate nella mappatura verranno raggruppate in base alla loro appartenenza al maintenance o all'asset register (concetti illustrati nella sezione relativa allo stato dell'arte aziendale), per poi procedere all'approfondimento dettagliato di ciscuna di esse. In ogni grafico vengono riportati in grigio (nella parte alta della tabella) i campi chiave della struttura dati.

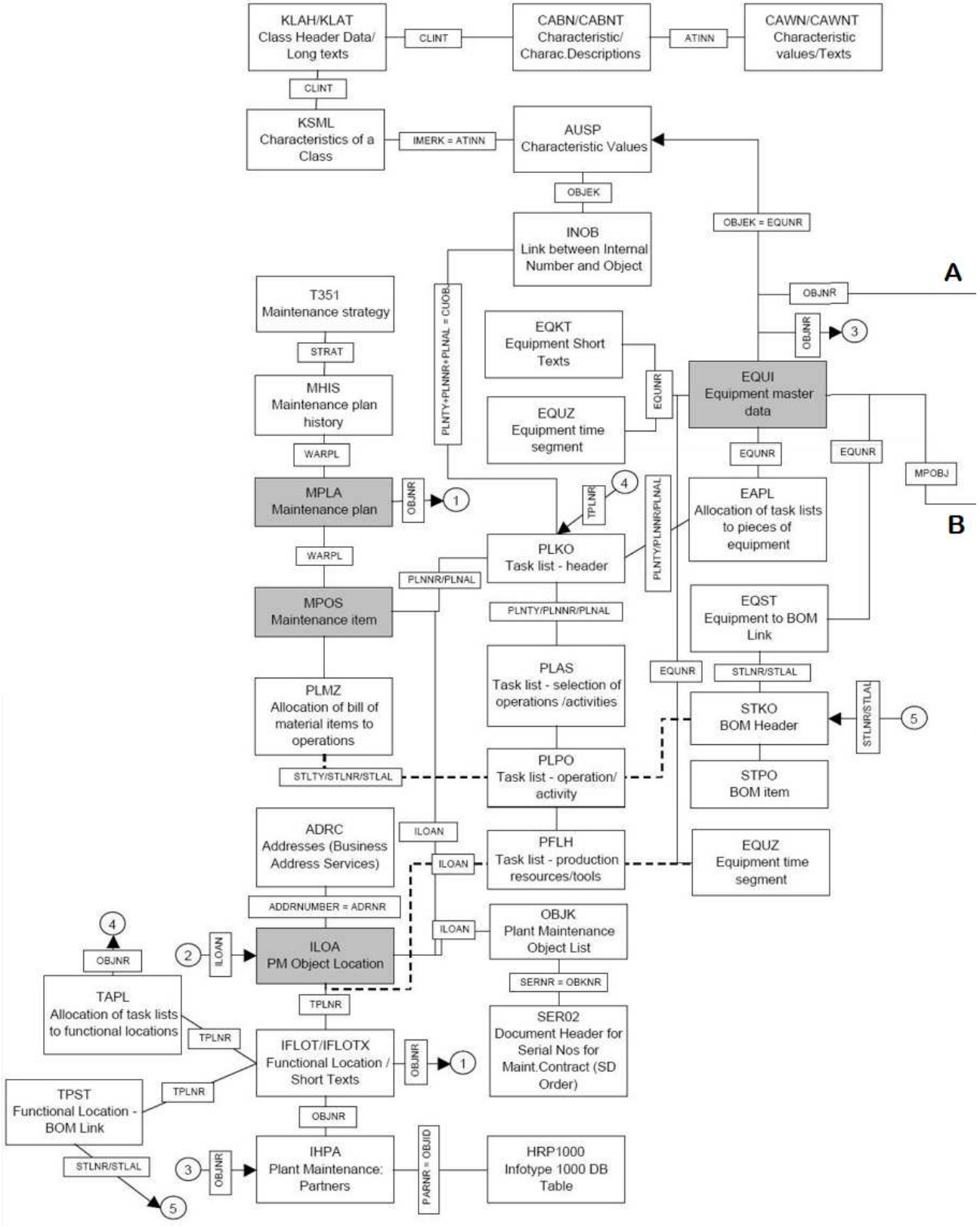


Figura 3.9: Mapping di SAP PM (prima parte)

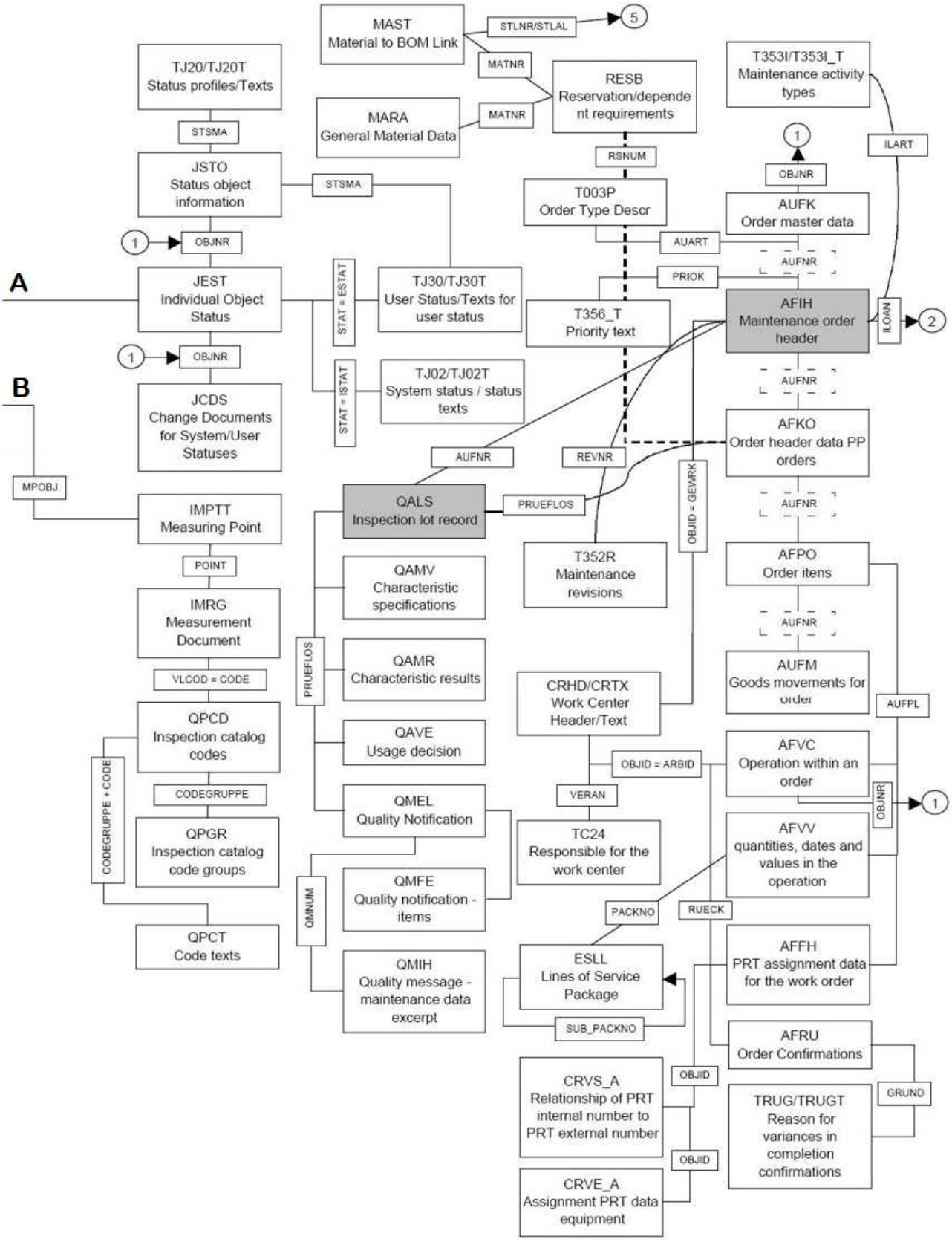


Figura 3.10: Mapping di SAP PM (seconda parte)

3.5 Elementi Asset Register

Come accennato nelle sezioni precedenti, nell'asset register troviamo tutti quei concetti legati in qualche modo alle risorse fisiche in possesso ad un'organizzazione, con annesse delle informazioni di carattere tecnico come descrizioni, tipologia di apparecchiatura, valore economico, stato dell'asset, e molto altro ancora.

3.5.1 Equipment

Si tratta forse del concetto più alla base di tutti, in quanto identifica un'apparecchiatura vera e propria, la quale può essere sottoposta a manutenzione. Esempi di equipment possono essere: pompe, motori, valvole, ma anche linee di trasmissione o cablaggi.

Nel contesto aziendale di Pansoinco l'equipment rappresenta il blocco di partenza dal quale poi vengono sviluppati tutti i lavori; infatti, è l'elemento atomico di più basso livello, che consente la definizione dettagliata dell'asset register stesso. Le tabelle nella mappatura contenenti informazioni in merito al concetto di equipment sono EQUI e EQUZ.

EQUI

Questa tabella viene denominata *Equipment Master Data* e contiene tutte quelle informazioni di carattere generale riguardanti l'attrezzatura in questione. Di seguito viene riportata una tabella (Figura 3.11) contenente gli attributi di maggior interesse (verranno trascurati tutti quei campi relativi alle indicizzazioni interne di SAP).

Per quanto riguarda i campi identificativi di questa prima tabella troviamo il campo *Client (MANDT,* di cui già si è discusso nella sezione introduttiva di SAP PM) ed il campo *Equipment Number (EQUNR)*. Come accennato in precedenza, l'unione del campo identificativo con Client permette di ottenere in SAP PM una separazione verticale; saranno, quindi, consentite duplicazioni dello stesso Equipment Number se queste sono relative a "Clienti" differenti.

Spostando ora il focus sui campi di carattere descrittivo, è possibile osservare all'interno di questa struttura dati tutta una serie di attributi legati ad informazioni generali riguardanti l'equipment. Sono presenti caratteristiche relative alla struttura fisica dell'attrezzatura, come il suo peso (*Weight of Object*) e le sue dimensioni (*Size/Dimension)*; sono, anche, riscontrabili informazioni di identificazione, come numeri di serie/modello (*Manufacturer serial number, Manufacturer model number*) e dati riguardanti il produttore/fornitore (*Manufacturer of asset, Country of manufacturer, Vendor number, Year/Month of construction*).

Tuttavia gli attributi di maggior rilevanza agli scopi del progetto sono:

- *Equipment category (EQTYP)*: con questo attributo viene introdotta una categorizzazione dell'equipment di carattere più generale, spesso ricondotta al campo di utilizzo. Può essere utilizzato per aggregare attrezzature che condividono caratteristiche simili, indipendentemente dal tipo tecnico specifico.

  Esempi di raggruppamenti ottenuti grazie a questo attributo possono essere "Pompe", "Motori" o "Veicoli", dove ogni categoria raggruppa attrezzature simili indipendentemente dal fatto che siano, ad esempio, pompe centrifughe o pompe volumetriche.

- *Type of technical object (EQART)*: si tratta di uno dei campi più importanti di tutta la struttura dati gestita da SAP PM. Questo campo mira a dare una classificazione più specifica di quella fornita dall'attributo Equipment Category precedentemente descritto, basata sul tipo tecnico dell'attrezzatura, indicando la natura specifica dell'oggetto.

  Esso specifica il tipo tecnico dell'attrezzatura ed è collegato a informazioni tecniche più dettagliate; esempi di valori attribuibili a questo campo possono essere "Pompa Centrifuga", "Motore Elettrico" o "Auto".

Field	Description	Data Element	Data Type	length (Dec)	Check table
EQTYP	Equipment category	EQTYP	CHAR	1	T370T
EQART	Type of Technical Object	EQART	CHAR	10	T370K
INVNR	Inventory number	INVNR	CHAR	25	
GROES	Size/dimension	GROSS	CHAR	18	
BRGEW	Weight of object	OBJ_WEIGHT	QUAN	13(3)	
GEWEI	Unit of weight	WEIGHT_UNIT	UNIT	3	T006
ANSDT	Acquisition date	ANDTI	DATS	8	
ANSWT	Acquisition Value	ANSWT	CURR	13(2)	
WAERS	Currency Key	WAERS	CUKY	5	TCURC
ELIEF	Vendor number	ELIEF	CHAR	10	LFA1
GWLEN	Date on which the warranty ends	GWLEN	DATS	8	
GWLDT	Warranty Date	GWLDT	DATS	8	
WDBWT	Equipment replacement value	WDBWT	CURR	13(2)	
HERST	Manufacturer of asset	HERST	CHAR	30	
HERLD	Country of manufacture	HERLD	CHAR	3	T005
HZEIN	Manufacturer drawing number	HZEIN	CHAR	30	
SERGE	Manufacturer serial number	SERGE	CHAR	30	
TYPBZ	Manufacturer model number	TYPBZ	CHAR	20	
BAUJJ	Year of construction	BAUJJ	CHAR	4	
BAUMM	Month of construction	BAUMM	CHAR	2	
APLKZ	Indicator: Task List Exists	APLKZ	CHAR	1	
AULDT	First delivery date of the equipment	AULDT	DATS	8	
INBDT	Start-up Date of the Technical Object	ILOM_DATAB	DATS	8	
GERNR	Serial Number	GERNR	CHAR	18	
GWLDV	Warranty date for Sales and Distribution	GWLDV	DATS	8	
OBJNR	Object number	J_OBJNR	CHAR	22	Assigned to domain

Figura 3.11: Tabella Equipment Master Data

- *Object number (OBJNR)*: con l'object number viene introdotto un ulteriore concetto chiave, ovvero l'*oggetto*. In SAP ogni attrezzatura (equipment, ma anche functional location e material, che verranno approfonditi in seguito) è vista come un "oggetto tecnico". Con questo termine si vuole identificare un livello di astrazione superiore con il quale vengono accomunati e raggruppati elementi differenti.

L'unione di questi campi permette di ottenere una categorizzazione su due livelli (uno più generico ed uno più dettagliato) di tutti gli equipment registrati nel sistema.

Ciò avrà un importanza fondamentale quando si discuterà della possibilità di applicare in futuro dei modelli presi dal mondo del Machine Learning al fine di attribuire

automaticamente operazioni di manutenzione a determinate tipologie di apparecchiature.

EQUZ

Questa seconda tabella relativa al concetto di Equipment viene definita da SAP come "Equipment time segment data"; le informazioni al suo interno si concentrano sullo stato dell'attrezzatura. È facile capire, quindi, che si sta facendo riferimento a delle informazioni variabili nel tempo (da qui il nome "time segment"). La divisione del concetto di equipment in due tabelle è proprio dovuta a questo fatto; la tabella EQUI fornisce informazioni di base sull'attrezzatura, mentre la tabella EQUZ si concentra sugli stati dell'attrezzatura e sui cambiamenti di questi ultimi nel tempo. Segue un'illustrazione dei campi principali della tabella (Figura 3.12).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
EQUNR	Equipment Number	EQUNR	CHAR	18	EQUI
DATBI	Valid To Date	DATBI	DATS	8	
TIMBI	Equipment usage period time stamp	TIMBI	TIMS	6	
DATAB	Valid-From Date	DATAB	DATS	8	
IWERK	Maintenance Planning Plant	IWERK	CHAR	4	T399I
IWERKI	Data origin indicator	INHER	CHAR	1	
HEQUI	Superordinate Equipment	HEQUI	CHAR	18	EQUI
HEQNR	Equipment position at InstallLoc (Superior Equip./FunctLoc)	HEQNR	CHAR	4	
INGRP	Planner Group for Customer Service and Plant Maintenance	INGRP	CHAR	3	T024I
INGRPI	Data origin indicator	INHER	CHAR	1	
PM_OBJTY	Object Type of CIM Resources for Work Center	PM_OBJTY	CHAR	2	
GEWRK	Object ID of the Work Center	LGWID	NUMC	8	CRID
ILOAN	Location and account assignment for technical object	ILOAN	CHAR	12	ILOA
KUND1	Customer number	KUND1	CHAR	10	KNA1
KUND2	End customer number	KUND2	CHAR	10	KNA1
KUND3	Operator	KUND3	CHAR	10	KNA1
LIZNR	Equipment license number	LIZNR	CHAR	20	
RBNR	Catalog Profile	RBNR	CHAR	9	T352B

Figura 3.12: Tabella Equipment Time Segment

Fra i campi chiave di questa tabella, oltre ad avere Client ed Equipment number, troviamo un terzo elemento chiamato *Valid To Date*. Quest'ultimo rappresenta una data finale di validità (il tipo DATBI è lo standard di SAP per la rappresentazione delle date) dello stato esposto dalla tabella, ed è proprio grazie ad esso che è possibile osservare l'evoluzione dell'equipment

nel tempo. Il collegamento ai dati generali dell'equipment presenti in EQUI è garantito dagli altri due campi facenti parte della chiave.

Parlando, ora, dei campi descrittivi osservabili nella tabella troviamo informazioni relative a possibili licenze associabili all'asset (*Equipment license number*), informazioni riguardanti la gestione/responsabilità (*Planner Group for customer service and Plant Maintenance, Object ID of the Work Center*), e molto altro ancora.

Di maggiore interesse sono i seguenti campi:

- *Object ID of the Work Center (GEWRK)*: con questo campo si introduce un concetto rilevante nell'ambito dello standard SAP e anche degli standard interni a Pansoinco, ovvero il *Work Center*. In italiano viene definito "Centro di Lavoro" e si definisce come un qualcosa che viene utilizzato per pianificare e monitorare le attività di produzione/manutenzione all'interno di un'organizzazione.

  Un Work Center rappresenta un'entità fisica o logica in cui vengono eseguite determinate operazioni o processi. Alcuni esempi possono essere macchine, stazioni di lavoro, reparti o aree specifiche in cui vengono svolte attività di produzione o manutenzione. Come avviene per equipment, anche i Work Center vengono raccolti nella categoria generale "Object" e, di conseguenza, possiedono un Object ID identificativo; è proprio a questo campo che si fa riferimento dalla tabella EQUZ. L'Object ID of the Work Center identifica il centro di lavoro che si deve occupare dell'equipment, in maniera non meglio definita. Il campo, quindi, si trova nel time segment in quanto questa responsabilità può cambiare nel tempo.

- *Location and account assignment for techinical object (ILOAN)*: su questo campo si tornerà a discutere quando si introdurranno le Functional Location; per ora è sufficiente sapere che in EQUZ questo dato fornisce informazioni sulla posizione/locazione dell'equipment; con ciò si intende il luogo fisico in cui è posizionata l'attrezzatura. Come è facile intuire, anche questa informazione può cambiare nel tempo (in alcuni casi frequentemente).

- *Superordinate Equipment (HEQUI)*: questo campo si riferisce a un'attrezzatura superordinata o principale a cui l'equipment in questione (di livello inferiore) è collegato. Questo attributo è utilizzato per stabilire relazioni gerarchiche tra diverse attrezzature, creando una struttura ad albero o gerarchia. Non è raro, infatti, avere a che fare con delle attrezzature che presentano delle sotto-componenti.

  È di particolare interesse notare come la tabella a cui fa riferimento questo attributo è proprio EQUI; ciò vuol dire che si ha a che fare con una relazione ricorsiva.

Per concludere la discussione legata al concetto di equipment ci sarebbe da parlare anche della tabella *EQKT* - Equipment Short Texts, utilizzata per memorizzare informazioni testuali o descrittive associate alle attrezzature. Per ragioni legate alla poca importanza di questa tabella non si scenderà nel dettaglio come fatto per EQUI ed EQUZ; tuttavia si vuole cogliere l'occasione per parlare di come SAP gestisce le descrizioni testuali.

Non è, infatti, raro riscontrare, all'interno della struttura dati, delle tabelle rilegate al solo contenimento di informazioni testuali/descrittive; questo viene fatto in quanto SAP vuole garantire la possibilità di specificare diverse descrizioni in varie lingue per uno stesso oggetto, e, di conseguenza, utilizza queste tabelle di supporto non potendo risolvere il problema utilizzando un campo unico. Guardando rapidamente i campi di EQKT (Figura 3.13) ci si accorge che a fare parte della chiave abbiamo il campo EQUNR, ad indicare l'equipment di riferimento, ed il campo SPRAS, utilizzato per specificare la lingua in cui viene espressa la descrizione.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
EQUNR	Equipment Number	EQUNR	CHAR	18	EQUI
SPRAS	Language Key	SPRAS	LANG	1	T002
EQKTX	Description of technical object	KTX01	CHAR	40	

Figura 3.13: Tabella Equipment Short Texts

3.5.2 Functional Location

Con functional location, "sede tecnica" in italiano, si intende un concetto superiore all'equipment, che spesso viene definito come una posizione di installazione. In generale, quindi, per ogni functional location si potranno avere un certo numero di equipment collegati. Il legame fra equipment e functional location, come vedremo più avanti, rappresenta uno dei tanti punti di criticità verso il raggiungimento dell'omogeneità. Questo perché, in base allo standard definito dal cliente per il quale si sta definendo un piano di manutenzione, sarà possibile avere solo uno, molti oppure nessun equipment collegato a ciascuna sede tecnica. Nella mappatura SAP le tabelle che possono essere ricondotte al concetto di Functional Location sono IFLOT ed ILOA.

IFLOT

Questa prima tabella è utilizzata per archiviare informazioni generali relative alla functional location. Non si ha, in questo caso, come per l'equipment, una divisione in dati "fissi" e dati time-dependent in quanto il concetto stesso di functional location è un qualcosa che tendenzialmente cambia poco nel tempo; di conseguenza, avrebbe poco senso riservare un'intera struttura dati come viene fatto per EQUZ. Di seguito viene riportata un'illustrazione dei campi di maggior interesse presenti in IFLOT (Figura 3.15).

A formare la chiave di questa tabella, oltre al campo Client, troviamo il campo *TPLNR*, il quale consiste in un codice alfanumerico che identifica la functional location in maniera univoca.

Per quanto riguarda i campi descrittivi presenti in IFLOT troviamo alcuni attributi praticamente sovrapponibili a quelli riscontrati in EQUI, quindi relativi all'attribuzione di responsabilità legate alla functional location e, nel caso di sedi funzionali corrispondenti a sistemi fisici, anche attributi descrittivi di proprietà, quali peso dimensioni etc. Tuttavia le proprietà della functional location veramente interessanti agli scopi del progetto sono le seguenti:

- *Functional location category (FLTYP)*: questo campo fornisce informazioni sulla categoria o la classificazione della localizzazione funzionale all'interno del sistema SAP. Il valore contenuto nel campo FLTYP identifica la natura della sede funzionale e può essere utilizzato per scopi di classificazione e gestione. Ad esempio, alcuni valori che potrebbero essere presenti nel campo FLTYP includono:
  - *P (Position)*: indica che la localizzazione funzionale è associata a una posizione specifica all'interno di un impianto o di un'area.
  - *L (Location)*: indica una localizzazione generica, che potrebbe essere utilizzata per rappresentare un'unità operativa, un'area o un'apparecchiatura.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
TPLNR	Functional Location	TPLNR	CHAR	30	
TPLKZ	Functional location structure indicator	TPLKZ	CHAR	5	T370S
FLTYP	Functional location category	FLTYP	CHAR	1	T370F
TPLMA	Superior functional location	TPLMA	CHAR	30	IFLOT
DATAB	Start-up Date of the Technical Object	ILOM_DATAB	DATS	8	
BEGRU	Technical object authorization group	IAUTG	CHAR	4	T370B
IEQUI	Installation of equipment allowed at the functional location	IEQUI	CHAR	1	
EINZL	Single equipment installation at functional location	EINZL	CHAR	1	
IWERK	Maintenance Planning Plant	IWERK	CHAR	4	T399I
INGRP	Planner Group for Customer Service and Plant Maintenance	INGRP	CHAR	3	T024I
PM_OBJTY	Object Type of CIM Resources for Work Center	PM_OBJTY	CHAR	2	
LGWID	Object ID of the Work Center	LGWID	NUMC	8	CRID
RBNR	Catalog Profile	RBNR	CHAR	9	T352B

Figura 3.14: Tabella Functional Location Data (prima parte)

  - *B (Building)*: indica che la localizzazione funzionale è un edificio all'interno dell'impianto.
  - *R (Room)*: indica che la localizzazione funzionale è una stanza o un locale all'interno di un edificio.

- *Superior functional location (TPLMA)*: questo attributo, facente riferimento alla tabella IFLOT stessa, permette di definire una *gerarchia delle functional location*; ognuna di queste infatti può essere collegata in una struttura ad albero composta da altre sedi funzionali.

  Dato che, come detto precedentemente, in SAP PM le functional location rappresentano le diverse unità operative, attrezzature o aree interne, questa gerarchia può essere utilizzata per riflettere la struttura di un impianto, con localizzazioni funzionali principali che contengono sotto-localizzazioni funzionali o componenti.

  Ad esempio, si potrebbe avere una sede principale per un'area di produzione, con sotto-localizzazioni funzionali per macchinari specifici all'interno di quell'area.

- *Functional location structure indicator (TPLKZ)*: questo campo è legato alla modalità di rappresentazione della gerarchia della functional location.

  Infatti, nella pratica spesso fornisce delle indicazioni riguardanti la forma che un id della sede funzionale deve avere per far capire a chi lo legge la posizione di tale sede nella gerarchia.

ILOAN	Location and account assignment for technical object	ILOAN	CHAR	12	ILOA
OBJNR	Object number	J_OBJNR	CHAR	22	ONR00
POSNR	Position in superior technical object	ILOM_POSNR	CHAR	4	
EQART	Type of Technical Object	EQART	CHAR	10	T370K
INVNR	Inventory number	INVNR	CHAR	25	
GROES	Size/dimension	GROSS	CHAR	18	
BRGEW	Gross Weight	BRGEW	QUAN	13(3)	
GEWEI	Weight Unit	GEWEI	UNIT	3	T006
ANSWT	Acquisition Value	ANSWT	CURR	13(2)	
WAERS	Currency Key	WAERS	CUKY	5	TCURC
ANSDT	Acquisition date	ANDTI	DATS	8	
HERST	Manufacturer of asset	HERST	CHAR	30	
HERLD	Country of manufacture	HERLD	CHAR	3	T005
BAUJJ	Year of construction	BAUJJ	CHAR	4	
BAUMM	Month of construction	BAUMM	CHAR	2	
TYPBZ	Manufacturer model number	TYPBZ	CHAR	20	
EMATN	Material Number Corresponding to Manufacturer Part Number	EMATN	CHAR	18	Assigned to domain
SERGE	Manufacturer serial number	SERGE	CHAR	30	

Figura 3.15: Tabella Functional Location Data (seconda parte)

Ad esempio nello standard ENI, il valore ZFORM dello structure indicator dice che ogni functional location deve essere identificata una stringa di 30 caratteri in cui si hanno gli id di tutte le sedi di livello superiore, separate dal carattere "-", seguite dall'identificatore della sede in questione (ed esempio l'id "*RC01-RO04-ROM1-ST03*" indica che si parte dalla sede tecnica radice RC01, per poi passare in sequenza dai nodi figli R004 e ROM1, per raggiungere la nostra sede tecnica ST03; RC01 potrebbe essere un impianto, RO04 un dipartimento, ROM1 una stanza, e finalmente, ST03 la posizione).

- *Installation of equipment allowed (IEQUI)*: questa volta si tratta di un attributo booleano, il quale permette di capire se la functional location che stiamo considerando permette di avere degli equipment ad essa collegati o meno. In generale ad una floc è associato un numero variabile di attrezzature, che potrebbe, in caso di valore False, essere anche zero.

- *Single equipment installation (EINZL)*: questo secondo attributo booleano, in unione con quello precedentemente descritto, consente di delineare il rapporto fra la sede tecnica e gli equipment ad essa collegati; infatti, un eventuale valore True specificato in questo campo starebbe ad indicare una functional location in grado di avere collegato al massimo un equipment; per lo standard ENI, ad ogni sede si fa corrispondere una ed una sola attrezzatura, e quindi in SAP questo valore sarà sempre True.

- *Location and account assignment (ILOAN)*: questo campo può essere descritto per ora come un concetto superiore alla functional location, ovvero la *Location* vera e propria. Senza spendere troppe parole, in questa sezione saranno approfonditi tali concetti e gli usi che se ne fanno in SAP nella prossima sezione.

ILOA

Introdotta nella sezione precedente, questa tabella è in strettissimo contatto con IFLOT e, insieme a quest'ultima, vanno a delineare il concetto generale di functional location. Infatti, ILOA è una struttura dati atta al contenimento di informazioni riguardanti la locazione fisica vera e propria. I lettori più attenti avranno notato che, sebbene si sia parlato più volte del legame fra equipment e functional location, il collegamento fisico fra le tabelle che descrivono questi due concetti non vede come estremi EQUI e IFLOT, bensì EQUI ed ILOA. Per questa ragione, se qualcuno volesse risalire alla sede tecnica relativa ad una data attrezzatura, l'unico modo per farlo sarebbe passare prima da EQUI ad ILOA, e solo successivamente risalire a IFLOT.

Rilevante è, anche, il fatto che la relazione fra IFLOT e ILOA sia del tipo "uno a molti" in ambedue i sensi; avendo il riferimento alla location in IFLOT si potrebbero verificare situazioni in cui più sedi tecniche fanno riferimento alla stessa posizione fisica; d'altra parte, avendo anche il riferimento a sede tecnica in ILOA, si potrebbero avere più location per una stessa sede tecnica.

Quest'ultimo scenario potrebbe, in un primo momento suscitare confusione; tuttavia bisogna sempre tenere in considerazione l'approccio che SAP segue nella gestione delle strutture dati; infatti, la sua filosofia è quella di lasciare a chi sfrutta il sistema quanta più libertà di scelta possibile, permettendo, anche, delle situazioni che a livello pratico potrebbero avere poco senso. Di seguito una rappresentazione dei campi principali presenti in ILOA (Figura 3.16).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
ILOAN	Location and account assignment for technical object	ILOAN	CHAR	12	
TPLNR	Functional Location	TPLNR	CHAR	30	IFLOT
SWERK	Maintenance plant	SWERK	CHAR	4	T001W
STORT	Location of maintenance object	PMLOC	CHAR	10	T499S
MSGRP	Room	RAUMNR	CHAR	8	
BEBER	Plant section	BEBER	CHAR	3	T357
GSBER	Business Area	GSBER	CHAR	4	TGSB
KOKRS	Controlling Area	KOKRS	CHAR	4	TKA01
KOSTL	Cost Center	KOSTL	CHAR	10	CSKS
PROID	Work Breakdown Structure Element (WBS Element)	PS_PSP_PNR	NUMC	8	PRPS
BUKRS	Company Code	BUKRS	CHAR	4	T001

Figura 3.16: Tabella Object Location and Account Assignment

Fra i campi presenti nella tabella, oltre al già illustrato riferimento a IFLOT, è possibile trovare anche dei dati relativi alla posizione fisica descritta dalla tabella, quindi attributi come *Location of maintenance object*, *Room* o *Plant section*.

Tuttavia è anche possibile riscontrare dei campi relativi all'attribuzione di responsabilità legate alla functional location:

- *Business Area (GSBER)*: è un attributo (associato anche ad altri oggetti tecnici) che rappresenta l'unità organizzativa incaricata di raggruppare ed analizzare i dati relativi ai costi e alle attività all'interno legate alla functional location.

- *Controlling Area (KOKRS)*: questo attributo rappresenta uno dei tanti punti di collegamento fra SAP PM ed altri moduli dell'ERP SAP, in questo caso SAP Finance e Controlling (FICO); in particolare, con controlling area si intende un'area di controllo responsabile dei processi finanziari e contabili. Questo attributo può essere utile per monitorare e analizzare i costi associati alla manutenzione di determinate aree degli impianti.

- *Cost Center (KOSTL)*: con centro di costo si intende una struttura organizzativa e contabile che consente di monitorare e assegnare i costi a determinate attività o oggetti all'interno di un'organizzazione.

  L'assegnazione di un centro di costo consente di tracciare le spese associate alla manutenzione, alle attività operative e ad altre operazioni connesse alle sedi funzionali. Questa informazione è preziosa per l'analisi finanziaria, il controllo dei costi e la gestione generale delle risorse aziendali.

- *WBS Element (PROID)*: la WBS è una struttura gerarchica che suddivide un progetto in componenti più gestibili. In SAP PM l'elemento WBS è utilizzato per collegare gli ordini di manutenzione, gli ordini di lavoro o altre attività di manutenzione, a progetti specifici o alle strutture organizzative di un'azienda. Nel contesto di una functional location, quindi, essa sta ad indicare il ramo aziendale incaricato della gestione diretta di quella sede tecnica.

  Inoltre, l'utilizzo dell'elemento WBS consente di aggregare i costi e le risorse associati a diverse attività di manutenzione facenti capo alla stessa struttura organizzativa. Questo aiuta le aziende a monitorare e controllare i costi in modo più efficace, soprattutto quando le attività di manutenzione sono parte di progetti più ampi.

Come succedeva per equipment, anche per la functional location si ha una tabella relagata al contenimento delle sole informazioni di tipo testuale multilingua; questa tabella è chiamata *IFLOTX* e la sua struttura è analoga (Figura 3.17).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
TPLNR	Functional Location	TPLNR	CHAR	30	IFLOT
SPRAS	Language Key	SPRAS	LANG	1	T002
PLTXT	Description of functional location	PLTXT	CHAR	40	

Figura 3.17: Tabella Functional Location Short Texts

D'ora in avanti per ragioni legate alla ripetitività di queste tabelle di supporto descrittive e alla poca influenza di queste agli scopi del progetto, si eviterà di specificare la loro presenza.

3.5.3 Characteristic - Class

Una volta fornita una panoramica completa sui due concetti alla base dell'asset register, ovvero equipment e functional location, si passerà, ora, ad illustrare il concetto di caratteristica; con questo termine in SAP ci si riferisce ad un parametro/attributo misurabile legato ad una tipologia di oggetti. Ricollegandosi a dei concetti visti in precedenza, a determinati object-type o classi possono essere associate soltanto alcune caratteristiche precise. A livello logico determinati attributi possono aver senso solo se legati a determinate tipologie di apparecchiature; ad esempio ci si aspetta che un motore sia caratterizzato da elementi come

corrente assorbita, potenza in uscita, rendimento, e così via. In SAP, quindi, oltre alle categorizzazioni già introdotte dai concetti di object type e category, ne esiste una terza tipologia, più rilevante per gli scopi del progetto, identificata dal concetto di classe.

Con classe ci si riferisce ad un insieme di oggetti o elementi di manutenzione che condividono caratteristiche comuni. A livello semantico si ricalca parzialmente la definizione di categoria vista durante la discussione degli attributi legati alle due tipologie di item; questo è un altro esempio della struttura flessibile di SAP, il quale lascia all'utente finale la libertà di associare un significato semantico a determinati concetti/tabelle presenti nella struttura, secondo quelle che sono le necessità del progetto in questione. Un'altra particolarità legata al concetto di classe in SAP è rappresentata dalla possibilità di definire una gerarchia; tramite degli attribiti, SAP lascia la possibilità di stabilire una struttura ad albero in cui posizionare le varie classi di oggetti; in questo modo, per ogni classe, si potranno ereditare le caratteristiche associate al nodo padre, senza dover definire ogni volta da capo tutta la lista. Entrando ora più nello specifico, introducendole strutture dati relegate al contenimento dei dati relativi alle caratteristiche e alle classi, abbiamo le tabelle *KLAH, KSML, AUSP, CABN* e *CAWN*.

KLAH

SAP PM chiama questa tabella *Class Header Data*, lasciando intendere che si tratta della tabella relegata al contenimento delle informazioni di carattere generale riguardanti una determinata classe. Segue un'illustrazione dei campi principali di questa tabella (Figura 3.18).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
CLINT	Internal Class Number	CLINT	NUMC	10	
KLART	Class Type	KLASSENART	CHAR	3	TCLA
CLASS	Class number	KLASSE_D	CHAR	18	
STATU	Class status	KLSTATUS	CHAR	1	TCLU
KLAGR	Class Group	KLASSENGR	CHAR	10	TCLG

Figura 3.18: Tabella Class Header Data

Come è possibile vedere nella figura, come attributo chiave viene utilizzato il campo *CLINT*, il quale rappresenta un identificatore interno di tipo numerico che SAP utilizza per far riferimento in maniera univoca ad una classe. Per quanto riguarda i campi descrittivi presenti in essa abbiamo:

- *KLART*: si tratta di un attributo utilizzato per far riferimento al "tipo di classe"; è un campo legato alla tabella esterna *TCLA*, la quale viene definita da SAP come "*Class Types in Classification System*". Le informazioni che possono essere riscontrate al suo interno, oltre a fornire una descrizione del tipo di classe, specificano una categoria per la classe in questione e, soprattutto, specificano se alla classe è consentito di ereditare caratteristiche dalle classi superiori.

- *STATU*: come per il campo precedente, questo attributo è collegato ad una tabella esterna chiamata *TCLU* e definita come "*Class: Allowed Classes in Classification System*"; si tratta di una tabella gestionale la quale contiene informazioni sulle classi consentite o permesse per un determinato tipo di classe. Quindi, mentre la tabella TCLA definisce i tipi di classe, la tabella TCLU stabilisce le relazioni fra di esse, specificando quali tipi di classe possono essere associati o ereditati l'uno dall'altro. Entrambe queste tabelle

contribuiscono alla configurazione e alla gestione efficace del sistema di classificazione in SAP PM.

- *KLAGR*: si tratta di un ulteriore attributo collegato ad una tabella di supporto che in questo caso è *TCLG, Class Groups data*. È parte del sistema di classificazione e contiene informazioni sui gruppi di classi. Un gruppo di classi è un'entità organizzativa che consente di raggruppare insieme diverse classi per semplificarne la gestione e l'assegnazione.

CABN

Si passerà, ora, a fornire una descrizione della seconda tabella relativa alla gestione degli aspetti legati alle classi e alle caratteristiche. CABN viene definita come *Characteristic data* e, come è possibile intendere dal nome, rappresenta il concetto di caratteristica/attributo di una classe di oggetti. Contiene informazioni di carattere generale riguardanti le proprietà associabili ad ogni classe. Segue un immagine dei principali campi presenti nella tabella (Figura 3.19).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
ATINN	Internal characteristic	ATINN	NUMC	10	
ATNAM	Characteristic Name	ATNAM	CHAR	30	
ATFOR	Data type of characteristic	ATFOR	CHAR	4	
ANZST	Number of Characters	ANZST	INT2	5	
ANZDZ	Number of Decimal Places	ANZDZ	INT2	5	
ATVOR	Value with Plus or Minus Sign	ATVOR	CHAR	1	
ATSCH	Value with template	ATSCH	CHAR	30	
ATKLE	Case Sensitive	ATKLE	CHAR	1	
ATERF	Entry Required	ATERF	CHAR	1	
ATEIN	Single value	ATEIN	CHAR	1	
ATAME	Multilingual characteristic	ATAME	CHAR	1	
ATWME	Values in more than one language	ATWME	CHAR	1	
MSEHI	Unit of Measurement	MSEHI	UNIT	3	T006
ATDIM	Exponent display	ATDIM	INT2	5	

Figura 3.19: Tabella Characteristic Data

In questa struttura il campo chiave è rappresentato dall'attributo *ATINN*, il quale, come succede di solito in SAP, è definito come un identificatore interno di tipo numerico. Per quanto riguarda gli altri attributi, è presente una lunga lista di proprietà riconducibili al tipo di dato associabile alla caratteristica.

- *ATNAM*: stringa contenente il nome della caratteristica in questione.
- *ATFOR*: stringa descrittiva del tipo di dato associato alla caratteristica.
- *ANZST*: numero di caratteri associabile ad un eventuale data type testuale.
- *ANZDZ*: numero di cifre relative alla parte decimale di un eventuale data type numerico a virgola mobile.

- *ATVOR*: segno associabile ad un data type numerico.
- *ATKLE*: specifica se un eventiale tipo di dato testuale è case sensitive o meno.
- *ATERF*: campo che indica se il valore della caratteristica deve essere fornito obbligatoriamente o meno.
- *ATEIN*: indica se la caratteristica è univoca o se è possibile associare ad essa una serie di valori differenti.
- *MSEHI*: questo campo introduce un elemento chiave nello schema che si sta trattando, ovvero la *Unit of Measurement* (abbreviato UoM). Si tratta dell'unità di misura associabile ad una determinata caratteristica; è, infatti, associato alla tabella esterna di validazione *T006*, contenente tutte le informazioni relative all'unità di misura. Questa tabella, inoltre, è un esempio del collegamento interno fra i vari moduli di SAP ERP; infatti, T006 non fa capo a SAP PM, ma bensì al modulo SAP MM (Material Management).

KSML, AUSP e CAWN

Fornita una descrizione più dettagliata delle due tabelle principali riguardanti i concetti di classe e caratteristica, si passerà, ora, ad illustrare in maniera più sommaria le restanti tre tabelle delle cinque precedentemente introdotte. Le tre strutture dati KSML, AUSP e CAWN, sono tutte riconducibili all'assegnazione di valori alle caratteristiche riconducibili ad un oggetto, data la sua classe di appartenenza.

- *KSML*: viene definita da SAP come "Characteristics of a Class data" e, come è possibile capire leggendo il nome, rappresenta il punto di collegamento fra ogni classe di oggetti e le caratteristiche ad essa associabili. Segue un'illustrazione dei principali campi di interesse (Figura 3.20).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
CLINT	Internal Class Number	CLINT	NUMC	10	KLAH
ADZHL	Internal counter for archiving objects via engin. chg. mgmt	ADZHL	NUMC	4	
HERKU	Characteristic origin	HERKUNFT	CHAR	10	
IMERK	Internal characteristic	ATINN	NUMC	10	Assigned to domain
OMERK	Object-Dependent Characteristic	ATINO	NUMC	10	Assigned to domain
KLART	Class Type	KLASSENART	CHAR	3	TCLA
RELEV	Relevancy Indicator	RELEVANZKZ	CHAR	1	

Figura 3.20: Tabella Characteristics of a Class data

Come è possibile osservare, si ha sia un campo identificatore della classe in questione (*CLINT*), che un campo relativo alla caratteristica caratteristica da collegare (*IMERK*).

- *AUSP*: questa quarta tabella viene definita come "Characteristic Values data" e, secondo la documentazione di SAP, presenta al suo interno dati relativi agli effettivi valori di ogni caratteristica associata ad un determinato oggetto fisico (data la sua classe di appartenenza). Segue un'illustrazione degli attributi più rilevanti presenti in AUSP (Figura 3.21).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
OBJEK	Key of object to be classified	OBJNUM	CHAR	50	
ATINN	Internal characteristic	ATINN	NUMC	10	Assigned to domain
ATWRT	Characteristic Value	ATWRT	CHAR	30	
ATFLV	Internal floating point from	ATFLV	FLTP	16(16)	
ATAWE	Unit of Measurement	MSEHI	UNIT	3	T006
ATFLB	Internal floating point value to	ATFLB	FLTP	16(16)	

Figura 3.21: Tabella Characteristic Values data

Come è possibile osservare, il legame precedentemente descritto fra la caratteristica e l'oggetto fisico in questione è introdotto grazie alla presenza di due campi; il primo è *ATINN*, il quale permette di capire a quale attributo si sta facendo riferimento; il secondo è *OBJEK*. Questo secondo campo è di particolare interesse in quanto fa riferimento al concetto di "Object" introdotto durante la trattazione delle tabelle relative ad equipment e functional location. Specificare il valore di una caratteristica facendo riferimento all'identificatore dell'oggetto generico permette a questa tabella di descrivere valori di proprietà attribuibili ad ogni tipologia di oggetto fisico (descritto dal livello di astrazione introdotto da "Object"). Infine, fra i campi non chiave, è possibile osservare il campo *ATWRT*, contenente il valore vero e proprio della caratteristica in questione; quest'ultimo è espresso come una stringa di trenta caratteri; questa scelta ribadisce la volontà di SAP di lasciare quanta più libertà rappresentativa possibile all'utente finale (usando una stringa è possibile esprimere ogni tipo di valore, che sarà poi adattato al tipo specifico dall'applicazione di turno).

- *CAWN*: per concludere la trattazione relativa alle caratteristiche delle classi, si ha la tabella CAWN (Figura 3.22). Quest'ultima viene utilizzata in SAP per esprimere un livello di validazione sui valori attribuibili ad ogni caratteristica.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
ATINN	Internal characteristic	ATINN	NUMC	10	Assigned to domain
ATZHL	Int counter	ATZHL	NUMC	4	
ATWRT	Characteristic Value	ATWRT	CHAR	30	
ATSTD	Default value	ATSTD	CHAR	1	

Figura 3.22: Tabella Characteristic (allowed) Values data

Guardando i campi in essa presenti (Figura 3.22), è infatti facile capire la logica dietro questa struttura dati. Infatti, si fa riferimento alla tabella delle caratteristiche tramite il campo *ATINN* (caratteristica per la quale si sta esprimendo un possibile valore associabile), e si specifica un valore permesso tramite il campo *ATWRT*. A completare il campo chiave, oltre al riferimento alla caratteristica, è presente un contatore interno utilizzato per identificare tutti i valori possibili associati ad essa. Oltre a tutto ciò, SAP permette anche di definire quali fra i valori associabili alla caratteristica in questione è

da considerare come valore di default; questo grazie alla presenza del campo True o False *ATSTD* (che, logicamente, dovrà essere True solo per uno dei valori associabili alla caratteristica).

3.5.4 Bill of Materials - Assembly

In questa sezione verranno approfonditi i concetti di *Bill Of Materials* (*BOM*) e *Assembly*. Sono due elementi parzialmente sovrapponibili ed identificano un insieme di attrezzature. Il concetto di "material" fa riferimento a un oggetto che rappresenta beni o merci che possono essere utilizzati o consumati nel contesto delle attività di manutenzione. Questi materiali possono essere parti di ricambio, componenti, utensili, lubrificanti, o qualsiasi altro oggetto tangibile rilevante ai lavori di manutenzione sugli asset.

In generale, quindi, una BOM rappresenta, in SAP, un insieme di material non meglio definiti; con il concetto di assembly si vuole fare un passo in avanti, facendo riferimento ad un insieme di equipment. Si tratta, quindi, di un insieme di oggetti singolarmente manutenibili che vanno a formare un elemento più complesso, anche esso globalmente sottoponibile a manutenzione. Basti pensare ad un macchinario industriale sufficientemente elaborato da prevedere delle attività di manutenzione generiche (ad esempio, mirate a garantire il corretto funzionamento generale) e delle attività indirizzate in maniera più mirata ai singoli sottocomponenti.

Per quanto riguarda le tabelle che SAP sfrutta per dare forma a questi concetti, queste sono: *EQST*, *STKO* e *STPO*. Ora, saranno illustrate più in dettaglio la struttura e i campi di queste tabelle.

STKO

Questa prima tabella legata ai concetti precedentemente esposti viene descritta da SAP come "*BOM Header data*"; si tratta della struttura dati relegata al contenimento delle informazioni generali della Bill Of Materials. Nella seguente illustrazione (Figura 3.23) è possibile osservare i campi di maggior rilievo presenti in STKO.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
STLTY	BOM category	STLTY	CHAR	1	
STLNR	Bill of material	STNUM	CHAR	8	
DATUV	Valid-From Date	DATUV	DATS	8	
LKENZ	Deletion Indicator	LKENZ	CHAR	1	
LOEKZ	Deletion flag for BOMs	LOKNZ	CHAR	1	
BMEIN	Base unit of measure for BOM	BASME	UNIT	3	T006
BMENG	Base quantity	BASMN	QUAN	13(3)	
CADKZ	CAD Indicator	CADKZ	CHAR	1	
LABOR	Laboratory/design office	LABOR	CHAR	3	T024L
STKTX	Alternative BOM Text	STKTX	CHAR	40	
STLST	BOM status	STLST	NUMC	2	T415S

Figura 3.23: Tabella BOM Header data

Come è possibile osservare dalla figura, come attributi chiave della tabella, oltre ad un classico identificatore alfanumerico (*STLNR*), si ha un campo chiamato *STLTY - BOM Category*;

questo attributo permette di eseguire una distinzione fra diverse categorie di BOM, fra le quali è presente, anche, la Equipment BOM (corrispondente al concetto di Assembly); esso specifica la natura della lista di materiali aiutando a definirne l'uso o la funzione. Di seguito vengono riportati alcuni esempi di valori solitamente associati all'attributo STLTY, dando anche una spiegazione del loro significato.

- *1 - Material BOM*: questo tipo di lista di materiali è utilizzato per rappresentare la struttura di un prodotto finale in cui i componenti sono materiali fisici.
- *2 - Document BOM*: indica una lista di materiali utilizzata per associare documenti a un oggetto, come i documenti di progetto o le istruzioni di montaggio.
- *3 - Equipment BOM (Assembly)*: questo tipo di lista di materiali è utilizzato per rappresentare la struttura degli asset o degli impianti tecnici. Ad esempio, può essere utilizzato in SAP PM per definire la struttura di sistema complesso di produzione.
- *4 - Standard BOM*: indica una lista di materiali standard, che rappresenta una struttura di base comune per più varianti di prodotti.
- *5 - Document Structure*: utilizzato per rappresentare la struttura di documenti o documenti tecnici.
- *6 - Material BOM with History*: simile alla Material BOM, ma con la possibilità di mantenere una cronologia delle modifiche.

Altri attributi, presenti nella struttura dati, e degni di essere approfonditi sono i sequenti:

- *STLST - BOM Status*: rappresenta lo stato corrente della lista di materiali associata all'oggetto. Gli stati della lista di materiali possono variare in base al ciclo di vita della BOM e alle fasi del processo aziendale. Alcuni possibili valori che solitamente vengono associati a questo campo sono:
  - *01 - BOM attiva*: la lista di materiali è attiva e può essere utilizzata nelle transazioni aziendali.
  - *02 - BOM obsoleta*: la lista di materiali è considerata obsoleta o fuori uso, ma le informazioni possono essere ancora visualizzate.
  - *03 - BOM in preparazione*: la lista di materiali è in fase di preparazione o modifica.
  - *04 - BOM con restrizioni*: la lista di materiali è attiva ma con alcune restrizioni o limitazioni.
  - *05 - BOM bloccata*: la lista di materiali è bloccata e non può essere utilizzata nelle transazioni.
- *CASKZ - CAD Indicator*: questo campo è parte delle informazioni utilizzate per la gestione dei documenti tecnici all'interno di SAP PM. Associare documenti alle posizioni funzionali, alle attrezzature o ad altre entità consente di avere accesso rapido e organizzato alle informazioni pertinenti durante le attività di manutenzione e gestione degli impianti. Alcuni possibili valori possono essere i seguenti:
  - *"D"*: indica che si tratta di un documento di progettazione o di disegno tecnico associato all'oggetto tecnico.
  - *"C"*: può indicare un documento di controllo di qualità associato all'oggetto tecnico.

- *LKENZ - Delete Indicator*: è un flag (utilizzato anche in altre tabelle SAP) che indica se un record è stato contrassegnato per la cancellazione o meno. Quando un record viene contrassegnato con l'indicatore di cancellazione vuol dire che è stato programmato per l'eliminazione dal sistema, ma la cancellazione effettiva può avvenire in un secondo momento.

  L'uso dell'indicatore di cancellazione è parte integrante delle funzionalità di gestione dei dati in SAP per garantire che le informazioni obsolete o non più valide non siano più considerate nel sistema operativo, ma siano comunque mantenute per scopi di audit e storico.

  Quando un record viene contrassegnato con l'indicatore di cancellazione può essere oggetto di un processo di eliminazione successivo. La cancellazione effettiva può essere programmata in modo che i dati siano rimossi dal sistema solo dopo una certa data o dopo che sono state prese determinate misure. Ciò consente di gestire in modo controllato l'eliminazione dei dati.

STPO

Proseguendo con l'approfondimento delle tre tabelle precedentemente introdotte, l'attenzione sarà ora focalizzata sulla descrizione di STPO, anche detta "*BOM Item data*". Quest'ultima contiene informazioni relative alle parti (componenti) di una lista di materiali (BOM). Questi elementi possono far riferimento a svariate tipologie di oggetti, come materiali grezzi, parti di ricambio o semilavorati. In alcune industrie nelle BOM possono essere anche inclusi dei servizi, ad esempio la manodopera necessaria per l'assemblaggio o l'installazione di un prodotto. In figura 3.24 è possibile osservare un'illustrazione dei campi principali della tabella STPO.

Partendo dall'analisi dei campi chiave, nella Figura si puo vedere che essi sono gli stessi presenti nella tabella BOM Header Data, con l'aggiunta di un "*BOM Item node number*" (*STLKN*) identificativo del materiale all'interno della lista.

Allo stesso modo, osservando gli attributi descrittivi, è immediato accorgersi della presenza di alcuni elementi già visti in STKO, come la data di validità (*DATUV*), il Delete Indicator (*LKENZ*) e il CAD Indicator (*CADPO*). D'altra parte, però, nella struttura dati è presente anche un gran numero di campi relativi alle caratteristiche del materiale o, meglio ancora, relativi alla voce della lista in questione. Fra questi attributi troviamo dati come:

- *POSTP - Item category*: è un campo che indica la categoria dell'elemento all'interno della lista di materiali. Questo attributo definisce la natura o la funzione dell'elemento, consentendo di distinguere tra diversi tipi di componenti all'interno di una BOM. Di seguito alcune categorie comuni:

  - *Material Component*: indica un componente materiale fisico che sarà parte del prodotto finito.
  - *Document*: indica un documento associato all'elemento della BOM. Può includere documenti tecnici, manuali di istruzioni, specifiche, etc.
  - *Drawing* : indica un disegno tecnico o un documento grafico associato all'elemento.
  - *Document Link*: indica un collegamento a un documento o a un oggetto esterno.
  - *Operations* : indica un'operazione o un'attività di lavorazione associata all'elemento.
  - *Configuration Node*: indica un nodo di configurazione o variante all'interno della BOM.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
STLTY	BOM category	STLTY	CHAR	1	
STLNR	Bill of material	STNUM	CHAR	8	
STLKN	BOM item node number	STLKN	NUMC	8	
DATUV	Valid-From Date	DATUV	DATS	8	
LKENZ	Deletion Indicator	LKENZ	CHAR	1	
VGKNT	Predecessor Node	VGKNT	NUMC	8	
POSTP	Item Category (Bill of Material)	POSTP	CHAR	1	T418
MEINS	Component unit of measure	KMPME	UNIT	3	T006
MENGE	Component quantity	KMPMG	QUAN	13(3)	
NETAU	Indicator: Net scrap	NETAU	CHAR	1	
SCHGT	Indicator: Bulk Material	SCHGT	CHAR	1	
ERSKZ	Indicator: spare part	ERSKZ	CHAR	1	T413
RVREL	Indicator: item relevant to sales	RVREL	CHAR	1	TVRL
SANFE	Indicator: item relevant to production	SANFE	CHAR	1	
SANIN	Indicator: item relevant to plant maintenance	SANIN	CHAR	1	
SANKA	Indicator for relevancy to costing	CS_SANKA	CHAR	1	TCK08
SANKO	Indicator: item relevant to engineering	SANKO	CHAR	1	
SANVS	Indicator: high-level configuration	SANVS	CHAR	1	
STKKZ	PM assembly indicator	STKKZ	CHAR	1	
REKRI	Indicator: BOM is recursive	REKRI	CHAR	1	
REKRS	Indicator: recursiveness allowed	REKRS	CHAR	1	
CADPO	CAD Indicator	CADPO	CHAR	1	

Figura 3.24: Tabella BOM Item data

  - *Packing Material*: indica un materiale utilizzato per l'imballaggio del prodotto finito.
  - *Sales Kit*: indica un kit di vendita, che può essere un insieme di prodotti venduti insieme.
  - *Individual Purchase*: indica un componente che può essere acquistato individualmente.
  - *Stock Item*: indica un articolo di magazzino.
  - *Labor*: indica una componente di tipo manodopera o lavoro.

- *VGKNT - Predecessor node*: è uno dei campi più interessanti presenti in STPO in quanto permette di definire una struttura gerarchica fra i material associabili ad una BOM.

Il campo "Predecessor Node" indica quale nodo precede quello corrente nella gerarchia della BOM. In altre parole, specifica il componente o il nodo che deve essere posizionato prima dell'elemento attuale nella sequenza di assemblaggio. Questo attributo è utile per stabilire l'ordine di collegamento dei componenti.

Ad esempio, consideriamo una BOM di un prodotto complesso che include diverse parti. Se un componente A deve essere installato prima di un componente B, il campo "Predecessor Node" per il record corrispondente a B potrebbe indicare il numero di nodo relativo al componente A. In questo modo, la sequenza di assemblaggio può essere correttamente interpretata e gestita.

La struttura gerarchica della BOM e l'utilizzo del campo "Predecessor Node" aiutano a definire la sequenza di montaggio e a garantire che i componenti siano assemblati nell'ordine corretto durante il processo produttivo o di manutenzione.

- *MENGE - Component quantity*: si tratta di un valore che indica la quantità del material in questione da inserire nella BOM.
- *MEINS - Component Unit of measurement*: rappresenta l'unità di misura di base per il componente. Questo campo indica l'unità di misura di quantità per il componente specificato nella BOM. Ad esempio, la quantità di un material potrebbe essere misurata in "pezzi" (unità di misura "PC") mentre un altro componente nella BOM potrebbe avere la sua quantità misurata in "chilogrammi" (unità di misura "KG").
- *STKKZ - Assembly indicator*: indica se un componente nella lista di materiali è un'unità di assemblaggio o se è un componente semplice. Quando l'Assembly Indicator è impostato su "X" (attivo), significa che il componente è un'unità di assemblaggio, il che suggerisce che include ulteriori componenti all'interno di sé. In altre parole, è un sottoassemblaggio che deve essere assemblato o smontato in fasi successive del processo produttivo.

  D'altra parte, se l'Assembly Indicator è vuoto o non attivo, il componente è considerato un componente singolo o semplice, che non include ulteriori livelli di dettaglio nella lista di materiali. Questo tipo di componente rappresenta un oggetto tangibile o una parte che può essere utilizzata direttamente senza bisogno di ulteriori assemblaggi interni.

  L'uso di questo indicatore è cruciale per comprendere la struttura della BOM e come i vari componenti siano collegati tra loro nell'ambito del processo di produzione o assemblaggio.
- *Lista di Indicators*: oltre agli attributi sopra descritti, nell'immagine è possibile notare la presenza di una lunga lista di indicatori; questi ultimi servono per specificare l'ambito di utilizzo del material in questione. In base ai valori attribuiti a questi campi è possibile capire a quali scopi o rami aziendali può essere ricondotto l'utilizzo del componente (ad esempio, Produzione, Vendite, Ingegneria).

EQST

SAP definisce questa tabella come "Equipment to BOM Link data" e, come è facile intendere dal nome, si tratta di una struttura dati che rappresenta il collegamento fra gli equipment e le Bill Of Materials. Di fatto, quindi, questa tabella permette di stabilire i collegamenti tra gli asset e i materiali necessari per le attività di manutenzione, garantendo una corretta pianificazione e gestione delle risorse. Di seguito viene riportata un'illustrazione dei principali campi presenti nella tabella (Figura 3.25).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
EQUNR	Equipment Number	EQUNR	CHAR	18	EQUI
WERKS	Plant	WERKS_D	CHAR	4	T001W
STLAN	BOM Usage	STLAN	CHAR	1	T416
STLNR	Bill of material	STNUM	CHAR	8	

Figura 3.25: Tabella Equipment to BOM Link data

Gli attributi che vanno a formare la chiave identificativa della tabella (gli unici riportati in figura) sono i seguenti:

- *EQUNR - Equipment Number*: riferimento all'equipment in questione; dice per quale attrezzatura verrà utilizzato l'insieme di componenti specificati nella BOM in fase di manutenzione.
- *WERKS - Plant*: è un identificatore che indica a quale stabilimento o impianto è associato il collegamento tra un equipaggiamento specifico e la sua lista di materiali.
- *STLAN - BOM Usage*: indica l'uso specifico della lista di materiali associata a un determinato equipment. Questo campo fornisce informazioni sul contesto o lo scenario in cui la BOM è destinata ad essere utilizzata rispetto all'equipaggiamento specifico.

  Le possibili categorie di "BOM Usage" possono variare a seconda dell'implementazione specifica e delle esigenze aziendali; esse, solitamente includono:

  - *1 - As-Maintained (Come mantenuto)*: indica che la lista di materiali è progettata per rappresentare la configurazione attuale dell'equipaggiamento mantenuto. Questo può includere tutti i componenti installati e le relative quantità.
  - *2 - As-Planned (Come pianificato)*: indica che la lista di materiali riflette la configurazione pianificata o la struttura attesa dell'equipaggiamento in base a piani di manutenzione preventiva o interventi futuri.
  - *3 - As-Built (Come costruito)*: indica che la lista di materiali rappresenta la configurazione effettiva dell'equipaggiamento al momento della sua costruzione o assemblaggio.
  - *4 - Others (Altro)*: può indicare scenari specifici o usi particolari che non rientrano nelle categorie precedenti.

- *STLNR - Bill of material*: si tratta del riferimento diretto alla lista di componenti da collegare all'attrezzatura.

Questa struttura dati, quindi, da forma ad una relazione di tipo molti a molti fra gli equipment e le liste di materiali. Logicamente, questo vuol dire che, da una parte, per ogni equipment potranno essere utilizzati, in fase di manutenzione/riparazione, un numero n di BOM, e dall'altra, che ogni gruppo di materiali (ad esempio, se visto come un kit di riparazione) può essere utilizzato su diversi equipment (ad esempio, equipment appartenenti ad una stessa categoria con tutta probabilità avranno in comune lo stesso insieme di componenti da utilizzare in caso di riparazioni).

Per concludere l'approfondimento di questa terza tabella è necessario specificare che è stato scelto di tralasciare i campi descrittivi in quanto non sono stati reputati di interesse ai fini del progetto (si tratta di campi utilizzati da SAP per la gestione interna delle modifiche ai record; quindi campi relativi alla data e all'utente riconducibili all'ultimo update).

3.6 Elementi Maintenance Register

Come accennato nelle sezioni precedenti, nell'asset register troviamo tutti quei concetti le- gati in qualche modo alle risorse fisiche in possesso ad un'organizzazione, con annesse delle informazioni di carattere tecnico, come descrizioni, tipologia di apparecchiatura, valore economico, stato dell'asset, e molto altro ancora.

Già introdotto parzialmente nella sezione relativa allo stato dell'arte aziendale, il concetto di Maintenance Register identifica un registro, contestuale ad un singolo progetto, finalizzato al tracciamento delle attività di manutenzione svolte su asset, impianti o attrezzature all'interno dell'organizzazione. Questo registro, come accennato nella sezione introduttiva riguardante la storia della manutenzione, era una volta un vero e proprio libro fisico utilizzato al fine di documentare in modo dettagliato le operazioni preventive e correttive. Per la maggior parte dei progetti, le informazioni che il maintenance register fornisce riguardano:

- *Data e Ora dell'Intervento*: registra la data e l'orario in cui è stata eseguita l'attività di manutenzione.
- *Descrizione dell'Attività*: fornisce una descrizione dettagliata delle attività di manutenzione svolte. Esse potrebbero riguardare riparazioni, sostituzioni di componenti, pulizia, ispezioni e altro.
- *Asset Coinvolto*: specifica l'asset o gli asset che sono stati oggetto di manutenzione. Può includere informazioni dettagliate sull'asset, come il numero di identificazione, l'ubicazione e le caratteristiche specifiche.
- *Tipo di Manutenzione*: indica se l'attività di manutenzione è stata di tipo preventivo o correttivo. La manutenzione preventiva è programmata in anticipo per evitare guasti, mentre la manutenzione correttiva si occupa di problemi già verificatisi.
- *Risorse Coinvolte*: registra le risorse umane e materiali impiegate per svolgere l'attività di manutenzione. Ciò potrebbe includere i nomi degli operatori, il tempo dedicato, i materiali utilizzati e altre risorse necessarie.
- *Costi Associati*: documenta i costi associati all'attività di manutenzione, come costi della manodopera, costi dei materiali, costi di viaggio, etc.
- *Note e Commenti*: fornisce uno spazio per annotazioni, commenti o informazioni aggiuntive rilevanti per l'attività di manutenzione.
- *Stato dell'Asset dopo la Manutenzione*: descrive lo stato dell'asset dopo l'intervento di manutenzione. Ad esempio, indica se esso è stato riparato, sostituito o se sono state eseguite ulteriori azioni.
- *Firma o Approvazione*: se necessario, include spazi per l'approvazione del personale coinvolto nell'attività di manutenzione.

Si tratta perciò di uno strumento essenziale per garantire la tracciabilità e la conformità delle attività di manutenzione, facilitando l'analisi delle prestazioni degli asset nel tempo e aiutando a pianificare future attività di manutenzione in modo più efficace.

Come fatto per l'Asset Register, verranno in seguito approfondite ed illustrate le tabelle che SAP PM sfrutta per dar forma al Maintenance Register. Anche in questo caso, per ognuna delle strutture dati affrontate, il focus verrà posto maggiormente sui campi più rilevanti e sui collegamenti fra le varie entità, cercando sempre di trovare un significato pratico alle varie rappresentazioni fornite da SAP.

3.6.1 Task list - Operations

L'approfondimento degli elementi riconducibili al maintenance register seguirà un approccio dal basso; ciò vuol dire che si partirà illustrando i concetti più alla base per poi passare a descrivere quelli che si basano su di essi.

A tal proposito, al fine di una corretta comprensione, si ritiene corretto cominciare introducendo i concetti di *Task list* e *Operations*. Chiamati, rispettivamente, Cicli e Operazioni in lingua italiana, rappresentano due elementi congiunti da un legame stretto.

Un task list può essere definito come una lista strutturata di attività di manutenzione che devono essere eseguite su un asset o su un gruppo di asset correlati. Le attività presenti in questa lista sono le Operations. L'elenco può includere istruzioni dettagliate ed informazioni rilevanti necessarie per eseguire con successo le attività di manutenzione.

Una task list può essere associata a materiali, risorse e documentazione necessari per l'esecuzione delle attività, il che facilita la gestione degli approvvigionamenti e delle risorse durante l'esecuzione delle attività di manutenzione. Oltre a questo, la struttura di questi cicli è pensata per essere utilizzata più volte, ad esempio in manutenzioni preventive regolari; questo permette di standardizzare le attività di manutenzione e di semplificare il processo di pianificazione. Infine, le operazioni presenti in una task list possono essere organizzate gerarchicamente, con livelli e sotto-attività, andando a formare una struttura che consente di rappresentare in modo dettagliato le diverse fasi di una procedura di manutenzione.

Tornando alla mappatura interna di SAP PM, le tabelle che possono essere collegate ai due concetti sopra esposti sono *PLKO, PLAS, PLPO, PFLH, PLMZ* e *TAPL*. Seguirà, ora, un approfondimento di queste strutture dati.

PLKO

Descritta da SAP come "Task list - Header data", come tutte le tabelle viste in precedenza contenenti la nomenclatura "Header Data", contiene informazioni generali riguardanti il ciclo di operazioni. PLKO è fondamentale per la gestione dei task list in SAP PM; al suo interno si possono trovare dati riguardanti il tipo, il numero e lo stato delle task list, fornendo una base per la pianificazione e l'esecuzione delle attività di manutenzione. Di seguito, viene fornita un'illustrazione dei principali campi che formano la tabella (Figura 3.26).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
PLNTY	Task List Type	PLNTY	CHAR	1	TCA01
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
PLNAL	Group Counter	PLNAL	CHAR	2	
VERWE	Task list usage	PLN_VERWE	CHAR	3	T411
WERKS	Plant	WERKS_D	CHAR	4	T001W
STATU	Status	PLNST	CHAR	3	T412
PLNME	Task list unit of measure	PLNME	UNIT	3	T006
LOSVN	From Lot Size	LOSGRVON	QUAN	13(3)	
LOSBS	To lot size	LOSGRBIS	QUAN	13(3)	
VAGRP	Responsible planner group/department	VAGRP	CHAR	3	T024A
KTEXT	Task list description	PLANTEXT	CHAR	40	
KOKRS	Controlling Area	KOKRS	CHAR	4	TKA01

Figura 3.26: Tabella Task List Header data

Nella tabella possiamo trovare tre attributi principali (ad esclusione del solito campo Client) che identificano la chiave. Ogni task list è, infatti, identificato da un "Task list group" (attributo *PLNNR - Key for task list group)* e da un contatore interno del gruppo (attributo *PLNAL - Group Counter*). Questi due attributi permettono di identificare ogni ciclo tramite un gruppo di appartenenza e un contatore interno al gruppo. Ad esempio, si potrebbero avere dei task list group identificati dalla tipologia di attrezzatura su cui agiscono, oppure dal tipo di manutenzione, oppure tramite un altro criterio rilevante per la gestione delle attività; dati questi gruppi, si avrà una divisione interna che identificherà i singoli cicli nello specifico.

Come sopra accennato, è presente, anche, un terzo campo nella chiave, chiamato *PLNTY - Task list type*; quest'ultimo denota le caratteristiche e le funzionalità specifiche del ciclo indicando il contesto e le regole che la lista seguirà. In sintesi, questo campo definisce il tipo di attività o processo per cui la lista di attività è stata creata. Ad esempio, potrebbe essere associato a manutenzione preventiva, manutenzione correttiva, ispezioni, lavori di produzione, o altri tipi specifici di attività.

Ora, saranno approfonditi i campi descrittivi della tabella PLKO.

- *VERWE - Task list usage*: è un attributo che serve per definire lo scopo o l'utilizzo di una lista di attività. Indica il contesto o la situazione in cui la task list può essere impiegata. Questo campo fornisce informazioni importanti sulla finalità della lista di attività, facilitando la corretta assegnazione e l'utilizzo delle task list in diversi scenari.

- *STATU - Task list status*: indica lo stato corrente di una lista di attività. Esso riflette la fase in cui si trova task list nel suo ciclo di vita, fornendo informazioni sullo stato della sua creazione, modifica, attivazione, esecuzione o completamento. Il "Task List Status" è un elemento chiave per tracciare e gestire lo stato delle attività di manutenzione e produzione associate ad un ciclo specifico.

  Alcuni esempi di valori associabili a questo campo sono:

  - *Created*: la task list è stata creata, ma potrebbe non essere ancora stata attivata per l'uso pratico. In questa fase, potrebbe essere soggetta a revisioni o modifiche.
  - *Released*: la task list è stata rilasciata e quindi resa attiva per l'uso. Questo significa che la lista di attività è pronta per essere utilizzata nell'esecuzione di lavori di manutenzione o produzione.
  - *Active*: la lista è attiva e può essere utilizzata per pianificare ed eseguire attività di manutenzione o produzione. È la fase in cui la lista è pronta per essere messa in pratica.
  - *Complete*: la lista ha completato il suo ciclo di vita e tutte le attività associate sono state eseguite con successo. In questa fase, potrebbero essere disponibili informazioni sulle prestazioni o sull'esito delle attività.
  - *Obsolete*: la task list è obsoleta e non dovrebbe essere utilizzata. Potrebbe essere stata sostituita da versioni più recenti o dichiarata non valida per altri motivi.
  - *Locked*: la lista potrebbe essere bloccata per impedire modifiche non autorizzate. Questo stato è particolarmente rilevante durante l'esecuzione di attività critiche o in determinati scenari di revisione.

- *PLNME - Task list unit of measurement*: fa riferimento all'unità di misura associata alle quantità o grandezze espresse nelle liste di attività. Questo campo consente di specificare l'unità di misura in cui vengono definite le quantità per le attività all'interno della task list.

- *LOSVN/LOSBS - From/To lot size*: questi due campi indicano la dimensione del lotto da cui inizia la task list. Essi sono particolarmente rilevanti quando si tratta di attività di manutenzione preventive e di gestione delle attività di manutenzione basate su quantità definite.

  Il termine "lot size" si riferisce alla quantità specifica di un oggetto o di un materiale che viene prodotto o gestito in un'unica operazione o lotto. Nel contesto della manutenzione, potrebbe essere la quantità di un determinato materiale o componente che richiede un'attività di manutenzione preventiva o correttiva.

  Va notato che la specifica del lotto potrebbe non essere pertinente per tutte le task list, e il campo potrebbe essere vuoto in molti casi.

- *VAGRP - Responsible planner group*: fa riferimento al gruppo di pianificatori responsabili delle liste di attività ed è parte integrante della gestione delle attività di manutenzione in SAP PM; consente di associare le task list a un gruppo specifico di pianificatori responsabili.

- *KTEXT - Task list description*: una classica descrizione testuale atta a spiegare agli utenti l'uso o il fine del ciclo. Nella tabella è anche presente un riferimento alla lingua in cui viene espressa questa descrizione.

- *KOKRS - Controlling Area*: questo attributo, presente anche in altre tabelle viste nelle pagine precedenti, è utilizzato per associare una task list a una specifica area di controllo (controlling area). La controlling area viene utilizzata per la contabilità interna e la gestione dei costi all'interno dell'organizzazione. L'effettivo utilizzo e la rilevanza del campo possono variare a seconda della configurazione specifica di SAP PM nell'ambiente organizzativo; l'associazione con la controlling area è parte integrante della gestione finanziaria e del controllo delle attività di manutenzione in SAP PM.

PLPO

Si andrà, ora, ad approfondire il contenuto della tabella PLPO, identificata da SAP come "Task list - operation/activity data"; prima di procedere con la descrizione dei campi e della funzione generale della tabella è necessario precisare che, a causa dell'alta complessità di questa struttura dati, si è scelto di ridurre notevolmente il numero di campi riportati; questo perché SAP fornisce un'altissima flessibilità di utilizzo al concetto di "Operation" e, di conseguenza, la struttura dati contiene al proprio interno un elenco di attributi vastissimo, necessario quando una tabella deve poter essere utilizzabile in una miriade di contesti ed applicazioni differenti.

Fatta questa precisazione, di seguito si riporta un'illustrazione della struttura ridotta di PLPO (Figura 3.27)

Partendo dall'analizzare i campi che formano la chiave della tabelle, si nota che questi ricalcano i campi visti per PLKO, con l'aggiunta di un quarto campo, chiamato *ZAEHL - Internal counter*. Di norma, il prefisso "Z" nei nomi dei campi in SAP indica che si tratta di un campo personalizzato, aggiunto dall'utente o da un consulente SAP per estendere o adattare la funzionalità di base del sistema secondo le esigenze specifiche dell'azienda.

Il campo "Internal Counter" è, in questo caso, utilizzato per assegnare un identificatore univoco e crescente a ciascun record nella tabella PLPO. I restanti tre campi della chiave sono il riferimento al ciclo in cui l'operazione è inserita.

Passando, ora, ad approfondire i campi descrittivi che si è scelto di riportare, essi forniscono un quadro generale che specifica varie caratterstiche legate allo svolgimento dell'attività.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
PLNTY	Task List Type	PLNTY	CHAR	1	TCA01
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
PLNKN	Number of the task list node	PLNKN	NUMC	8	
ZAEHL	Internal counter	CIM_COUNT	NUMC	8	
SUMNR	Node number of the superior operation	SUMKNTNR	NUMC	8	
VORNR	Operation/Activity Number	VORNR	CHAR	4	
STEUS	Control key	STEUS	CHAR	4	T430
WERKS	Plant	WERKS_D	CHAR	4	T001W
KTSCH	Standard text key	KTSCH	CHAR	7	T435
LTXA1	Operation short text	LTXA1	CHAR	40	
DAUNO	Normal duration of the activity	DAUNOR	QUAN	5(1)	
DAUNE	Normal duration/unit	DAUNORE	UNIT	3	T006
DAUMI	Minimum activity duration	DAUMIN	QUAN	5(1)	
DAUME	Unit for the minimum duration	DAUMINE	UNIT	3	T006
DDEHN	Indicator: flexible duration	DAUDEHN	CHAR	1	
PPRIO	Priority	PRIORITAET	CHAR	2	
ISTNR	Bill of material	STNUM	CHAR	8	
TPLNR	Functional Location	TPLNR	CHAR	30	Assigned to domain
EQUNR	Equipment Number	EQUNR	CHAR	18	Assigned to domain

Figura 3.27: Tabella Task list - operation/activity data

- *SUMNR - Node number of the superior operation*: è uno dei campi più interessanti presenti nella tabella; permette di definire una struttura gerarchica fra le operazioni, identificando una "Superior operation", ovvero un'attività che deve necessariamente essere svolta prima dell'inizio dell'operazione in questione.
- *VORNR - Operation/Activity number*: può essere visto come un secondo identificatore dell'operazione all'interno del ciclo; spesso in Pansoinco questi valori seguono i multipli di 10 e, quindi, per ogni tasklist si avranno delle operazioni identificate da valori come 10, 20, 30, etc.

  È proprio a questo campo che fa riferimento l'attributo SUMNR per definire la gerarchia delle attività.
- *STEUS - Control Key*: la chiave di controllo influenza il comportamento dell'ordine di manutenzione quando viene eseguita la lista di attività. Un ordine di manutenzione è un documento che viene utilizzato per pianificare, eseguire e monitorare attività di manutenzione su attività, apparecchiature, impianti o altre risorse gestite da un'organizzazione; esso fornisce una struttura organizzativa per gestire e tracciare tutte le fasi coinvolte nelle operazioni di manutenzione. Ogni operazione, quindi, in base alla frequenza associata, genererà degli ordini di manutenzione ad indicare la necessità di

svolgere dei lavori; la control key è un parametro che influisce sulle modalità con le quali vengono generati questi ordini.

Alcuni aspetti legati ai "work order" su cui la chiave di controllo può influire sono:

- *Creazione e Pianificazione*: gli ordini di manutenzione vengono creati per pianificare le attività di manutenzione in anticipo. Durante la creazione dell'ordine, vengono specificate informazioni cruciali, come la descrizione del lavoro, la data di inizio e fine previste, le risorse necessarie, i materiali richiesti e altri dettagli pertinenti.
- *Tipo di ordine*: gli ordini di manutenzione possono essere di diversi tipi, tra cui ordini di manutenzione preventiva, correttiva o di ispezione. Ogni tipo di ordine è progettato per affrontare un aspetto specifico delle attività di manutenzione.
- *Stato dell'ordine*: gli ordini di manutenzione attraversano diversi stati durante il loro ciclo di vita, come "Creato", "Pianificato", "In Esecuzione", "Completato" o "Chiuso". Questi stati riflettono la fase corrente in cui si trova l'ordine.

- *WERKS - Plant*: questo campo permette di specificare un impianto che verrà considerato come la locazione di default nella quale l'attività verrà svolta.

- *KTSCH - Standard Text Key*: con questo attributo si introduce il concetto di standard text, il quale altro non è che un testo utilizzato per fornire dettagli aggiuntivi, spiegazioni o istruzioni relative alle operazioni specificate nella lista di attività. Per contenere queste descrizioni, come già accennato in precedenza per altri campi descrittivi, SAP utilizza delle tabelle esterne, sia per i soliti motivi legati alla lingua, sia per ragioni legate alla lunghezza del testo.

- *Campi legati alla durata*: sono presenti nella tabella una serie di campi, chiamati *DAUNE, DAUMI, DAUNO* e *DAUME*, i quali servono a fornire informazioni legate alla durata che l'operazione dovrebbe avere. In particolare, gli attributi *DAUNO - Normal duration* e *DAUNE - Normal duration unit* servono a specificare la durata che, idealmente, e in condizioni normali, l'operazione dovrebbe avere, specificando anche l'unità di misura usata per misurare questo tempo; invece i restanti due campi *DAUMI - Minimun duration* e *DAUME - Minimun duration unit* ne specificano la durata minima.

  Infine, è anche presente un quinto campo chiamato *DDEHN - Flexible duration* che specifica se nel contesto operativo è ammesso che la durata dell'operazione subisca degli aggiustamenti.

- *PPRIO - Priority*: indica la priorità associata a una specifica operazione all'interno di una lista di attività; il campo è utilizzato per definire l'importanza relativa delle operazioni all'interno di una task list ed influisce sulla sequenza in cui tali operazioni vengono eseguite durante l'esecuzione delle attività di manutenzione. Al contrario di quello che si potrebbe pensare, solitamente, le operazioni con una priorità più alta vengono eseguite prima di quelle con una priorità più bassa. La possibilità di assegnare questo valore alle operazioni consente una maggiore adattabilità nelle operazioni di manutenzione, in modo che le risorse possano essere allocate in base alle esigenze più critiche.

- *ISTNR - Bill of Material*: questo campo rimanda all'ultimo concetto visto durante l'approfondimento degli elementi riconducibili all'asset register; nell'ambito di questa tabella, l'attributo permette di specificare un elenco di componenti o parti che sono necessarie per eseguire un'attività specifica o per effettuare la manutenzione di un oggetto specifico. In altre parole, la BOM, in questo contesto, identifica i materiali o le risorse necessarie per completare l'attività in questione.

- *TPLNR/EQUNR - Functional location/Equipment*: per concludere, fra gli attributi della tabella è possibile notare due riferimenti, rispettivamente alle tabelle IFLO e EQUI; in questo contesto, i due collegamenti servono nel caso in cui una determinata operazione dovesse essere unicamente riconducibile ad un solo equipment o ad un unica sede tecnica.

PLAS

Questa tabella, definita come "Task list - selection of operations/activities data" ha il solo scopo di creare il collegamento fra un ciclo e un operazione espressa tramite il suo "Operation Number" (i numeri 10, 20, 30 che si specificavano nell'esempio). SAP quindi sceglie di dedicare una tabella apposita all'inserimento di operazioni nelle task list; è quindi presente un certo grado di ridondanza, causato dalla presenza dell'informazione relativa alla task list di riferimento sia in PLAS che in PLPO.

Per ragioni legate alla brevità, non verranno approfondite le ragioni dietro questa ridondanza, ma ci si limiterà a fornire un'illustrazione della struttura della tabella (Figura 3.28) la quale esprime tutta la sua utilità tramite i soli campi presenti nella chiave (di fatto la tabella rappresenta un collegamento).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
PLNTY	Task List Type	PLNTY	CHAR	1	TCA01
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
PLNAL	Group Counter	PLNAL	CHAR	2	
PLNFL	Sequence	PLNFOLGE	CHAR	6	
PLNKN	Number of the task list node	PLNKN	NUMC	8	
ZAEHL	Internal counter	CIM_COUNT	NUMC	8	

Figura 3.28: Tabella Task list - selection of operations/activities data

Per concludere la breve parentesi riservata a questa struttura dati, si vuole precisare il tipo di relazione espresso da questa tabella; infatti, tramite la sua introduzione, SAP consente un rapporto molti a molti fra operazione e ciclo. Grazie infatti alla presenza dell'identificatore interno in PLPO sarà possibile attribuire una stessa operazione (o meglio, dei duplicati della stessa operazione) a più task list differenti.

Tabelle TAPL e PLMZ

Per completare l'approfondimento di tutte le tabelle riguardanti i concetti di operations e task list introdotte in precedenza, mancherebbero soltanto le tabelle *TAPL* e *PLMZ*; tuttavia queste due strutture dati non verranno approfondite con lo stesso livello di dettaglio con il quale sono state trattate le tabelle precedenti. Questo perché in entrambi i casi, esse ribadiscono dei concetti già spiegati durante la trattazione di attributi presenti in altre tabelle.

Di conseguenza, verrà fornita per ognuna una descrizione sommaria delle funzionalità, senza scendere nel particolare di ogni attributo.

Per quanto riguarda la prima delle due, ovvero la tabella TAPL, essa viene descritta come *"Allocation of task list to functional location"*; come già accennato svariate volte, SAP è un sistema che intende fornire quanta più libertà e flessibilità di rappresentazione possibile agli utenti finali, e questa tabella è un'ulteriore conferma a questo approccio. Di fatto, è

una struttura dati che permette di avere una relazione di tipo molti a molti tra task list e functional location di riferimento; approfondendo la tabella PLKO erano stati introdotti due attributi che permettevano di specificare una "reference functional location" e un "reference equipment". Quindi nel caso in cui la relazione uno a molti dovesse essere restrittiva per la rappresentazione del contesto reale, si potrebbe ricorrere proprio alla tabella TAPL. Di seguito, viene riportata un'illustrazione della struttura dati (Figura 3.29).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
TPLNR	Functional Location	TPLNR	CHAR	30	IFLOT
PLNTY	Task List Type	PLNTY	CHAR	1	TCA01
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
PLNAL	Group Counter	PLNAL	CHAR	2	
ZKRIZ	Counter for additional criteria	DZKRIZ	NUMC	7	
ZAEHL	Internal counter	CIM_COUNT	NUMC	8	
IWERK	Maintenance Planning Plant	IWERK	CHAR	4	T001W
DATUV	Valid-From Date	DATUV	DATS	8	
TECHV	Technical status from	TECHV	CHAR	12	
LOEKZ	Deletion Indicator	LKENZ	CHAR	1	
PARKZ	Indicator: inactive changes	PARKZ	CHAR	1	
AENNR	Change Number	AENNR	CHAR	12	AENR
ANDAT	Date record created on	ANDAT	DATS	8	
ANNAM	User who created record	ANNAM	CHAR	12	
AEDAT	Changed On	AEDAT	DATS	8	
AENAM	Name of Person Who Changed Object	AENAM	CHAR	12	

Figura 3.29: Tabella Allocation of task list to functional location

La tabella PLMZ viene chiamata "Allocation of bill of material to operations". Il concetto che sta dietro a questa struttura dati è lo stesso visto per TAPL; infatti, durante la trattazione di PLPO, era stata specificata la presenza di un attributo (ISTNR - Bill of Material) finalizzato al legame dell'operazione in questione con una BOM precisa. Come per la tabella precedente, anche qui SAP vuole permettere agli utenti di scegliere la rappresentazione più aderente al contesto applicativo (uno a molti o molti a molti) e, di conseguenza, con PLMZ consente sia di legare più di una BOM ad una stessa operation che associare una stessa bill of material a più attività differenti. Verrà ora riportata una rappresentazione dei campi presenti nella tabella (Figura 3.30).

3.6.2 Strategy

Dopo aver fornito una panoramica sui concetti di Task List e Operations secondo SAP, si passerà ora ad introdurre un ulteriore elemento, fondamentale per il maintenance register, che è il concetto di *Strategia*.

Col termine "Maintenance Strategy" ci si riferisce ad un approccio pianificato ed organizzato per gestire le attività di manutenzione degli impianti, finalizzato ad ottimizzare le

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
PLNTY	Task List Type	PLNTY	CHAR	1	Assigned to domain
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
ZUONR	No. for material comp. allocation to task list	CIM_ZUORD	NUMC	8	
ZAEHL	Internal counter	CIM_COUNT	NUMC	8	
LOEKZ	Deletion Indicator	LKENZ	CHAR	1	
PARKZ	Indicator: inactive changes	PARKZ	CHAR	1	
PLNAL	Group Counter	PLNAL	CHAR	2	
PLNFL	Sequence	PLNFOLGE	CHAR	6	
PLNKN	Task list node number for operation	CIM_PLNKN	NUMC	8	
STLTY	BOM category	STLTY	CHAR	1	
STLNR	Bill of material	STNUM	CHAR	8	
STLAL	Alternative BOM	STALT	CHAR	2	
STLKN	Node Number	KNTNR	NUMC	8	
GP_MATNR	Material Number	MATNR	CHAR	18	Assigned to domain
GP_WERKS	Plant	WERKS_D	CHAR	4	Assigned to domain
GP_UVORN	Suboperation	UVORN	CHAR	4	

Figura 3.30: Tabella Allocation of bill of material to operations

operazioni, migliorare l'efficienza e garantire che la manutenzione venga eseguita in modo appropriato. Stabilire una strategia di manutenzione ha, quindi, come scopo quello di massimizzare la disponibilità degli impianti e prolungare la loro RUL (Remaining Useful Life).

Nella pratica, in SAP, la strategia viene sfruttata per stabilire quando le attività di manutenzione devono essere svolte nel contesto di un piano di manutenzione (concetto che verrà approfondito più avanti). La maintenance strategy permette di definire dei "Package" contenenti informazioni relative ad una certa frequenza di esecuzione. Legando, quindi, una strategia ad una serie di operazioni, queste ultime potranno avere una o più frequenze di esecuzione, sulla base di quelle definite nella strategia.

Le tabelle che in SAP permettono di dare forma a questi concetti sono T351, T351P e PLWP. Seguirà, ora, un approfondimento di queste tre tabelle, molto importanti per la definizione del maintenance register.

T351P

Nonostante potrebbe sembrare controintuitivo, si ritiene più utile al fine di una comprensione lineare dei concetti sopra introdotti, iniziare questo approfondimento partendo dalla tabella relativa alle frequenze di manutenzione, ovvero *T351P*; questa struttura dati viene anche chiamata "*Maintenance packages data*" e contiene tutte le informazioni necessarie a stabilire una frequenza precisa da associare a delle attività di manutenzione.

Una frequenza di manutenzione in SAP, può essere definita in due modi principali.

Da una parte si hanno le *frequenze time-based* che, come è possibile capire dal nome, definiscono una frequenza di manutenzione basata su calendario. Un esempio di cadenze definite da questo tipo di pacchetti potrebbero essere "Ogni 2 settimane", "Ogni giorno", "Ogni 2 anni" etc.

Si tratta, quindi, di interventi svolti ad intervalli regolari che non dipendono da nessun fattore aggiuntivo.

Oltre alle frequenze time-based, l'altra tipologia di frequenze che è possibile riscontrare in SAP sono le *frequenze performance-based*. In questo caso non ci si basa più su un calendario, ma bensì su determinati parametri operativi dell'asset in questione. Alcuni esempi di frequenze appartenenti a questa categoria potrebbero essere "Ogni 2000 ore di funzionamento", "Quando la temperatura raggiunde X gradi" oppure "Quando si accende la spia Y". È facile comprendere come, al contrario della categoria precedente, in questo caso non si hanno interventi con cadenza regolare; la frequenza con la quale verranno svolte le attività dipenderà dall'utilizzo che verrà fatto dell'asset. Ad esempio, un utilizzo intensivo potrebbe portare ad interventi più ravvicinati, mentre, al contrario, un uso saltuario implicherebbe una manutenzione più rara.

Fornita la definizione di questi due approcci, è molto importante specificare che, nel contesto aziendale, molto frequentemente questi ultimi vengono usati contemporaneamente. Ciò vuol dire che si definisce sia una frequenza costante (approccio time-based) che una frequenza basata sul funzionamento (approccio performance-based). Ad esempio, si potrebbe voler specificare che un macchinario ha necessità di essere controllato ogni 1000 ore di attività, ma al contempo in maniera periodica ogni anno; in questo caso se le 1000 ore di funzionamento non dovessero essere raggiunte entro l'anno, le attività di controllo verrebbero svolte ugualmente.

Fornita questa breve panoramica sul concetto di pacchetto, si passerà, ora, a descrivere la struttura dati fisica che in SAP dà forma a tutto ciò, ovvero T351P. A tal fine viene riportata di seguito un'illustrazione dei campi principali presenti nella tabella (Figura 3.31).

Come è possibile capire osservando l'immagine, i campi chiave presenti nella tabella, oltre a Client, sono due: *STRAT - Maintenance strategy* e *ZAEHL - Maintenance Package Number*; questi ultimi fanno si che una frequenza in SAP verrà identificata a partire dalla strategia in cui è inserita e da un "contatore" interno (quindi, ad esempio, pacchetto 2 della strategia S1). Da notare il riferimento esterno del campo STRAT alla tabella T351.

Per quanto riguarda i campi descrittivi inseriti nella tabella, è possibile trovare delle informazioni legate ai due approcci visti in precedenza. Gli attributi più importanti sono i seguenti:

- *ZEIEH e LEIEH*: questi due attributi, descritti ,rispettivamente, come "Unit for the performance of maintenance tasks" e "Maintenance strategy activity unit", sono relativi a due unità di misura (notare il riferimento esterno alla tabella T006 - UoM). In particolare, ZEIEH si riferisce all'unità di misura da utilizzare per misurare la caratteristica dell'asset che ne determinerà la frequenza di manutenzione; esempi di valori associabili a questo attributo potrebbero essere unità di misura temporali (Working Hours, Days, Months etc.) oppure unità di misura di parametri funzionali (ad esempio, gradi-temperatura o bar-pressione etc.).

  L'attributo LEIEH, invece, viene "valorizzato" quando si ha a che fare con un approccio time-based; di conseguenza, le uniche unità di misura che potranno comparire in questo campo sono di tipo temporale, quindi, ad esempio, giorni, settimane, mesi, oppure anni etc.

- *ZYKZT e ZYKLS*: questi due campi, rispettivamente chiamati "Maintenance package cycle/offset" e "Total counter reading on call date", identificano il valore relativo alla

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
STRAT	Maintenance strategy	STRAT	CHAR	6	T351
ZAEHL	Maintenance Package Number	PAKETZAEHL	NUMC	2	
ZEIEH	Unit for the performance of maintenance tasks	DZEIEH	UNIT	3	T006
LEIEH	Maintenance strategy activity unit	LEIEH	UNIT	3	T006
ZYKZT	Maintenance package cycle/offset	DZYK_ZEIT	FLTP	16(16)	
ZYKLS	Total counter reading on call date	DZYK_LEIST	FLTP	16(16)	
OFFZT	Start offset of a time-based maintenance package	DZYK_OFFZT	FLTP	16(16)	
OFFLS	Start offset of a performance-based maintenance package	DZYK_OFFLS	FLTP	16(16)	
HIERA	Nature of the maintenance packages (additional, replacement)	HIERARCHIE	NUMC	2	
PUFFP	Lead Float in Days	PUFF_VORL	NUMC	3	
PUFFN	Follow-up float in days	PUFF_NACHL	NUMC	3	
PERKN	Period type indicator for maintenance package	PERIODEKNZ	CHAR	1	
EROEF	Call horizon in days (per maintenance package)	EROEF	NUMC	3	

Figura 3.31: Tabella Maintenance packages data

frequenza di intervento. ZYKZT verrà "valorizzato" in un contesto time-based e di conseguenza farà riferimento all'unità di misura espressa da LEIEH, mentre, al contrario, ZYKLS assumerà il valore di performance e, di conseguenza, farà riferimento alla UoM espressa da ZEIEH.

- *OFFZT e OFFLS*: si tratta di attributi che esprimono il concetto di offset (nel caso time e performance based), ovvero il delay iniziale da tenere in conto quando i valori di frequenza e performance iniziano a scorrere. Ad esempio, definendo una frequenza di manutenzione di 1000 ore, ed un offset iniziale di 800 ore, il primo intervento verrà svolto effettivamente dopo 200 ore, per poi ricominciare il ciclo delle 1000 ore.

- *HIERA - Nature of maintenance packages*: questo attributo indica la tipologia o la natura dei pacchetti di manutenzione definiti nel sistema.

  Il termine "nature" in questo contesto potrebbe riferirsi al tipo di attività di manutenzione inclusa nel pacchetto. Ad esempio, quest'ultimo potrebbe essere classificato in base al tipo di lavoro da svolgere, come manutenzione preventiva, correttiva, ispettiva, etc. Questa classificazione consente di organizzare e raggruppare in modo logico le attività di manutenzione all'interno del sistema.

  L'utilizzo di questa informazione può facilitare la pianificazione e l'esecuzione delle attività di manutenzione, consentendo agli utenti di identificare rapidamente la natura o il tipo di lavoro coinvolto in un determinato pacchetto. Inoltre, può essere utile per scopi di reportistica e analisi, permettendo di valutare le distribuzioni delle attività di manutenzione in base alla loro tipologia.

- *EROEF - Call horizon in days*: con questo attributo si introduce un altro concetto chiave in SAP, ovvero quello del *Call Horizon*, anche chiamato orizzonte di apertura. Questo valore, in questo caso espresso in giorni, indica quanto prima della data effettiva di manutenzione verrà generato l'ordine di manutenzione. Ad esempio, definendo una frequenza di intervento di 250 giorni ed un call horizon di 20 giorni, l'ordine di lavoro verrà generato dal sistema dopo 230 giorni, ovvero 20 giorni prima della scadenza effettiva. Questo concetto può essere riscontrato anche in altre tabelle SAP, e nella maggior parte dei casi viene espresso come una percentuale dell'intervallo di tempo in questione (ad esempio, una call-horizon pari all'80 percento su 250 giorni corrisponderebbe ad un anticipo di 200 giorni).

T351

Data una definizione alla struttura dati dei package, si passerà ora ad affrontare il concetto di strategia, parlando della tabella *T351*; questa tabella viene descritta da SAP come "*Maintenance strategy data*" e, al suo interno, possono essere trovate informazioni che definiscono le caratteristiche generali di una strategia. Approfondire i campi presenti nella struttura dati è utile a comprendere a pieno il concetto di strategia e le informazioni in essa contenute. Viene riportata, di seguito, una rappresentazione dei campi principali presenti nella tabella (Figura 3.32).

Come unico campo chiave della tabella viene usato *STRAT - Maintenance strategy*, ovvero un idenificatore alfanumerico della strategia (campo al quale si fa riferimento nella tabella T351P vista in recedenza).

Per quanto riguarda i campi descrittivi presenti in T351, i più rilevanti agli scopi del progetto vengono riportati di seguito:

- *Campi ZEIEH/LEIEH e ZYK1G/ZYK2G*: questi attributi rispecchiano dei concetti già visti durante l'approfondimento della tabella T351P. In particolare, servono nel caso in cui, per la strategia in questione, è sufficiente definire una frequenza unica (time o performance based che sia); di conseguenza, SAP permette di specificare l'unità di misura e il valore/frequenza del ciclo direttamente in questa struttura dati. Questa scelta è, come al solito, riconducibile alla flessibilità rappresentativa garantita da SAP.

- *Smallest step between two maintenance packages*: anche in questo caso sono presenti due attributi riconducibili ad un concetto comune; questo step fa riferimento alla minima differenza che può intercorrere fra due pacchetti. Ad esempio, se lo smallest step dovesse essere di 2 settimane, non si potrebbero avere dei pacchetti la cui differenza di frequenze è inferiore a tale valore (esempio un pacchetto con frequenza 2 settimane e un altro con frequenza di 3).

- *Tolerance for early/late completion*: questi due attributi sono dei valori numerici rappresentanti delle percentuali. Essi identificano la tolleranza ammessa per la fine anticipata/ritardata dell'attività in questione. Le percentuali sono calcolate sulla base della durata dell'operazione in questione, che costituisce il 100%.

- *Shift Factor for early/late completion*: anche in questo caso si ha a che fare con due percentuali; questi valori sono strettamente collegati ai due attributi visti precedentemente e completano il quadro riguardante gli effetti di un ritardo/anticipo sulla fine di un'attività manutentiva. Con "Shift Factor" si intende lo spostamento della data del prossimo intervento, calcolato aggiungendo/sottraendo una percentuale del ritardo/anticipo del ciclo corrente.

Ad esempio, supponendo di avere un'attività con durata stimata di 100 giorni, collegata ad un pacchetto con shift factor for late completion del 20%, se l'attività dovesse ritardare di 10 giorni il suo completamento, l'esecuzione del prossimo ciclo dell'operazione sarebbe posticipato di 2 giorni (20% dei 10 giorni di ritardo).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
STRAT	Maintenance strategy	STRAT	CHAR	6	
ZEIEH	Unit for the performance of maintenance tasks	DZEIEH	UNIT	3	T006
LEIEH	Maintenance strategy activity unit	LEIEH	UNIT	3	T006
ZYK1G	Total cycle duration of a maintenance strategy	DZYK1_GES	FLTP	16(16)	
ZYK2G	Total cycle duration of a maintenance strategy	DZYK2_GES	FLTP	16(16)	
ZYK1K	Smallest step between two maintenance packages	DZYK1_GGT	FLTP	16(16)	
ZYK2K	Smallest step between two performance-based MaintPackages	DZYK2_GGT	FLTP	16(16)	
ANZP1	Number of MaintPackages in the MaintStrategy	ANZPAK1	NUMC	2	
ANZP2	Number of MaintPackages in the MaintStrategy (peformance)	ANZPAK2	NUMC	2	
TOLE1	Tolerance for Late Completion (%)	TOLERPLUS	NUMC	3	
TOLE2	Tolerance for Early Completion (%)	TOLERNEG	NUMC	3	
VERS1	Shift Factor for Late Completion	VERSCHPLUS	NUMC	3	
VERS2	Shift Factor for Early Completion	VERSCHNEG	NUMC	3	
TERMK	Scheduling indicator	TERMKNZ	CHAR	1	
HORIZ	Call horizon for maintenance plan calls	HORIZONT	NUMC	3	
HORIZ_QUALIFIER	Calculation Type of Call Horizon	EAM_CC_CALLHORIZON_QUALIFIER	CHAR	1	
FABKL	Factory calendar key	FABKL	CHAR	2	TFACD

Figura 3.32: Tabella Maintenance strategy data

PLWP

Questa struttura dati viene descritta come "*Allocation of maintenance packages to task list operations data*"; come accennato in precedenza SAP tramite l'uso delle strategie e, in particolare, dei pacchetti, vuole permettere di associare alle attività manutentive delle frequenze

di esecuzione; la tabella PLWP è la struttura dati che materialmente rappresenta questo collegamento.

All'interno di questa tabella possono essere trovati una serie di campi che permettono di realizzare questa relazione, la quale è di tipo molti a molti; ciò significa che potranno essere associati più package per ogni operazione (si consideri l'esempio fatto nel caso in cui si segue un approccio ibrido time/performance based) e, all'inverso, più operazioni per uno stesso package (evitando, così, la ridondanza di specificare una stessa frequenza più volte).

Gli attributi che permettono di realizzare questo collegamento sono i seguenti:

- *Task list type, Task list group, Group counter, Number of the task list node*: sono attributi che permettono di capire l'operazione a cui si sta facendo riferimento. Un'operazione, come visto nei paragrafi precedenti, è identificata da una task list di appartenenza e una posizione interna al ciclo.
- *Maintenance strategy, Maintenance package number*: questi altri due attributi permettono di specificare il pacchetto (frequenza) a cui la relazione fa riferimento; specificando una strategia (insieme di pacchetti) e un contatore interno ad essa si identifica in maniera univoca un determinato package.

A completamento della descrizione di PLWP, come fatto per tutte le tabelle precedenti, viene riportata un'illustrazione della sua struttura interna (Figura 3.33).

3.6.3 Measuring Point

Terminata la trattazione degli aspetti riguardanti la frequenza delle operazioni in SAP, si passerà ora ad illustrate il concetto di *Measuring Point*; quest'ultimo è un elemento chiave del maintenance register e viene utilizzato per registrare/monitorare le misurazioni relative a determinati asset o sedi tecniche all'interno di un impianto. I punti di misurazione sono spesso associati a componenti o attrezzature specifiche e vengono utilizzati per raccogliere dati operativi importanti. Queste letture, nella maggior parte dei casi, servono per capire quando devono essere svolti degli interventi di manutenzione; se si pensa a quanto detto nella sezione precedente riguardo l'approccio performance based, è facile capire come siano proprio i measuring point a fornire i valori che poi verranno usati per capire se si deve effettuare un intervento o meno. In SAP il concetto di measurement point viene rappresentato dalla tabella IMPTT.

IMPTT

All'interno di questa struttura dati, che viene definita "Measuring Point data", si può osservare una lunga lista di campi atti a fornire una rappresentazione completa del concetto di punto di misura.

Poco c'è da dire riguardo l'unico attributo chiave, chiamato *POINT - Measuring Point*, che consiste in un classico identificatore alfanumerico di 12 caratteri. Al contrario, la tabella presenta un gran numero di campi descrittivi i quali vengono usati per fornire informazioni riguardanti vari aspetti del concetto di measuring point; senza entrare nello specifico, questi dati possono riguardare, ad esempio:

- l'asset che rappresenta l'oggetto fisico in grado di misurare la grandezza in questione;
- la categoria e ambito applicativo del punto di misura;
- la descrizione del measuring point;

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
PLNTY	Task List Type	PLNTY	CHAR	1	TCA01
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
PLNAL	Group Counter	PLNAL	CHAR	2	
PLNFL	Sequence	PLNFOLGE	CHAR	6	
PLNKN	Number of the task list node	PLNKN	NUMC	8	
PAKET	Maintenance Package Number	PAKETZAEHL	NUMC	2	Assigned to domain
ZAEHL	Internal counter	CIM_COUNT	NUMC	8	
DATUV	Valid-From Date	DATUV	DATS	8	
TECHV	Technical status from	TECHV	CHAR	12	
STRAT	Maintenance strategy	STRAT	CHAR	6	Assigned to domain
IWPOK	Maintenance package indicator required	IFLG_WPOK	CHAR	1	
FLGWP	Ind.: Allocation maintained through maint.plan	IFLG_WPLAN	CHAR	1	
LOEKZ	Deletion Indicator	LKENZ	CHAR	1	
PARKZ	Indicator: inactive changes	PARKZ	CHAR	1	
AENNR	Change Number	AENNR	CHAR	12	AENR
ANDAT	Date record created on	ANDAT	DATS	8	
ANNAM	User who created record	ANNAM	CHAR	12	
AEDAT	Changed On	AEDAT	DATS	8	
AENAM	Name of Person Who Changed Object	AENAM	CHAR	12	

Figura 3.33: Tabella Allocation of maintenance packages to task list operations data

- la grandezza misurata;
- il formato dell'output fornito.

Come accennato in precedenza, per ragioni di brevità non si andrà a descrivere nel dettaglio ognuno di questi campi, ma, in questo caso, ci si limiterà a fornire una rappresentazione grafica della struttura interna alla tabella, osservabile in Figura 3.34.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
POINT	Measuring Point	IMRC_POINT	CHAR	12	
MPOBJ	Object Number for the Measuring Point Object	IMRC_MPOBJ	CHAR	22	ONR00
PSORT	Position Number of the Object Measuring Point (Sort Field)	IMRC_PSORT	CHAR	20	
PSORTR	Indicator: Position Number Comes From Reference MeasPoint	IMRCRPSORT	CHAR	1	
PTTXT	Description of Measuring Point	IMRC_PTTXT	CHAR	40	
MLANG	Language Key	SPRAS	LANG	1	T002
KZLTX	Indicator: Long text exists	ILOM_KZLTX	CHAR	1	
MPTYP	Measuring Point Category	IMRC_MPTYP	CHAR	1	T370P
IRFMP	Indicator: MeasPoint Is a Reference Measuring Point	IMRC_IRFMP	CHAR	1	
ERDAT	Date on which the object was created	ICRDT	DATS	8	
ERNAM	Name of User Who Created the Object	ICRNA	CHAR	12	
AEDAT	Date on which the object was last changed	IUPDT	DATS	8	
AENAM	Name of the user who last changed the object	IUPNA	CHAR	12	
BEGRU	Technical object authorization group	IAUTG	CHAR	4	T370B
INACT	Indicator: Measuring Point Inactive	IMRC_INACT	CHAR	1	
LVORM	Deletion Flag for 2-Level Deletion Logic	IMRC_DEL2	CHAR	1	
LOCAS	Assembly for Localization of Measuring Point	IMRC_LOCAS	CHAR	18	MARA
REFMP	Reference Meas.Pt	IMRC_REFMP	CHAR	12	IMPTT
ATINN	Internal characteristic	ATINN	NUMC	10	CABN
ATINNR	Indicator: Charac Comes From Reference Measuring Point	IMRCRATINN	CHAR	1	
EXPON	Power of Ten Exponent for Floating Point Display	IMRC_EXPON	INT2	5	
DECIM	Number of Decimal Places for Number Display	IMRC_DECIM	INT2	5	

DESIR	Measuring Point Target Value	IMRC_DESIR	FLTP	16(16)	
DESIRI	Indicator: Associated Number Field Contains a Value	IVALU	CHAR	1	
DESIRR	Indicator: Target Value Comes From Reference Measuring Point	IMRCRDESIR	CHAR	1	
DSTXT	Aditional Short Text for Measuring Point	IMRC_DSTXT	CHAR	40	
DSTXTR	Indicator: Additional Text Comes From Reference MeasPoint	IMRCRDSTXT	CHAR	1	
MRMIN	Lower Measurement Range Limit/Minimum Total Counter Reading	IMRC_MRMIN	FLTP	16(16)	
MRMINI	Indicator: Associated Number Field Contains a Value	IVALU	CHAR	1	
MRMAX	Upper Measurement Range Limit/Maximum Total Counter Reading	IMRC_MRMAX	FLTP	16(16)	
MRMAXI	Indicator: Associated Number Field Contains a Value	IVALU	CHAR	1	
MRNGU	Measurement Range Unit	IMRC_MRNGU	UNIT	3	T006
INDCT	Indicator to Show that Measuring Point Is a Counter	IMRC_INDCT	CHAR	1	
INDRV	Indicator: Counter Runs Backwards	IMRC_INDRV	CHAR	1	
INDTR	Indicator That Measurement Reading Transfer is Supported	IMRC_INDTR	CHAR	1	
TRANS	Measuring Point from Which Meas. Reading Was Transferred	IMRC_TRANS	CHAR	12	IMPTT
CJUMP	Counter Overflow Reading in SI Unit	IMRC_CJUMP	FLTP	16(16)	
CJUMPI	Indicator: Associated Number Field Contains a Value	IVALU	CHAR	1	
PYEAR	Annual Estimate in SI Unit	IMRC_PYEAR	FLTP	16(16)	
PYEARI	Indicator: Associated Number Field Contains a Value	IVALU	CHAR	1	
CODCT	Catalog Type - Measurement Reading Valuation Code	IMRC_CODCT	CHAR	1	TQ15
CODGR	Code Group - Measurement Reading Valuation Code	IMRC_CODGR	CHAR	8	QPGR
CODGRR	Indicator: Code Group Comes From Reference Measuring Point	IMRCRCODGR	CHAR	1	
CDSUF	Indicator: Valuation Code Sufficient for Measurmnt. Document	IMRC_CDSUF	CHAR	1	
MODTR	Mode of Counter Reading Transfer	IMRC_MODTR	CHAR	1	
INDTRR	Indicator: Measurement Transfer Comes from Ref. Meas. Point	IMRCRINDTR	CHAR	1	

Figura 3.34: Tabella Measuring Point data

3.6.4 Maintenance Plan

Fino ad ora sono stati approfonditi dei concetti che, seppur di fondamentale importanza, costituiscono il "contorno" del maintenance register. Si passerà, ora, a trattare dei concetti che, invece, formano il core di questo registro, rappresentando, anche, il punto di giunzione con l'asset register.

Il primo degli elementi che verrà approfondito è il *Maintenance Plan*. In italiano chiamato "Piano di manutenzione", si tratta dell'oggetto che tiene insieme tutti i concetti visti fino ad ora nell'ambito delle attività manutentive. Di fatto, si sta parlando di un oggetto che fornisce una struttura organizzativa finalizzata alla gestione coesa e coerente di tutti gli altri elementi visti fino ad adesso.

Un maintenance plan, quindi, nella pratica, può essere definito come un piano in grado di tenere insieme:

- le operazioni di manutenzione da svolgere (concetti di task list e operations);
- le strategie/frequenze di ogni attività (concetti di strategy e package);
- gli asset su cui queste operazioni devono essere svolte (concetti di object list e posizione di manutenzione che verranno affrontati più avanti).

In SAP PM la tabella assegnata al concetto di maintenance plan è *MPLA*; di seguito si approfondirà la struttura di tale tabella.

MPLA

Ponendo il focus sulla struttura di questa tabella, questa viene descritta come "Maintenance plan data" e contiene informazioni generali sul piano di manutenzione; molti dei dati in essa contenuti sono informazioni già viste anche in altre tabelle, ma, inserite in questa tabella, assumono la valenza di valori di default da considerare nell'associazione con gli altri concetti.

Si procederà, adesso, con un approfondimento dei campi presenti nella tabella, osservabili anche nell'illustrazione riportata (Figura 3.35).

Come è possibile osservare, è presente un unico campo chiave, chiamato *WARPL*, che assume la forma di un codice alfanumerico di 12 caratteri.

Per quanto riguarda i campi descrittivi, come già accennato, abbiamo dei campi ridondanti già presenti in altre tabelle, ma che, in questo caso, devono essere intesi, se inseriti, come valori di default. Tuttavia, in altri casi, è anche possibile che questi valori siano semplicemente delle ridondanze di campi presenti in altre tabelle, riportati in MPLA per scopi legati all'efficienza delle operazioni di read.

Ad esempio, sono presenti due attributi chiamati *EQUNR* e *TPLNR*, i quali identificano, rispettivamente, l'equipment e la functional location di riferimento; di conseguenza questi campi, se "valorizzati", rappresentano gli asset che per default vengono assegnati alle operazioni legate al maintenance plan.

Per quanto riguarda i campi introdotti dalla tabella, e quindi non presenti altrove, questi sono tutti riconducibili alla definizione del periodo di scheduling del progetto. La funzione principale che un piano di manutenzione svolge, è quella di definire l'orizzionte temporale all'interno del quale dovrà avvenire lo scheduling delle attività manutentive. Se, ad esempio, si definisce uno scheduling period di 3 anni, tutta la pianificazione delle operazioni avverà all'interno del periodo definito e non oltre.

A tal fine sono presenti i seguenti attributi:

- *WPTXT - Maintenance Plan Text*: campo descrittivo breve in grado di fornire una panoramica generale sul piano.

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
WARPL	Maintenance Plan	WARPL	CHAR	12	
WPTXT	Maintenance Plan Text	WPTXT	CHAR	40	
STRAT	Maintenance strategy	STRAT	CHAR	6	T351
ABRHO	Scheduling Period	ABRHO	NUMC	3	
EQUNR	Equipment Number	EQUNR	CHAR	18	Assigned to domain
TPLNR	Functional Location	TPLNR	CHAR	30	Assigned to domain
VSPOS	Shift Factor for Late Completion	VERSCHPLUS	NUMC	3	
VSNEG	Shift Factor for Early Completion	VERSCHNEG	NUMC	3	
TOPOS	Tolerance for Late Completion (%)	TOLERPLUS	NUMC	3	
TONEG	Tolerance for Early Completion (%)	TOLERNEG	NUMC	3	
SFAKT	Cycle modification factor	SFAKT	DEC	3(2)	
STICH	Scheduling indicator	TERMKNZ	CHAR	1	
TGOON	Scheduling to reoccur when lead float reached	TGON	NUMC	3	
HORIZ	Call horizon for maintenance plan calls	HORIZONT	NUMC	3	
HORIZ_DAYS	Call horizon for maintenance plan calls	HORIZONT	NUMC	3	
MPTYP	Maintenance plan category	MPTYP	CHAR	2	T399W
HUNIT	Unit in scheduling interval	HUNIT	UNIT	3	Assigned to domain
START_DATE	Start Date for Scheduling	DATE_START	DATS	8	
START_TIME	Start Time for Scheduling	TIME_START	TIMS	6	

Figura 3.35: Tabella Maintenance plan data

- *STRAT - Maintenance Strategy*: strategia di manutenzione su cui il maintenance plan è basato; come detto nelle sezioni precedenti, serve per stabilire le frequenze delle operazioni ed, in alcuni casi, lo scheduling period stesso.
- *ABRHO - Scheduling period*: questo è il parametro più importante che, tramite l'uso di un valore numerico, stabilisce la durata del periodo di scheduling.
- *HUNIT - Unit in scheduling interval*: a completare l'informazione dello scheduling period si ha la necessità di specificare una unità di misura per il valore espresso in ABRHO (ed esempio, mesi, anni, settimane oppure nel caso di performance based plans, temperatura, running hours, pressione etc.).
- *START_DATE/TIME - Start Date/Time for scheduling*: si tratta di attributi che, nel caso di piano time/calendar based, definiscono la data e l'orario a partire dai quali dovrà iniziare lo scheduling period. Nella tabella è anche presente un campo (non riportato nella figura), chiamato SZAEH - Start counter reading, che permette di definire il valore iniziale in caso di maintenance plan basati su performance.
- *SFAKT - Cycle modification factor*: potrebbe capitare, in alcuni casi, che il periodo di scheduling sia definito all'interno della strategia di manutenzione; in questo scenario, siccome il piano fa riferimento ad una strategia da cui eredita questa durata, si ha la necessità di avere un parametro di controllo in grado di agire sulla durata presa dalla strategy. Quindi questo attributo serve a calcolare lo scheduling period del maintenance plan a partire dal periodo preso dalla strategia. Se si assume una strategia con scheduling period di 60 giorni, la quale viene utilizzata in un piano con Cycle modification factor di 1.5, allora la durata dello scheduling period del piano in questione sarà 60*1.5, e quindi 90 giorni.
- *STICH - Scheduling indicator*: questo attributo serve a definire la tipologia di scheduling legata al piano. In particolare, in base al valore assegnato a questo campo, si potranno avere diversi approcci:
  - time-based scheduling;
  - scheduling based on a key date;
  - scheduling by factory calendar;
  - performance-based scheduling.

3.6.5 Maintenance Item e Object List

A completare quello che è stato definito il "core" della parte maintenance di SAP PM c'é il concetto di *Maintenance Item*. In italiano viene chiamata "Posizione di manutenzione" e viene definita come l'applicazione di una task list ad un determinato insieme di asset.

Le posizioni di manutenzione sono definite nel contesto di un piano di manutenzione (al quale, quindi, fanno riferimento), e quindi fungono a tutti gli effetti, come punto di collegamento fra tutti gli elementi visti finora (piano di manutenzione, operazioni, asset)

L'oggetto che, per ogni maintenance item, permette di definire gli elementi dell'asset register associati, è chiamato *Object list*. Questa lista ha il solo scopo di associare ad un insieme di attività (specificate tramite la task list associato al maintenance item stesso) un preciso asset (equipment, functional location o assembly). Di seguito viene riportato un esempio di una object list riscontrabile nel lavoro di tutti i giorni in Pansoinco (Figura 3.36).

Entrando, ora, nello specifico, si andrà a vedere, come fatto per tutti i casi precedenti, la rappresentazione fornita da SAP di questi due concetti. In particolare, le tabelle a cui si sta facendo riferimento sono *MPOS*, *OBJK* e *SER00*

	A	B	C	D
1		**Maintenance Item**	**Equipment**	
2		12	10	
3		WAPOS	EQUNR	
4	**PDG**	200260000001	2130010311	
5	**PDG**	200260000001	2130010312	
6	**PDG**	200260000001	2130010315	
7	**PDG**	200260000002	2130010316	
8	**PDG**	200260000002	2130010335	
9	**PDG**	200260000002	2130010336	
10	**PDG**	200260000003	2130010317	
11	**PDG**	200260000003	2130010318	
12	**PDG**	200260000003	2130010319	
13	**PDG**	200260000003	2130010322	
14	**PDG**	200260000003	2130010323	
15	**PDG**	200260000003	2130010334	
16	**PDG**	200260000004	2130010320	
17	**PDG**	200260000004	2130010321	
18	**PDG**	200260000004	2130010324	
19	**PDG**	200260000004	2130010325	
20	**PDG**	200260000004	2130010326	
21	**PDG**	200260000004	2130010327	
22	**PDG**	200260000004	2130010328	
23	**PDG**	200260000004	2130010329	
24	**PDG**	200260000004	2130010332	
25	**PDG**	200260000005	2130010330	
26	**PDG**	200260000005	2130010331	

Figura 3.36: Esempio di object list per equipment

MPOS

Cominciando l'approfondimento dalla tabella MPOS, questa viene identificata come "Maintenance item data" ed, essendo l'unica tabella finalizzata alla rappresentazione di questo concetto, contiene tutte le informazioni relative alla posizione di manutenzione. Nel caso di questa struttura dati sarà interessante osservare i punti di collegamento verso le altre entità presenti nella base di dati.

Si andrà, ora, ad approfondire i campi di maggior rilievo presenti nella tabella, seguendo il solito procedimento. A tal fine si riporta un'illustrazione dei campi più importanti di MPOS (Figura 3.37).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
WAPOS	Maintenance item	WAPOS	CHAR	16	
WARPL	Maintenance Plan	WARPL	CHAR	12	MPLA
WPPOS	Item in the maintenance plan	WPPOS	NUMC	4	
PSTXT	Item Short Text	POSTXT	CHAR	40	
EQUNR	Equipment Number	EQUNR	CHAR	18	EQUI
OBKNR	Object list number	OBJKNR	INT4	10	Assigned to domain
PLNTY	Task List Type	PLNTY	CHAR	1	TCA01
PLNNR	Key for Task List Group	PLNNR	CHAR	8	
PLNAL	Group Counter	PLNAL	CHAR	2	
STATUS	Maintenance Item Status	MPSTAT	CHAR	1	
WPGRP	Planner Group for Customer Service and Plant Maintenance	INGRP	CHAR	3	T024I
IWERK	Maintenance Planning Plant	IWERK	CHAR	4	T399I
INACT	Indicator that maintenance item is inactive	INACT	CHAR	1	
ILART	Maintenance activity type	ILA	CHAR	3	T353I
GSBER	Business Area	GSBER	CHAR	4	TGSB
PRIOK	Priority	PRIOK	CHAR	1	T356

Figura 3.37: Tabella Maintenance item data

Per quanto riguarda i campi identificativi, oltre al solito Client è presente un unico identificatore alfanumerico a 16 caratteri chiamato *WAPOS - Maintenance Item*.

Molto più c'è da dire in merito ai campi descrittivi, dei quali nell'immagine viene riportato solo un sottoinsieme per ragioni di brevità. Quindi, a questo punto, si illustrerà il significato di ognuno di questi campi.

- *WARPL - Maintenance Plan*: come accennato in precedenza, una posizione di manutenzione è definita nell'ambito di un maintenance plan; questo attributo, tramite un riferimento alla tabella MPLA (discussa in precedenza), permette di realizzare questo collegamento. Specificando il piano di riferimento in questo modo, la relazione che si viene a creare è di tipo uno a molti, in quanto ad uno stesso piano potranno essere associate più posizioni di manutenzione. Questo approccio segue giustamente la logica operativa, in quanto si ha sempre la necessità di definire più insiemi di operazioni (task list) per uno stesso piano sia di associare tali attività a più elementi dell'asset register.

Tuttavia, come si vedrà nel seguito di questa tesi, in alcuni casi, come per lo Standard ENI, questa relazione fra piano e posizione viene interpretata come una relazione uno ad uno.

- *WPPOS - Item in the maintenance plan*: identifica il numero della posizione di manutenzione all'interno del maintenance plan di riferimento. Ad esempio M.Item numero: 2 , del piano: MP10.
- *PSTXT - Item Short Text*: descrizione breve esplicativa della funzione associata al maintenance item in questione.
- *OBKNR - Object List number*: si è di fronte ad un altro collegamento esterno; questa volta si fa riferimento all'eventuale object list collegata alla posizione in questione. La tabella a cui si fa riferimento è la SER00.
- *EQUNR - Equipment number*: sono presenti nella tabella (non riportati in figura) anche dei riferimenti a functional location e assembly; questi collegamenti sono da intendere come dei riferimenti di default. Nel caso in cui non dovesse essere presente un collegamento ad una object list, questi attributi permetterebbero, comunque, di risalire agli elementi dell'asset register a cui applicare le operazioni di manutenzione.
- *PLNTY - PLNNR - PLNAL*: ancora dei riferimenti esterni che, in questo caso, servono per specificare l'insieme di operazioni associate alla posizione di manutenzione; per realizzare questo collegamento si fa riferimento alle entità legate al concetto di task list (e quindi tipo, gruppo e contatore interno al gruppo). In questo modo sarà possibile risalire alle attività che dovranno essere applicate ad un determinato insieme di asset.
- *WPGRP e IWERK*: questi attributi servono per specificare, nel caso ce ne dovesse essere il bisogno, un impianto in cui dovranno essere svolte le attività manutentive e un gruppo responsabile della loro corretta esecuzione.
- *INACT - Inactive Indicator*: si tratta di un indicatore che indica se la posizione di manutenzione è attiva o meno, e quindi se dovranno essere generati ordini di manutenzione congruenti con le attività correlate.
- *ILART - Maintenance activity type*: attributo che indica il tipo generico della posizione; questo campo potrebbe indicare l'ambito di applicazione o lo scopo finale dell'esecuzione delle attività di manutenzione correlate.
- *PRIOK - Priority*: valore di priorità assegnato all'esecuzione delle attività relative al maintenance item in questione; come accennato in altre occasioni, solitamente valori più bassi indicano un livello di priorità maggiore, ma non si tratta di una regola fissa.

SER00 e OBJK

Queste strutture dati, come accennato in precedenza, rappresentano un legame vero e proprio fra il maintenance e l'asset register; questo perché, come già detto, legano delle posizioni di manutenzione (quindi delle attività) a degli elementi dell'asset register (quindi equipment, functional location o assemblies). La tabella *SER00*, descritta come "General Header Table for Serial Number Management data", rappresenta il concetto di lista vera e propria, all'interno della quale si possono trovare le varie voci (singoli legami con gli object). Di seguito viene riportata un'illustrazione della tabella SER00 e dei suoi campi principali (Figura 3.38).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
OBKNR	Object list number	OBJKNR	INT4	10	

Figura 3.38: Tabella General Header Table for Serial Number Management data

Come è possibile osservare, la tabella presenta un unico campo chiave, il quale viene utilizzato per identificare univocamente l'oggetto "lista di asset".

A questa lista fa riferimento la tabella *OBJK*, definita come "Plant Maintenance Object List data", la quale rappresenta la singola voce della object list. Di conseguenza, al suo interno sarà possibile trovare tutta una serie di elementi che fanno riferimento ai possibili asset referenziabili; in particolare, si avranno dei riferimenti alle tabelle viste durante l'approfondimento dedicato all'asset register, quindi equipment, sede tecnica, assembly material etc.

Essendo definita sulla base di una lista complessiva, i campi chiave che contraddistinguono OBJK sono due: si ha un riferimento alla tabella SER00, il quale identifica la lista di appartenenza, in aggiunta ad un contatore interno all'elenco (ogni voce sarà, quindi, riconoscibile dalla coppia di valori lista e posizione nella lista). A completamento di questo approfondimento, viene riportata una rappresentazione grafica della tabella OBJK (Figura 3.39).

Field	Description	Data Element	Data Type	length (Dec)	Check table
MANDT	Client	MANDT	CLNT	3	T000
OBKNR	Object list number	OBJKNR	INT4	10	SER00
OBZAE	Object list counters	OBJZA	INT4	10	
EQUNR	Equipment Number	EQUNR	CHAR	18	EQUI
BAUTL	Assembly	BAUTL	CHAR	18	MARA
ILOAN	Location and account assignment for technical object	ILOAN	CHAR	12	ILOA
SORTF	Object list sort field	OBSORT	CHAR	20	
BEARB	Processing indicator	BEARBZS	CHAR	1	
OBJVW	Object list usage	OBJVW	CHAR	1	
SERNR	Serial Number	GERNR	CHAR	18	
MATNR	Material Number	MATNR	CHAR	18	MARA
UII	Unique Item Identifier	UII_CHAR72	CHAR	72	

Figura 3.39: Tabella Plant Maintenance Object List data

CAPITOLO 4

Sviluppo del Database

In questo quarto capitolo della tesi, verranno approfonditi l'ambito e le scelte fatte relativamente alla fase di sviluppo del database centrale. In un primo momento, quindi, il focus sarà posto sull'infrastruttura aziendale e sui tool software che hanno fatto da sfondo a questa fase del progetto. Successivamente, ci si concentrerà sulle scelte fatte per arrivare alla definizione del database. Verranno, quindi, illustrati i modelli ER e relazionale prodotti da questo processo; in conclusione, si approfondirà, in maniera più mirata, l'implementazione di alcune soluzioni di maggior interesse.

4.1 Introduzione allo sviluppo

In questo capitolo si illustreranno le fasi di progettazione e sviluppo che hanno portato alla definizione della struttura dati utilizzata come base di partenza per questo progetto; come accennato precedentemente, questa infrastruttura è fortemente basata sui concetti e sulle modalità di rappresentazione adottate da SAP PM per la gestione dei dati di manutenzione.

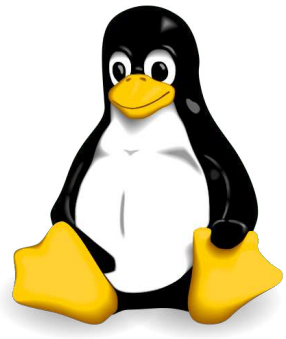
Infatti, nei capitoli introduttivi della tesi, era stato detto che si sarebbe ottenuta la base di dati finale operando sia dei tagli che delle aggiunte alla mappatura di SAP. Questo perché molte delle entità presenti in quest'ultima non risultano particolarmente rilevanti per il fine di questo progetto; si pensi, ad esempio, a tutta la parte che SAP riserva alla gestione dei costi, delle responsabilità e della documentazione. La rappresentazione di tali concetti, come verrà spiegato, sarà semplicemente ridotta all'uso di tabelle di validazione del tipo key-value (in questo modo verranno trascurate tutte le proprietà poco inerenti all'ambito operativo aziendale, mantenedo, però, la capacità di memorizzare elenchi di validazione associati a tali concetti).

Detto ciò, ora si illustreranno sia l'ambiente che l'infrastruttura tecnologica che hanno rappresentato il contesto di svolgimento delle fasi relative alla progettazione e allo sviluppo.

4.1.1 Infrastruttura tecnologica

Server Aziendale

La prima parte dell'approfondimento sarà dedicato a definire il contesto e l'ambiente in cui sono state svolte tutte le operazioni relative allo sviluppo. Si descriverà quindi questa infrastruttura a livello di caratteristiche tecniche generali senza scendere troppo in dettagli di scarsa rilevanza.



Per lo svolgimento di questo progetto, l'azienda ha messo a disposizione il suo server Linux interno; quest'ultimo offre prestazioni potenti, affidabilità, flessibilità, scalabilità ed economicità. Queste sono caratteristiche comuni ai server Linux i quali sono diventati la soluzione preferita per molte aziende e individui, anche grazie alla loro vasta gamma di funzionalità e applicazioni. Un server Linux è adatto a soddisfare le esigenze di aziende di tutte le dimensioni e, con il panorama tecnologico in continua evoluzione, è certo che questa tecnologia continuerà a fornire soluzioni innovative nel lungo periodo.

Di fatto, si tratta di un sistema operativo utilizzato per gestire in modo sicuro e affidabile una rete di computer e altri dispositivi, comunemente in un ambiente aziendale. I server Linux sono popolari grazie alla loro natura open source e alle elevate prestazioni, consentendo agli utenti di personalizzare il sistema secondo le proprie esigenze. I vantaggi dell'utilizzo di un server Linux includono un aumento dell'efficienza, scalabilità, risparmi sui costi, stabilità, sicurezza e una vasta gamma di applicazioni software disponibili. I server Linux sono, anche, noti per la loro base sicura e affidabile, che li rende la scelta ideale per ambienti critici come quelli governativi o finanziari.

Una rete integrata di diversi server Linux offre una piattaforma ideale per le aziende che richiedono ridondanza e alta disponibilità per i loro dati critici. La sicurezza è una considerazione chiave nella scelta di un server Linux, e diverse funzionalità di sicurezza, tra cui crittografia, autenticazione, controllo degli accessi e strumenti di gestione degli utenti, proteggono i dati memorizzati sul server.

SSH Tunnel

Per eseguire l'accesso a questo server, messo a disposizione dall'azienda, non è possibile procedere in maniera diretta, ma è necessario passare da un *Tunnel SSH*.

Lo SSH tunneling è una tecnica avanzata e potente che sfrutta il protocollo SSH (Secure Shell) per creare un canale sicuro tra due sistemi, consentendo la trasmissione di dati attraverso questa connessione cifrata. Questo meccanismo è ampiamente utilizzato per aumentare la sicurezza delle comunicazioni su reti non sicure, fornendo un'infrastruttura crittografica robusta e affidabile.

Di base, il tunneling SSH opera attraverso la creazione di un canale sicuro attraverso il quale i dati possono essere trasmessi in modo sicuro tra un client e un server. In termini semplici, SSH tunneling consente di instradare il traffico di rete attraverso una connessione sicura SSH, creando un "tunnel" crittografato tra le due estremità della comunicazione. Ciò può essere particolarmente utile quando si desidera garantire la sicurezza delle informazioni sensibili trasmesse su reti non affidabili o potenzialmente non sicure. Nella Figura 4.1 è riportato uno schema illustrativo del funzionamento di base dell'SSH Tunneling.

Esistono diverse modalità di SSH tunneling, ognuna progettata per scopi specifici, queste sono:

- *Local Port Forwarding:* in questo tipo di tunneling, una porta sul client viene inoltrata attraverso la connessione SSH al server. Quando qualcuno si connette a questa porta sul server, il traffico viene instradato attraverso il tunnel e risulta come se provenisse dal client. Questo è utile per accedere a servizi remoti, come database o applicazioni web.

- *Remote Port Forwarding:* invertendo il processo del local port forwarding, questa modalità consente di inoltrare una porta sul server attraverso la connessione SSH al client. È

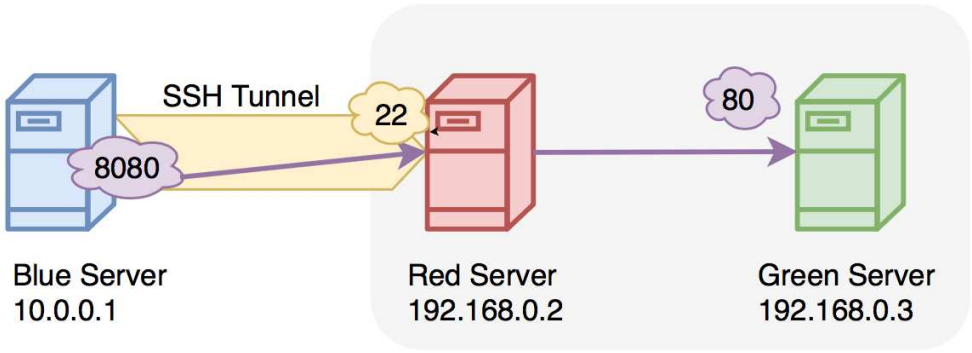


Figura 4.1: Schema di esempio di SSH Tunneling

utile quando è necessario rendere un servizio in esecuzione sul server accessibile da un client remoto.

- *Dynamic Port Forwarding (Dynamic SOCKS Proxy):* questa modalità crea una connessione dinamica che funge da proxy SOCKS. Tutte le richieste di rete inviate attraverso questo proxy vengono inoltrate mediante il tunnel SSH al server e poi alla destinazione finale. È un'opzione flessibile per instradare il traffico generale del web in modo sicuro attraverso una connessione SSH.

Si passerà, ora, a fornire una panoramica sulle applicazioni pratiche che, nella maggior parte dei casi, vedono come protagonisti i tunnel SSH:

- *Accesso Sicuro a Risorse Remote:* l'SSH tunneling è comunemente utilizzato per accedere in modo sicuro a risorse remote, come server o database, attraverso connessioni Internet non sicure. Ciò è particolarmente utile per gli amministratori di sistema che devono gestire server remoti in modo sicuro.
- *Navigazione Sicura su Reti Pubbliche:* quando ci si connette a reti pubbliche o Wi-Fi non sicure, l'SSH tunneling può essere utilizzato per proteggere la navigazione web e le comunicazioni online attraverso un tunnel crittografato, impedendo a potenziali attaccanti di intercettare i dati sensibili.
- *Bypassare Restrizioni di Rete:* in ambienti in cui alcune risorse sono bloccate o accessibili solo da determinate reti, SSH tunneling può essere impiegato per bypassare tali restrizioni instradando il traffico attraverso una connessione sicura SSH.

La crittografia integrata in SSH tunneling fornisce un elevato livello di sicurezza, rendendo difficile per gli aggressori intercettare o manipolare il traffico. Tuttavia, è essenziale utilizzare questo strumento con responsabilità e conformità alle leggi locali e aziendali per evitare usi impropri o illegali, come il superamento di restrizioni di sicurezza o la violazione della privacy. Si può dire, quindi, che l'SSH tunneling è una risorsa fondamentale per garantire la sicurezza delle comunicazioni attraverso reti non sicure, offrendo una solida base crittografica per una vasta gamma di applicazioni pratiche, soprattutto riguardanti gli ambiti aziendali. La sua flessibilità e la capacità di creare canali sicuri rendono questa tecnica uno strumento prezioso per chiunque cerchi di proteggere i dati sensibili durante la trasmissione su reti potenzialmente non sicure.

Nel caso del tunnel usato in Pansoinco per l'accesso sono richieste come credenziali un *utente* ed una *password*. Ciò avviene in quanto ad ogni utente registrato nel server Linux

saranno associati dei permessi di accesso a determinati progetti, cartelle e documenti. Per lo svolgimento di questo progetto sono state fornite dall'azienda le credenziali dell'account con permessi di accesso totali. Di seguito, in Figura 4.2, viene riportato un esempio di login al server tramite CLI.

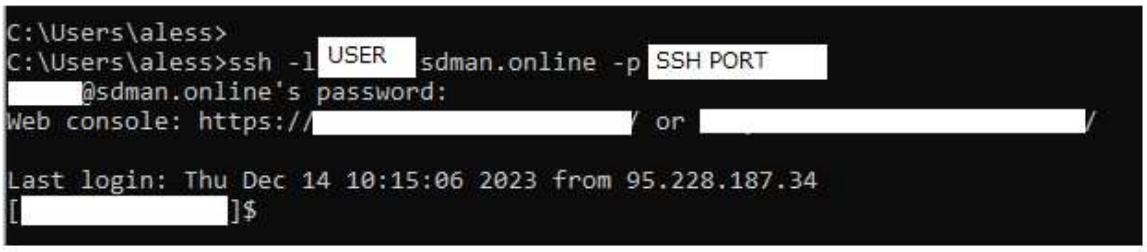


Figura 4.2: Accesso al server tramite SSH Tunneling

NAS - File Station

Oltre al server Linux descritto nella sezione precedente, per lo svolgimento di questo progetto, l'azienda ha anche messo a disposizione l'accesso al NAS aziendale.

Il *Network Attached Storage* (NAS) è una soluzione di archiviazione dedicata che offre un accesso centralizzato e condiviso ai dati attraverso una rete. Questa tecnologia si è evoluta come una risposta alle crescenti esigenze di archiviazione e condivisione dati nelle reti aziendali e domestiche. Di seguito, verranno approfonditi i principali aspetti del NAS, dalla sua architettura (Figura 4.3) al suo impatto sull'archiviazione dati.

Un NAS è costituito da hardware e software specializzati progettati per funzionare come un server di archiviazione dedicato. Tipicamente, il dispositivo è dotato di uno o più dischi rigidi, una connessione di rete e un sistema operativo incorporato. Quest'ultimo è progettato per gestire le operazioni di archiviazione e garantire l'accesso sicuro ai dati.

Per quanto riguarda le principali funzionalità del NAS messo a disposizione dall'azienda, esse sono:

- *Condivisione di File:* la funzione primaria di un NAS è consentire la condivisione di file all'interno di una rete. Agli utenti è consentito archiviare, accedere e condividere documenti, immagini, video e altri dati in modo centralizzato.
- *Accesso Remoto:* si ha la possibilità di accedere ai dati da remoto attraverso Internet. Questo è particolarmente utile per utenti che necessitano di recuperare file mentre sono fuori sede, come nel caso di questo progetto, svolto nella sede di Ancona.
- *Backup Automatici*: il NAS integra funzionalità di backup automatico, consentendo agli utenti di proteggere i propri dati da perdite accidentali o danneggiamenti dei file.
- *Servizi Multimediali*: alcuni NAS forniscono funzionalità avanzate, come la gestione di servizi multimediali. Essi possono fungere da server per lo streaming di contenuti audio e video all'interno della rete domestica. Nel caso del NAS utilizzato in Pansoinco, questa possibilità è garantita per immagini e video, ma non si hanno funzionalità di editing di documenti.
- *Accesso Autorizzato:* il NAS offre anche funzionalità di controllo degli accessi per garantire che solo utenti autorizzati possano accedere a determinati file o cartelle.

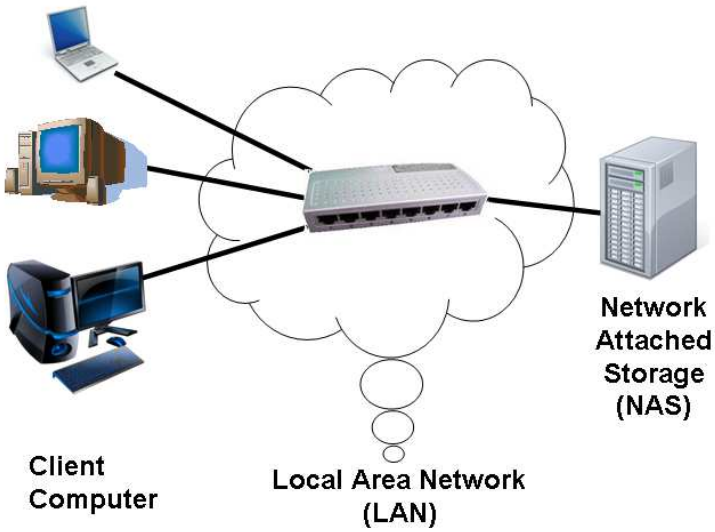


Figura 4.3: Esempio di architettura con NAS

Parlando, ora, dei vantaggi di cui si può beneficiare adottando come soluzione di storage quella che prevede l'uso di un NAS, i principali sono elencati di seguito:

- *Semplicità di Implementazione*: i NAS sono noti per la loro facilità di implementazione. Gli utenti non esperti possono configurare e gestire facilmente questi dispositivi senza avere necessariamente competenze tecniche avanzate.
- *Risparmio di Spazio*: rispetto a soluzioni di archiviazione tradizionali, i NAS occupano meno spazio fisico. Sono compatti e possono essere posizionati in modo discreto in ufficio o a casa.
- *Scalabilità*: il NAS consente una facile espansione dello spazio di archiviazione attraverso l'aggiunta di nuovi dischi rigidi o l'implementazione di unità esterne.
- *Accesso Condiviso*: i NAS facilitano la condivisione di risorse in reti di piccole e grandi dimensioni, promuovendo la collaborazione e semplificando la gestione dei dati.

Tuttavia, quando si usano soluzioni NAS, è di fondamentale importanza implementare misure di sicurezza robuste per proteggere i dati memorizzati. Ciò include l'uso di password robuste, la crittografia dei dati sensibili e l'applicazione di patch regolari per il sistema operativo del NAS.

L'accesso remoto dovrebbe essere configurato in modo sicuro, preferibilmente attraverso connessioni crittografate per prevenire eventuali rischi legati alle minacce online.

Il NAS è diventato, per l'azienda, un componente essenziale al fine di ottenere un'archiviazione dati centralizzata e condivisa. La sua versatilità, facilità di gestione e numerosi vantaggi lo rendono un elemento che, nel contesto aziendale, contribuisce, ogni giorno, in modo significativo, alla corretta gestione dell'infrastruttura generale di archiviazione dati.

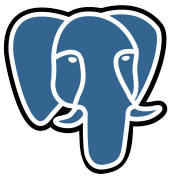
Il NAS fornito dall'azienda è stato di fondamentale importanza ai fini del progetto in quanto, al suo interno, sono stati immagazzinati tutti i dati storici aziendali. In esso è presente, quindi, un grandissimo numero di documenti, relativi allo svolgimento dei progetti che l'azienda ha portato a completamento negli ultimi 10 anni circa. Giusto al fine di dare una stima, nella cartella relativa allo storico progetti, sono presenti all'incirca un centinaio di

sottocartelle corrispondendi ad altrettanti progetti; come è facile capire, si tratta di una mole di dati notevole.

4.1.2 Strumenti utilizzati nello sviluppo

Passando, ora, ad approfondire i tool software che sono stati impiegati, all'interno dell'infrastruttura sopra illustrata, per portare avanti la progettazione e lo sviluppo della base di dati, si procederà elencando tali strumenti, fornendo per ognuno una rapida descrizione e le principali ragioni che hanno portato al loro impiego.

Postgres e PLpgsql



Si inizierà questo approfondimento, illustrando il DBMS scelto per lo sviluppo di questo progetto, ovvero *PostgreSQL*. Questa scelta, è stata di fatto guidata dall'azienda, la quale già ben prima dell'inizio di questo progetto faceva largo uso di questo tool. Infatti, all'interno del server Linux aziendale, è presente la base di dati contenente tutte le informazioni relative ai progetti in corso; questa struttura è rappresentata, appunto, da un database Postgres.

Si passerà, ora, a fornire una descrizione generale di cos'é PostgreSQL e delle regioni che lo hanno reso la sceltà più adatta nel contesto aziendale di Pansoinco.

Quando si parla di PostgreSQL ci si riferisce ad un potente sistema di gestione di database relazionali (RDBMS) open-source, il quale vanta una lunga storia di sviluppo continuo e di successo nella comunità del software libero. Conosciuto per la sua affidabilità, flessibilità e capacità di gestire compiti complessi, PostgreSQL è diventato una scelta preferita per organizzazioni di varie dimensioni. In questa panoramica, esploriamo la natura distintiva di PostgreSQL, comprese le sue caratteristiche chiave, la sua architettura e il suo impatto sul mondo della gestione dei database.

Le caratteristiche principali che contraddistinguono Postgres sono elencate di seguito:

- *Open-Source:* PostgreSQL è distribuito sotto la licenza PostgreSQL, una licenza open-source che promuove la libertà di utilizzo, modifica e distribuzione del software. Questa natura aperta favorisce l'innovazione e la collaborazione nella comunità degli sviluppatori.

- *Estensibilità:* una delle caratteristiche più distintive di PostgreSQL è la sua estensibilità. Gli utenti possono definire nuovi tipi di dati, operatori, funzioni e linguaggi procedurali, consentendo un elevato grado di personalizzazione e adattamento alle esigenze specifiche dell'applicazione.

- *ACID Compliance:* PostgreSQL è conforme agli standard ACID (Atomicità, Consistenza, Isolamento, Durata), garantendo la coerenza e l'integrità dei dati anche in situazioni di guasto del sistema o errori.

- *Supporto per JSON e Datatype:* oltre a supportare il modello relazionale tradizionale, PostgreSQL offre funzionalità avanzate per la gestione di dati semi-strutturati, inclusi JSON e array. Questa flessibilità è preziosa nell'era delle applicazioni web complesse.

- *Transazioni Multi-Version:* PostgreSQL utilizza un modello di transazioni multi-version (MVCC), consentendo la gestione efficiente della concorrenza nelle operazioni di scrittura senza compromettere l'integrità dei dati.

- *Gestione Ottimizzata delle Query:* il sistema dispone di un ottimizzatore di query sofisticato che analizza e ottimizza le query per massimizzare le prestazioni. Questo è particolarmente importante in ambienti in cui le prestazioni sono cruciali.

Per quanto riguarda l'architettura interna di PostgreSQL, questa è di tipo client-server; in essa i client si connettono al server attraverso connessioni TCP/IP (nel caso di questo progetto attraverso un SSH Tunnel). Questa configurazione consente l'accesso simultaneo da parte di più utenti e applicazioni.

Inoltre, PostgreSQL gestisce molteplici processi di backend per gestire le richieste dei client; ogni richiesta viene assegnata a un processo di backend dedicato, garantendo un'elevata concorrenza; i dati, in questo DBMS, vengono memorizzati in file di dati e indici, supportando varie tecniche di ottimizzazione delle prestazioni, tra cui la memorizzazione di dati in formato compresso e la gestione di diverse tipologie di indici.

Un ulteriore tool interno a Postgres utilizzato nelle fasi di sviluppo è stato il linguaggio *PLpgsql*. Acronimo di "Procedural Language/PostgreSQL Structured Query Language", è un linguaggio di programmazione procedurale specificamente progettato per PostgreSQL con lo scopo di estendere le capacità SQL standard di quest'ultimo consentendo la creazione di procedure, funzioni e trigger all'interno del database.

Il fatto che si tratta di un linguaggio di programmazione procedurale, implica che consente la scrittura di codice organizzato in procedure e funzioni. Ciò permette agli sviluppatori di definire logicamente passi sequenziali per eseguire operazioni complesse all'interno del database; è strettamente integrato con SQL, e chi lo utilizza è in grado di incorporare istruzioni SQL direttamente nel codice, consentendo una facile manipolazione e un facile recupero dei dati nel contesto delle procedure e delle funzioni.

Inoltre, il linguaggio supporta variabili, tipi di dati, cicli, condizioni e altre strutture di controllo tipiche dei linguaggi di programmazione procedurali, il che gli consente di gestire dati dinamici e creare algoritmi complessi. Infatti, come si accennava in precedenza, PLpgSQL è comunemente utilizzato per definire trigger e funzioni all'interno del database. I trigger possono essere attivati automaticamente in risposta a determinati eventi, mentre le funzioni possono essere chiamate esplicitamente per eseguire operazioni specifiche.

Questo linguaggio fornisce anche strutture ricorsive e di gestione degli errori; la creazione di funzioni iterative può essere utile per risolvere problemi complessi o iterare su strutture dati annidate.

SSH-FS

Al fine di avere un accesso più diretto al server Linux fornito dall'azienda, si è deciso di eseguire un *mounting* del file system remoto.

Per far ciò è stato sfruttato un tool software chiamato *SSHFS-Win Manager*, il quale consente di utilizzare il sistema SSHFS, nativo in Linux, in ambiente Windows.

SSHFS è un sistema di file virtuali basato su SSH, che consente di montare un sistema di file remoto sicuro su una directory locale. Esso utilizza il protocollo SSH per fornire un accesso sicuro ai file su una macchina remota come se fossero locali.

I principali vantaggi che hanno portato all'utilizzo di questo sistema sono i seguenti:

- *Montaggio Remoto:* SSHFS consente di montare una directory remota sul sistema locale, consentendo l'accesso ai file come se fossero memorizzati sulla macchina locale.

- *Accesso Sicuro:* poiché SSHFS si basa sul protocollo SSH (Secure Shell), tutti i dati trasmessi tra i dispositivi sono crittografati, garantendo un accesso sicuro e la protezione delle informazioni durante il trasferimento.

- *Configurazione Semplificata:* l'uso di SSHFS, di solito, non richiede configurazioni complesse. La maggior parte degli utenti deve solo avere un server SSH funzionante e le credenziali di accesso appropriate.
- *Interoperabilità:* SSHFS funziona su diverse piattaforme, inclusi sistemi Linux, macOS e Windows (utilizzando, come in questo caso, software aggiuntivo come WinFsp o appunto SSHFS-Win Manager). Ciò consente agli utenti di montare file system remoti indipendentemente dal sistema operativo in uso.
- *Comandi di Montaggio Semplici:* montare un sistema di file remoto tramite SSHFS di solito richiede solo un singolo comando, rendendo tale strumento accessibile anche agli utenti meno esperti. Per quanto riguarda la GUI del tool utilizzato per questo progetto, il montaggio risulta veramente immediato; una volta impostati i parametri di connessione (come IP/host, nome della connessione, password, etc.) l'operazione si svolge mediante un unico tasto.
- *Efficienza:* anche se SSHFS può introdurre una piccola latenza a causa dell'accesso remoto, è generalmente efficiente per l'uso quotidiano e può essere particolarmente utile per lavorare su file distribuiti su più server.
- *Aggiornamenti in Tempo Reale:* le modifiche apportate ai file sul lato remoto vengono riflesse in tempo reale sul lato locale, e viceversa, garantendo coerenza tra le due posizioni.

Nelle Figure 4.4 e 4.5, a scopo illustrativo, verranno riportate delle illustrazioni dell'interfaccia grafica del software utilizzato.

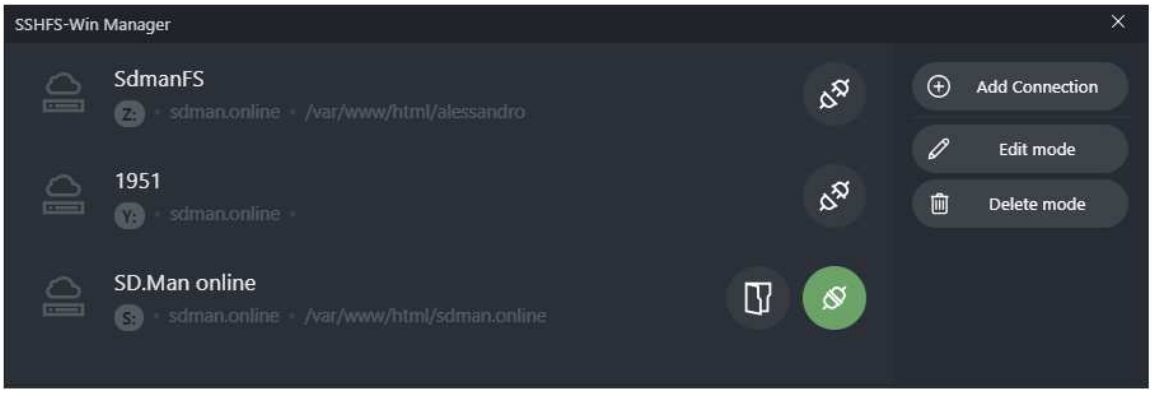


Figura 4.4: Schermata iniziale di SSHFS-Win Manager

DBeaver

Un ultimo tool che si ha la necessità di illustrare è *DBeaver*. Questo strumento è stato utilizzato per facilitare l'accesso, tramite SSH Tunneling, ai database presenti nel server Linux, senza dover ricorrere all'uso della CLI.

DBeaver può essere definito come un tool di amministrazione e sviluppo di database multi-piattaforma, open source e gratuito. Creato per soddisfare le esigenze degli sviluppatori, degli amministratori di database e degli analisti dati,

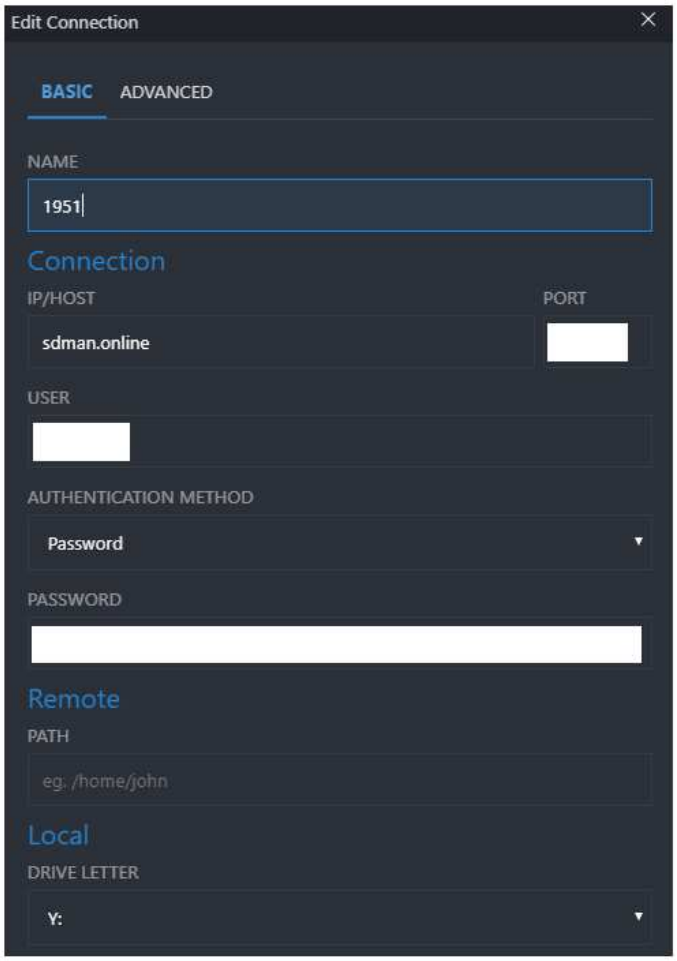


Figura 4.5: Parametri di connessione SSHFS

DBeaver offre un'interfaccia utente intuitiva, una vasta gamma di funzionalità e il supporto per numerosi database relazionali e NoSQL.

Alcuni degli aspetti più importanti che hanno portato al suo impiego sono i seguenti:

- *Multi-Piattaforma:* DBeaver è progettato per funzionare su diverse piattaforme, compresi Windows, Linux e macOS. Questa flessibilità consente agli utenti di lavorare su diverse macchine mantenendo una coerenza nell'esperienza utente.
- *Supporto per Diversi Database:* DBeaver supporta una vasta gamma di database relazionali e NoSQL, tra cui MySQL, PostgreSQL, Oracle, SQLite, MongoDB e molti altri. Ciò lo rende una scelta versatile per sviluppatori e amministratori che lavorano con diverse tecnologie di database.
- *Interfaccia Utente Intuitiva:* l'interfaccia utente di DBeaver è progettata per essere intuitiva e facile da navigare. Gli utenti possono esplorare i database, eseguire query SQL, modificare schemi e dati, tutto attraverso un'interfaccia coerente e ben organizzata. Di seguito, viene riportata un'illustrazione della GUI del software (Figura 4.6).

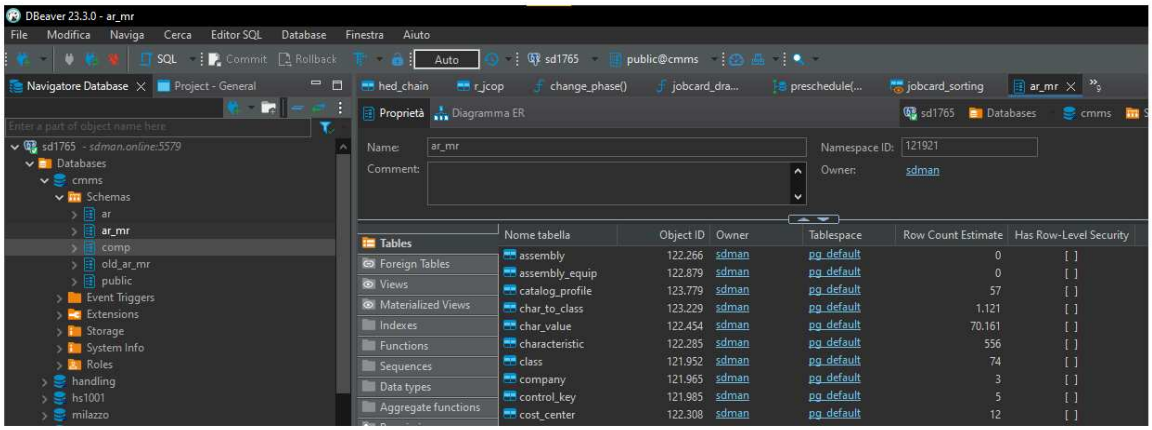


Figura 4.6: Interfaccia interna di DBeaver

- *Strumenti di Sviluppo SQL:* DBeaver offre potenti strumenti SQL per la creazione e l'esecuzione di query. Esso include funzionalità come l'autocompletamento, l'evidenziazione della sintassi, la formattazione del codice SQL e la visualizzazione dei risultati delle query in modo chiaro e dettagliato.
- *Gestione della Struttura del Database:* gli utenti possono gestire la struttura del database, compresi la creazione e la modifica di tabelle, viste, stored procedure e altri oggetti del database. Questo supporta il ciclo di vita completo dello sviluppo del database.
- *Importazione/Esportazione di Dati:* DBeaver semplifica l'importazione e l'esportazione di dati tra database. Gli utenti possono trasferire dati da e verso diverse origini e destinazioni con facilità, contribuendo a una maggiore efficienza nelle attività di gestione dei dati.
- *Gestione delle Connessioni:* DBeaver consente agli utenti di gestire facilmente le connessioni ai diversi database. I profili di connessione possono essere configurati e salvati, semplificando il processo di accesso a database diversi. Come accennato, nel caso di questo progetto, una feature molto utile è stata quella relativa alla gestione della connessione tramite SSH Tunnel.

- *Plugin e Estensibilità:* DBeaver supporta l'integrazione di plugin, consentendo agli utenti di personalizzare l'ambiente secondo le proprie esigenze. Questa flessibilità contribuisce a una maggiore adattabilità alle specifiche necessità di sviluppo e gestione.

È possibile intuire che DBeaver, quindi, si configura come un tool completo per lo sviluppo e la gestione dei database, offrendo un insieme di funzionalità robuste, un'interfaccia utente intuitiva e il supporto per una vasta gamma di database, rendendolo una scelta popolare nella comunità degli sviluppatori e degli amministratori di database.

4.2 Progettazione e sviluppo

In questa sezione, si passerà ad approfondire il risultato della fase di progettazione e sviluppo del database alla base dell'intero progetto; verranno analizzate le tabelle create e, per ognuna di esse, ci si concentrerà sugli apsetti più peculiari e sulle scelte effettuate in fase di modellazione.

Si partirà con una panoramica generale sullo schema ottenuto, riservando maggiore attenzione alle tabelle corrispondenti ai concetti di base visti nella sezione relativa al mapping di SAP; successivamente, verranno illustrate, in maniera più approfondita, alcune delle caratteristiche peculiari dello schema ottenuto.

Come già accennato in svariate occasioni, l'idea di base seguita per la progettazione di questa base dati, è stata quella di basarsi su un'ombra del sistema SAP; in altre parole, partendo dal mapping eseguito, si vogliono effettuare dei tagli finalizzati a non includere nello schema oggetti poco importanti agli scopi di gestione voluti dall'azienda. Ciò che si è fatto, è stato, di fatto, attaccare nei punti di taglio delle tabelle di validazione del tipo key-value. In questo modo, è stato possibile rimuovere molta della complessità introdotta dalla gestione di questi concetti, mantenendo la parte di validazione dei dati nelle tabelle principali (le quali si appoggiano su questi elementi secondari).

Nella Figura 4.7 viene illustrata la parte di interesse della mappatura SAP, quindi anche i "tagli effettuati" e le tabelle filtrate. La linea di taglio è visibile nell'immagine in rosso; le tabelle trascurate, o ricondotte a strutture key-value, si trovano nell'area in marrone chiaro; infine, le entità riconducibili a concetti più importanti sono nell'area verde.

Partendo, quindi, dalla mappatura "ritagliata" di SAP e, di conseguenza, basandosi sui concetti più rilevanti in essa presenti, si è iniziata a definire la struttura del database da sviluppare; questa struttura, con l'avanzare del progetto, ha subito un processo di costante aggiornamento fino al raggiungimento di quella che, per ora, è la sua forma definitiva.

Al fine di fornire, fin da subito, una panoramica chiara ed esaustiva sulla struttura di questo schema, verrà riportata un'immagine illustrativa del diagramma E/R (semplificato) ottenuto (Figura 4.8). Con l'obiettivo di consentire una comprensione più rapida, in questa figura, non sono riportate tutte le entità che poi verranno incluse nel database; infatti, sono stati trascurati tutti gli oggetti che non influiscono in maniera determinante sul funzionamento completo dello schema.

Da questo diagramma, in DBeaver, è stato ricavato lo schema relazionale del database vero e proprio.

Partendo da questa figura, verranno ora analizzate tutte le principali relazioni presenti al suo interno, per poi procedere ad elencare anche tutte le tabelle di validazione (key-value) presenti nello schema.

4.2.1 Relazioni primarie

Si inizierà questo approfondimento dello schema relazionale seguendo la stessa logica con la quale è stata approcciata la mappatura di SAP; verranno, quindi, prima affrontate le

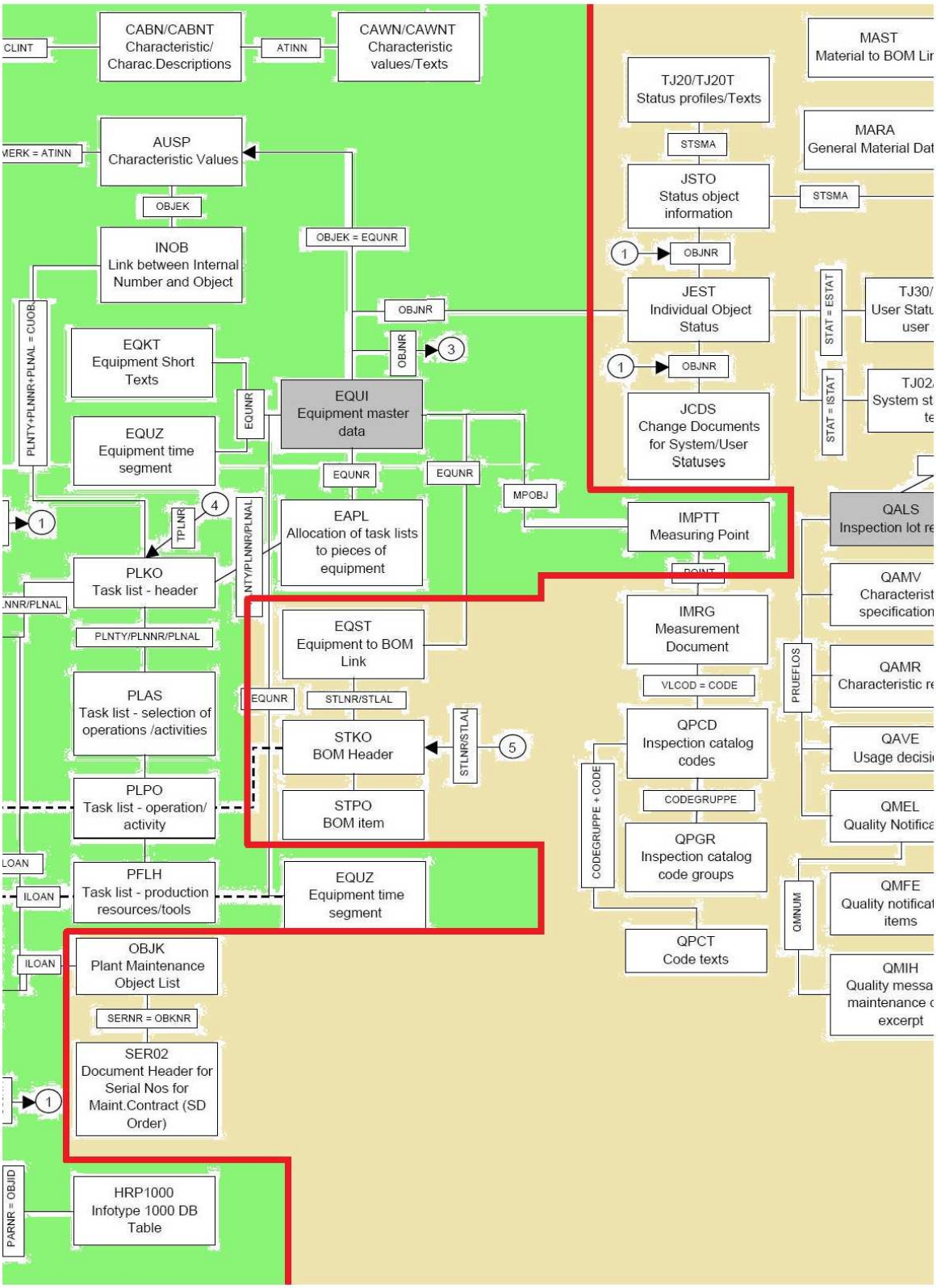


Figura 4.7: Tagli sullo schema SAP

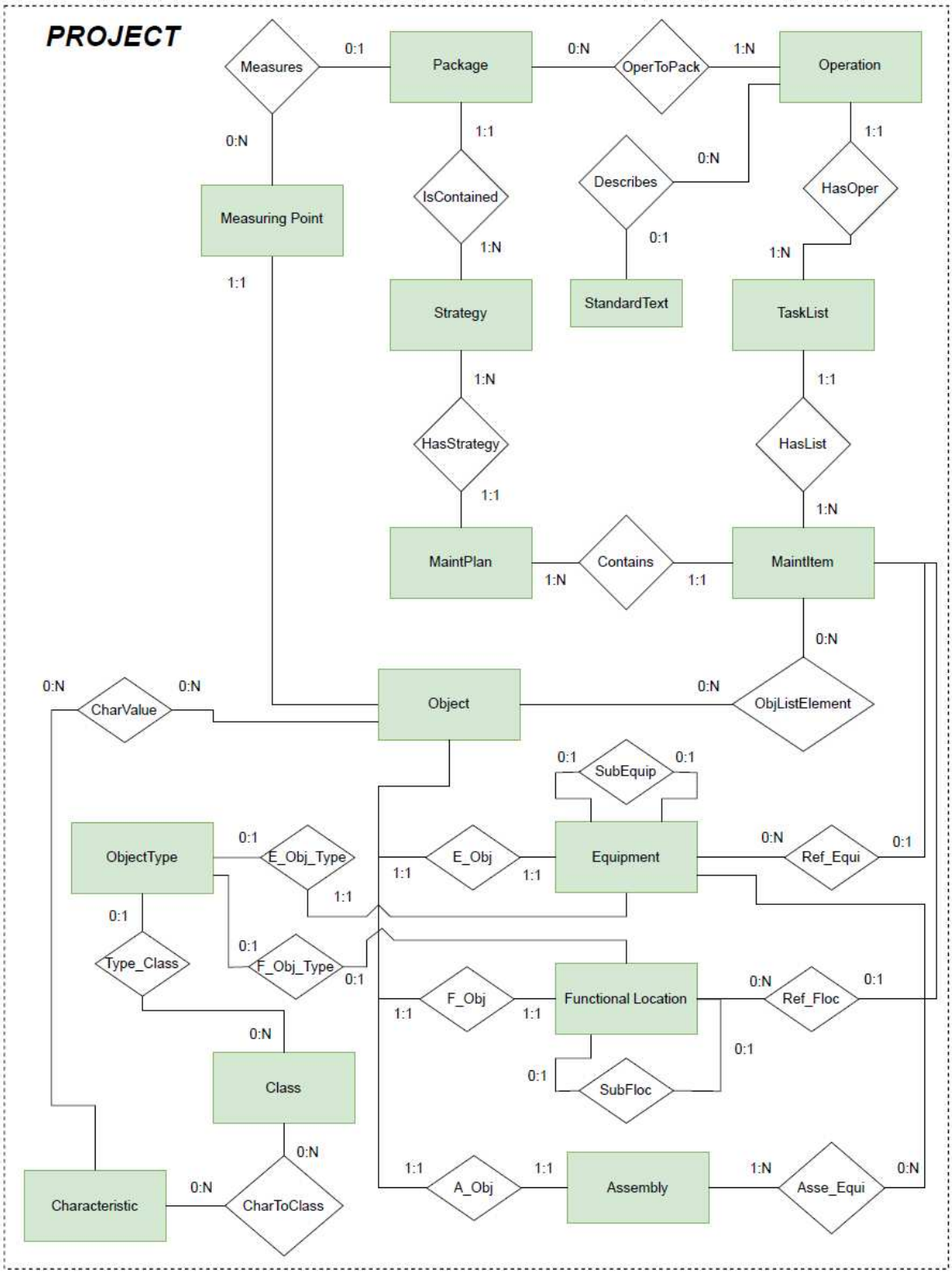


Figura 4.8: Schema E/R

relazioni chiave riconducibili all'asset register, per poi, in un secondo momento, procedere con quelle del maintenance register. Questa distinzione non verrà fatta per le tabelle secondarie, le quali saranno accorpate in un elenco unico.

Equipment

In questa relazione sono state incluse parte delle informazioni che nella mappatura SAP sono presenti nelle tabelle EQUI ed EQUZ. In Figura 4.9 viene riportata un'illustrazione degli attributi della relazione. Uno dei campi più interessanti è *equip_xlsx_code*; quest'ultimo è un campo unique finalizzato a mappare il valore dell'identificatore dell'equipment usato nel contesto del progetto. Le conseguenze di questa discrepanza fra l'identificatore interno al database e l'identificatore usato nella pratica del progetto saranno approfondite nella sezione relativa alla parte di dataexchange.

Per il resto, la maggior parte dei campi presenti in questa relazione sono relativi alle tabelle di validazione di cui si è discusso in precedenza. Infatti, è possibile notare la presenza di un gran numero di foreign key, utilizzate come mezzo di validazione per i valori degli attributi.

Una nota importante relativa a questa relazione riguarda la presenza degli attributi *Project* e *ObjectID*; senza entrare ora nel dettaglio, nelle prossime sezioni verranno dedicati degli approfondimenti alla gestione di questi attributi, per aggiunta presenti anche in altre relazioni.

In Figura 4.10 è possibile osservare i vincoli di chiave specificati per la relazione in questione. Un'ulteriore nota degna di attenzione è la presenza del campo project all'interno della primary key della relazione; anche questo aspetto verrà approfondito in una sezione dedicata.

Come è possibile osservare, per equipment il campo chiave è identificato dall'id dell'oggetto "astratto".

Le foreign key presenti in equipment, sono riportate in Figura 4.11.

Functional location

In questa relazione sono state incluse parte delle informazioni che nella mappatura SAP sono presenti nelle tabelle IFLOT e ILOA. In Figura 4.12 viene riportata un'illustrazione degli attributi della relazione.

In Figura 4.13 è possibile osservare i vincoli di chiave specificati per la relazione in questione. In questo caso, a differenza di quanto fatto per equipment, non si ha come campo chiave l'object ID; questo perché nelle applicazioni pratiche è più comodo avere la functional location identificata tramite una stringa, la quale può fornire informazioni utili sulla posizione della sede tecnica in questione nella gerarchia (concetto di struttura gerarchica delle functional location visto durante l'analisi delle tabelle SAP).

Le foreign key presenti in functional location sono riportate in Figura 4.14.

Assembly

In questa relazione sono state incluse le informazioni relative alla definizione di Assembly, che, come già accennato, in SAP viene ricondotto come concetto a "Material". In Figura 4.15 viene riportata un'illustrazione degli attributi della relazione.

In Figura 4.16 è possibile osservare i vincoli di chiave specificati per la relazione in questione. In questo caso, come fatto per equipment, si ha come campo chiave l'object ID.

Le foreign key presenti in assembly sono riportate in Figura 4.17.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
desc	2	varchar		default	[v]	
tech_id_num	3	varchar(25)		default	[ ]	
tech_obj_au...	4	varchar(4)		default	[ ]	
equip_cat	5	varchar(1)		default	[v]	
obj_type	6	varchar(10)		default	[v]	
invent_num	7	varchar(25)		default	[ ]	
dimension	8	varchar(18)		default	[ ]	
weight	9	float8			[ ]	
weight_uom	10	varchar(3)		default	[ ]	
acquisition_...	11	date			[ ]	
acquisition_...	12	float8			[ ]	
currency	13	varchar(5)		default	[ ]	
manufacturer	14	varchar(30)		default	[v]	
manu_coun...	15	varchar(3)		default	[ ]	
serial_num	16	varchar(30)		default	[v]	
model_num	17	varchar(20)		default	[v]	
startup_date	20	date			[ ]	
license_num	24	varchar(20)		default	[ ]	
user_status	25	varchar(4)		default	[v]	
floc	26	varchar		default	[ ]	
criticality	29	varchar(1)		default	[v]	
equip_tag_n...	31	varchar(30)		default	[v]	
manu_part_...	32	varchar(30)		default	[ ]	
constr_type	33	varchar(18)		default	[ ]	
effect	35	varchar(3)		default	[v]	
catalog_prof	36	varchar(9)		default	[ ]	
asset	37	varchar(12)		default	[ ]	
vendor	39	varchar(20)		default	[ ]	
constr_yy	41	int2			[ ]	
constr_mm	42	int2			[ ]	
objectid	44	int8			[v]	nextval('ar_mr....
equip_xlsx_...	45	varchar		default	[v]	
sup_equip	46	int8			[ ]	

Figura 4.9: Attributi di equipment

Nome	Attribute	Proprietario	Tipo
equipment_pk	—	equipment	PRIMARY KEY
project	project	—	—
objectid	objectid	—	—
equipment_un	—	equipment	UNIQUE KEY
equip_xlsx_code	equip_xlsx_cod	—	—
project	project	—	—

Figura 4.10: Chiavi di equipment

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
equipment_acquisitionCurrency	FOREIGN KEY	currency_pk	currency	Restrict	Cascade
equipment_authgroup	FOREIGN KEY	tech_obj_auth_group_	tech_obj_auth_grou	Set NULL	Cascade
equipment_catalogprof	FOREIGN KEY	catalog_profile_pk	catalog_profile	Restrict	Restrict
equipment_criticality	FOREIGN KEY	criticality_pk	criticality	Restrict	Cascade
equipment_effect	FOREIGN KEY	effect_pk	effect	Restrict	Cascade
equipment_equipcategory	FOREIGN KEY	equip_cat_pk	equip_cat	Restrict	Cascade
equipment_floc	FOREIGN KEY	fun_loc_pk	fun_loc	Set NULL	Cascade
equipment_manucountry	FOREIGN KEY	country_pk	country	Restrict	Cascade
equipment_objtype	FOREIGN KEY	obj_type_pk	obj_type	Set NULL	Cascade
equipment_supequip	FOREIGN KEY	equipment_pk	equipment	Restrict	Cascade
equipment_userstatus	FOREIGN KEY	user_status_pk	user_status	Restrict	Cascade
equipment_weightUoM	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.11: Chiavi esterne presenti in equipment

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
floc_id	2	varchar		default	[v]	
floc_cat	3	varchar(1)		default	[v]	
sup_floc	4	varchar		default	[ ]	
tech_obj_au...	6	varchar(4)		default	[ ]	
equip_instal...	7	bool			[v]	
single_equip	8	bool			[v]	
maint_plant	9	varchar(4)		default	[v]	
plann_group	10	varchar(3)		default	[ ]	
obj_type	12	varchar(10)		default	[ ]	
objectid	26	int8			[v]	nextval('ar_mr...
structure_in...	27	varchar		default	[v]	
desc	28	varchar		default	[v]	
work_center	29	varchar(8)		default	[ ]	
user_status	30	varchar(4)		default	[v]	
criticality	31	varchar(1)		default	[v]	
effect	32	varchar(3)		default	[v]	
location	33	varchar(10)		default	[v]	
plann_plant	34	varchar(4)		default	[v]	
sce	35	varchar(1)		default	[ ]	
cost_center	36	varchar(10)		default	[ ]	
wbs_element	37	varchar(24)		default	[ ]	
p&id	38	varchar		default	[ ]	

Figura 4.12: Attributi di functional location

Nome	Attribute	Proprietario	Tipo
fun_loc_pk	—	fun_loc	PRIMARY KEY
project	project	—	—
floc_id	floc_id	—	—
fun_loc_un	—	fun_loc	UNIQUE KEY
project	project	—	—
objectid	objectid	—	—

Figura 4.13: Chiavi di functional location

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
floc_authgroup	FOREIGN KEY	tech_obj_auth_group	tech_obj_auth_grou	Set NULL	Cascade
floc_costcenter	FOREIGN KEY	cost_center_pk	cost_center	Restrict	Cascade
floc_criticality	FOREIGN KEY	criticality_pk	criticality	Restrict	Cascade
floc_effect	FOREIGN KEY	effect_pk	effect	Restrict	Cascade
floc_floccategory	FOREIGN KEY	floc_cat_pk	floc_cat	Restrict	Cascade
floc_location	FOREIGN KEY	location_pk	location	Restrict	Cascade
floc_maintPlant	FOREIGN KEY	plant_pk	plant	Set NULL	Cascade
floc_objtype	FOREIGN KEY	obj_type_pk	obj_type	Set NULL	Cascade
floc_plannplant	FOREIGN KEY	plant_pk	plant	Restrict	Cascade
floc_structureindicator	FOREIGN KEY	structure_indicator_pk	structure_indicator	Restrict	Cascade
floc_userstatus	FOREIGN KEY	user_status_pk	user_status	Restrict	Cascade
floc_wbselem	FOREIGN KEY	wbs_elem_pk	wbs_elem	Restrict	Cascade
floc_workcenter	FOREIGN KEY	work_center_pk	work_center	Set NULL	Cascade
fun_loc_fk	FOREIGN KEY	mplann_group_pk	mplann_group	Restrict	Cascade
objectid_fk	FOREIGN KEY	object_pk	object	Cascade	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade
sup_floc	FOREIGN KEY	fun_loc_pk	fun_loc	Cascade	Cascade

Figura 4.14: Chiavi esterne presenti in functional location

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
assembly_c...	2	varchar(18)		default	[v]	
objectid	3	int8			[v]	nextval('ar_mr....

Figura 4.15: Attributi di assembly

Nome	Attribute	Proprietario	Tipo
assembly_pk	—	assembly	PRIMARY KEY
project	project	—	—
objectid	objectid	—	—
assembly_un	—	assembly	UNIQUE KEY
assembly_code	assembly_code	—	—

Figura 4.16: Chiavi di assembly

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.17: Chiavi esterne presenti in assembly

Assembly-Equipment

Questa struttura dati è il risultato della traduzione, dal modello E/R, della relazione fra le entità Equipment e Assembly; avendo come cardinalità massime N, viene rappresentata nel modello relazionale, con una relazione aggiuntiva avente come chiavi esterne i riferimenti alle due entità. In Figura 4.18 viene riportata un'illustrazione degli attributi della relazione; le chiavi sono ,invece, osservabili in Figura 4.19.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
assemblyid	2	int8			[v]	
equipid	3	int8			[v]	

Figura 4.18: Attributi di assembly-equipment

Nome	Attribute	Proprietario	Tipo	Expression
assembly_equip_pk	—	assembly_equip	PRIMARY KEY	
project	project	—	—	—
assemblyid	assemblyid	—	—	—
equipid	equipid	—	—	—

Figura 4.19: Chiavi di assembly-equipment

Le foreign key presenti nella relazione sono riportate in Figura 4.20. Come accennato in precedenza, è interessante osservare i collegamenti con le relazioni equipment e assembly

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
assembly_equip	FOREIGN KEY	equipment_pk	equipment	Cascade	Cascade
assembly_fk	FOREIGN KEY	assembly_pk	assembly	Cascade	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.20: Chiavi esterne in assembly-equipment

Object Type

Questa relazione è a limite dall'essere considerata una validation-table; tuttavia, data l'importanza del concetto di object type, dal quale è possibile dedurre moltissime informazioni legate alle diverse tipologie di attrezzature, si è deciso di inserirla fra le tabelle primarie. In Figura 4.21 viene riportata un'illustrazione degli attributi della relazione; le chiavi sono invece osservabili in Figura 4.22.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
obj_type	2	varchar(10)		default	[v]	
desc	3	varchar		default	[ ]	
class	4	varchar(18)		default	[ ]	

Figura 4.21: Attributi di object-type

Nome	Attribute	Proprietario	Tipo
obj_type_pk	—	obj_type	PRIMARY KEY
project	project	—	—
obj_type	obj_type	—	—

Figura 4.22: Chiavi in object-type

Le foreign key presenti nella relazione sono riportate in Figura 4.23. È interessante osservare la relazione che lega object type con class; quest'ultima, idealmente, avrebbe dovuto avere il campo "Not Null" uguale a true. Tuttavia, durante le fasi di caricamento dei dati storici, si è visto che, spesso, non è possibile risalire alla classe di appartenenza di un certo tipo di oggetto; di conseguenza per comodità, si è deciso di inserire la possibilià di avere valori nulli, e quindi object type non associati a classi. In futuro, probabilmente questo approccio dovrà essere rivisto.

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
obj_type_fk	FOREIGN KEY	class_pk	class	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.23: Chiavi esterne in object-type

Class

In questa relazione sono state incluse le informazioni relative alla definizione di Class, che, come specificato durante l'approfondimento di SAP, è un concetto molto vicino a quello di object type. In Figura 4.24 viene riportata un'illustrazione degli attributi della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
desc	2	varchar		default	[ ]	
class_code	4	varchar		default	[v]	

Figura 4.24: Attributi di class

In Figura 4.25 è possibile osservare i vincoli di chiave specificati per la relazione in questione.

Nome	Attribute	Proprietario	Tipo	Expression
class_pk	—	class	PRIMARY KEY	
project	project	—	—	—
class_code	class_code	—	—	—

Figura 4.25: Chiavi presenti in class

Le foreign key presenti in class sono riportate in Figura 4.26.

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.26: Chiavi esterne in class

Characteristic

In characteristic possono essere trovate alcune delle informazioni necessarie a descrivere una classe di oggetti. Durante la descrizione della mappatura SAP, questi dati sono stati riscontrati all'interno della tabella KSML. In Figura 4.27 viene riportata un'illustrazione degli attributi della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
char_code	2	varchar(30)		default	[v]	
data_type	3	varchar(4)		default	[v]	
characters_num	4	int4			[v]	0
decimals_num	5	int4			[v]	0
char_uom	6	varchar		default	[ ]	
desc	10	varchar(100)		default	[v]	

Figura 4.27: Attributi di characteristic

In Figura 4.28 è possibile osservare i vincoli di chiave specificati.

Nome	Attribute	Proprietario	Tipo
characteristic_pk	—	characteristic	PRIMARY KEY
char_code	char_code	—	—
project	project	—	—

Figura 4.28: Chiavi di characteristic

Le foreign key presenti nella relazione sono osservabili in Figura 4.29

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
characteristic_uom	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.29: Chiavi esterne presenti in characteristic

ChartoClass

Questa relazione è il risultato della traduzione, dal modello ER, della relationship CharToClass (molti a molti). Nelle Figure 4.30, 4.31, e 4.32 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

CharValue

Questa relazione contiene le informazioni relative ai valori assunti dalle caratteristiche che identificano determinati oggetti. Nello schema E/R per questo concetto era stata utilizzata

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
class	1	varchar(18)		default	[v]	
char_code	2	varchar(30)		default	[v]	
project	3	int4			[v]	

Figura 4.30: Attributi di char to class

Nome	Attribute	Proprietario	Tipo
char_to_class_pk	—	char_to_class	PRIMARY KEY
project	project	—	—
char_code	char_code	—	—
class	class	—	—

Figura 4.31: Chiavi di char to class

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
char_fk	FOREIGN KEY	characteristic_pk	characteristic	Cascade	Cascade
class_fk	FOREIGN KEY	class_pk	class	Cascade	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.32: Chiavi esterne in char to class

una relationship molti a molti fra Object e Characterists. Nelle Figure 4.33, 4.34, e 4.35 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
value	2	varchar		default	[v]	
objectid	3	int8			[v]	
char_code	5	varchar(30)		default	[v]	

Figura 4.33: Attributi di char value

Nome	Attribute	Proprietario	Tipo
char_value_pk	—	char_value	PRIMARY KEY
project	project	—	—
objectid	objectid	—	—
char_code	char_code	—	—

Figura 4.34: Chiavi in char value

Si passerà, ora, ad illustrare le relazioni logicamente appartenenti al maintenance register.

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
char_fk	FOREIGN KEY	characteristic_pk	characteristic	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.35: Chiavi esterne presenti in char value

Task List

Le informazioni presenti in questa relazione sono una parte di quelle riconducibili alle tabelle SAP PLKO, PLAS, PLPO, PFLH e TAPL. A differenza di quanto avviene in SAP, a far parte della chiave del TaskList non è presente il campo Task List Type. Nelle Figure 4.36, 4.37, e 4.38 sono riportati ,rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
tasklist_type	2	varchar(1)		default	[v]	
tasklist_status	3	varchar(3)		default	[ ]	
tasklist_group	4	varchar		default	[v]	
tasklist_usage	5	varchar(3)		default	[ ]	
planning_plant	6	varchar(4)		default	[ ]	
desc	7	varchar(40)		default	[ ]	
delete_indicator	10	varchar(1)		default	[ ]	
group_counter	11	int2			[v]	
resp_plann_gr...	12	varchar(3)		default	[ ]	
main_work_ce...	13	varchar(8)		default	[v]	

Figura 4.36: Attributi di task list

Nome	Attribute	Proprietario	Tipo
task_list_pk	—	task_list	PRIMARY KEY
project	project	—	—
tasklist_group	tasklist_group	—	—
group_counter	group_counter	—	—

Figura 4.37: Chiavi di task list

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade
task_list_fk	FOREIGN KEY	mplann_group_pk	mplann_group	Restrict	Cascade
tasklist_plant	FOREIGN KEY	plant_pk	plant	Restrict	Cascade
tasklist_tasklistGroup	FOREIGN KEY	tasklist_group_pk	tasklist_group	Restrict	Cascade
tasklist_tasklistStatus	FOREIGN KEY	tasklist_status_pk	tasklist_status	Restrict	Cascade
tasklist_tasklistType	FOREIGN KEY	tasklist_type_pk	tasklist_type	Restrict	Cascade
tasklist_tasklistUsage	FOREIGN KEY	tasklist_usage_pk	tasklist_usage	Restrict	Cascade
tasklist_workcenter	FOREIGN KEY	work_center_pk	work_center	Restrict	Cascade

Figura 4.38: Chiavi esterne in task list

Operation

Le informazioni presenti in questa relazione sono una parte di quelle riconducibili alle tabelle SAP PLAS, PLPO e PLMZ. Come chiave di questa relazione, come visto in SAP, fa parte anche il riferimento (tramite task list group e counter) al ciclo. Nelle Figure 4.39, 4.40, e 4.41 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
op_num	2	int2			[v]	
desc	3	varchar(40)		default	[v]	
control_key_c...	4	varchar(4)		default	[v]	
sys_condition	5	varchar(1)		default	[v]	
exec_fact	6	int2			[ ]	
fix_lot_ext_pr...	7	bool			[ ]	
duration	8	float4			[v]	
duration_uom	9	varchar(3)		default	[v]	
work_percent...	10	int2			[ ]	
tasklist_group	12	varchar		default	[v]	
group_counter	13	int2			[v]	
work_man_h	14	int4			[ ]	
work_center	16	varchar(8)		default	[v]	
std_text_code	18	varchar(7)		default	[ ]	
scmop	19	varchar(1)		default	[ ]	

Figura 4.39: Attributi di operation

Nome	Attribute	Proprietario	Tipo
operation_pk	—	operation	PRIMARY KEY
project	project	—	—
op_num	op_num	—	—
tasklist_group	tasklist_group	—	—
group_counter	group_counter	—	—

Figura 4.40: Chiavi di operation

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
operation_controlKey	FOREIGN KEY	control_key_pk	control_key	Restrict	Cascade
operation_stdtext	FOREIGN KEY	std_text_pk	std_text	Restrict	Cascade
operation_sysCondition	FOREIGN KEY	system_cond_pk	system_cond	Restrict	Cascade
operation_tasklist	FOREIGN KEY	task_list_pk	task_list	Cascade	Cascade
operation_workUoM	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.41: Chiavi esterne di operation

Standard Text

Questa relazione è a limite dall'essere considerata una validation table, ma, data la sua importanza per la definizione delle operazioni, viene inserita fra le tabelle primarie. Nelle Figure 4.42, 4.43, e 4.44 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
std_text_code	2	varchar(7)		default	[v]	
lang_key	3	varchar(2)		default	[ ]	
text	4	varchar		default	[v]	
desc	5	varchar		default	[ ]	

Figura 4.42: Attributi di standard text

Nome	Attribute	Proprietario	Tipo
std_text_pk	—	std_text	PRIMARY KEY
project	project	—	—
std_text_code	std_text_code	—	—

Figura 4.43: Chiavi di standard text

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.44: Chiavi esterne in standard text

Package

Questa relazione contiene le informazioni relative al concetto di package; quindi, in SAP, si sta facendo riferimento ai dati presenti nelle tabelle T351P e PLWP. Nelle Figure 4.45, 4.46, e 4.47 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

OperToPackage

La relazione in questione, riflette la relazione presente, nello schema E/R, fra le entità operation e package; è interessante osservare la cardinalità minima dalla parte di operation; infatti, questa è pari a 1. Ciò vuol dire che, per ogni attività, potranno essere definiti da uno a molti pacchetti di frequenza. Nelle Figure 4.48, 4.49 e 4.50 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Measuring Point

La relazione in questione serve a dare forma all'entità meausuring point, la quale, in SAP, viene fatta risalire alle informazioni presenti nella tabella IMPTT. Nelle Figure 4.51, 4.52 e 4.53 sono riportati ,rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
strategy	2	varchar(6)		default	[v]	
cycle_freq	4	float8			[ ]	
count_read	5	float8			[ ]	
offset_time	6	float8			[ ]	
offset_perf	7	float8			[ ]	
desc	8	varchar(30)		default	[v]	
counter_uom	10	varchar(10)		default	[ ]	
frequency_uom	11	varchar(10)		default	[ ]	
package_code	12	varchar(2)		default	[v]	
meas_point	13	varchar(20)		default	[ ]	

Figura 4.45: Attributi di package

Nome	Attribute	Proprietario	Tipo
package_pk	—	package	PRIMARY KEY
project	project	—	—
strategy	strategy	—	—
package_code	package_code	—	—

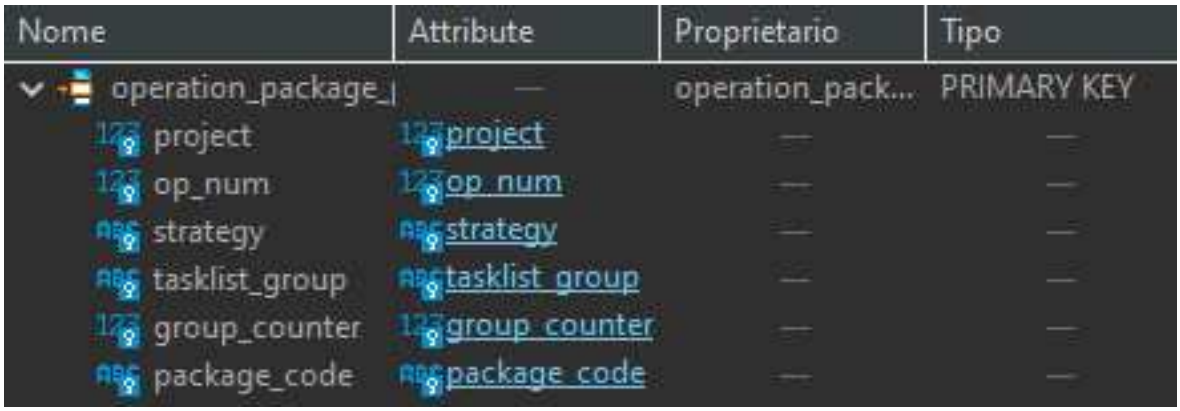
Figura 4.46: Chiavi di package

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
couteruom_fk	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
frequencyuom_fk	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
package_measpoint	FOREIGN KEY	meas_point_pk	meas_point	Restrict	Cascade
package_strategy	FOREIGN KEY	strategy_pk	strategy	Cascade	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.47: Chiavi esterne in package

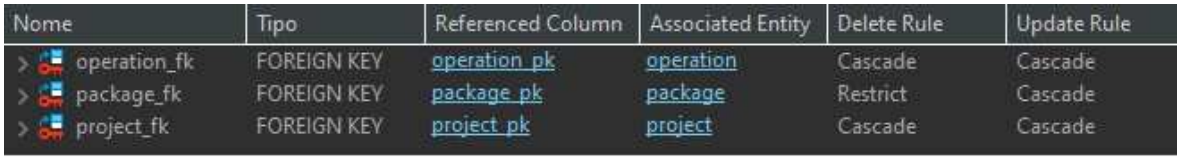
Nome colonna	#	Data type	Identity	Collation	Not Null
project	1	int4			[v]
op_num	2	int2			[v]
strategy	4	varchar(6)		default	[v]
tasklist_group	5	varchar		default	[v]
group_counter	6	int2			[v]
package_code	7	varchar(2)		default	[v]

Figura 4.48: Attributi di operation to package



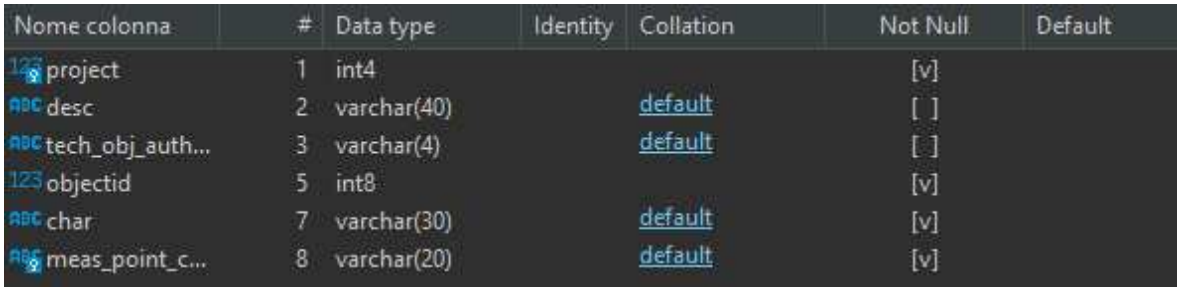
Nome	Attribute	Proprietario	Tipo
operation_package_	—	operation_pack...	PRIMARY KEY
project	project	—	—
op_num	op_num	—	—
strategy	strategy	—	—
tasklist_group	tasklist_group	—	—
group_counter	group_counter	—	—
package_code	package_code	—	—

Figura 4.49: Chiavi di operation to package



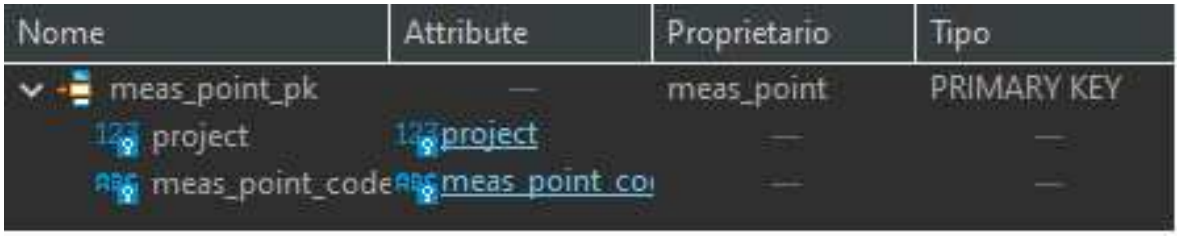
Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
operation_fk	FOREIGN KEY	operation_pk	operation	Cascade	Cascade
package_fk	FOREIGN KEY	package_pk	package	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.50: Chiavi esterne in operation to package



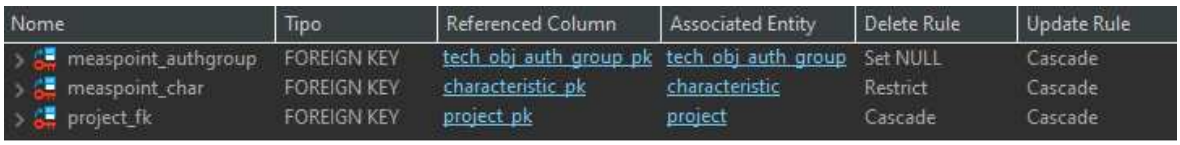
Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
desc	2	varchar(40)		default	[ ]	
tech_obj_auth...	3	varchar(4)		default	[ ]	
objectid	5	int8			[v]	
char	7	varchar(30)		default	[v]	
meas_point_c...	8	varchar(20)		default	[v]	

Figura 4.51: Attributi di measuring point



Nome	Attribute	Proprietario	Tipo
meas_point_pk	—	meas_point	PRIMARY KEY
project	project	—	—
meas_point_code	meas_point_co	—	—

Figura 4.52: Chiavi di measuring point



Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
measpoint_authgroup	FOREIGN KEY	tech_obj_auth_group_pk	tech_obj_auth_group	Set NULL	Cascade
measpoint_char	FOREIGN KEY	characteristic_pk	characteristic	Restrict	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.53: Chiavi esterne in measuring point

Strategy

Questa relazione contiene i dati della strategia di manutenzione (Tabella T351 in SAP). Nelle Figure 4.54, 4.55 e 4.56 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
strategy_code	2	varchar(6)		default	[v]	
performance_...	3	varchar(3)		default	[ ]	
frequency_uom	4	varchar(3)		default	[ ]	
tot_cycle_dur1	5	float8			[ ]	
tot_cycle_dur2	6	float8			[ ]	
min_step_mp...	7	float8			[ ]	
min_step_mp...	8	float8			[ ]	
desc	17	varchar(30)		default	[v]	
number_of_p...	18	int2			[ ]	

Figura 4.54: Attributi di strategy

Nome	Attribute	Proprietario	Tipo
strategy_pk	—	strategy	PRIMARY KEY
project	project	—	—
strategy_code	strategy_code	—	—

Figura 4.55: Chiavi di strategy

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade
strategy_frequencyUoM	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
strategy_performanceUoM	FOREIGN KEY	uom_pk	uom	Restrict	Cascade

Figura 4.56: Chiavi esterne di strategy

Maintenance Plan

Avvicinandosi sempre di più al core del maintenance register, si trova la relazione Maintenance Plan. Quest'ultima riflette la tabella SAP MPLA. Nelle Figure 4.57, 4.58 e 4.59 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Maintenance Item

Direttamente in successione al Maintenance Plan, si trova la relazione del Maintenance Item. Nelle Figure 4.60, 4.61 e 4.62 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
desc	2	varchar(40)		default	[v]	
strategy	3	varchar(6)		default	[v]	
sched_period_...	10	varchar(3)		default	[v]	
sched_indicator	11	varchar(1)		default	[ ]	
start_count_re...	15	varchar(22)		default	[ ]	
call_confirm	16	bool			[ ]	
tech_obj_auth...	17	varchar(4)		default	[ ]	
mplan_cat	18	varchar(2)		default	[ ]	
mplan_id	19	bigserial			[v]	nextval('ar_mr....
sf_latecomp	20	int2			[ ]	
sf_earlycomp	21	int2			[ ]	
tol_latecomp	22	int2			[ ]	
tol_earlycomp	23	int2			[ ]	
cycle_mod_fact	24	float4			[ ]	
sched_period	25	int2			[v]	
call_horiz	26	int2			[v]	
start_date	27	date			[ ]	
mplan_code	28	varchar		default	[ ]	

Figura 4.57: Attributi di Maintenance Plan

Nome	Attribute	Proprietario	Tipo
mplan_pk	—	mplan	PRIMARY KEY
mplan_id	mplan_id	—	—
project	project	—	—
mplan_un	—	mplan	UNIQUE KEY
mplan_code	mplan_code	—	—
project	project	—	—

Figura 4.58: Chiavi di Maintenance Plan

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
mplan_authGroup	FOREIGN KEY	tech_obj_auth_group_pk	tech_obj_auth_group	Set NULL	Cascade
mplan_mplanCategory	FOREIGN KEY	mplan_cat_pk	mplan_cat	Restrict	Cascade
mplan_schedPeriodUoM	FOREIGN KEY	uom_pk	uom	Restrict	Cascade
mplan_strategy	FOREIGN KEY	strategy_pk	strategy	Set NULL	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.59: Chiavi esterne in Maintenance Plan

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
desc	2	varchar(40)		default	[v]	
status	3	varchar(1)		default	[ ]	
plant	5	varchar(4)		default	[v]	
order_type	6	varchar(4)		default	[v]	
m_activity_type	8	varchar(3)		default	[ ]	
mitem_cat	9	varchar(2)		default	[ ]	
priority	10	varchar(5)		default	[v]	
not_release_i...	11	bool			[ ]	
task_list_fact	12	int2			[ ]	
mitem_id	16	bigserial			[v]	nextval('ar_mr....
mplan	17	int8			[v]	
tasklist_group	19	varchar		default	[v]	
group_counter	20	int2			[v]	
work_center	21	varchar(8)		default	[ ]	
plann_group	22	varchar(3)		default	[ ]	
mitem_code	23	varchar		default	[ ]	
ref_equip	25	int8			[ ]	
ref_floc	27	varchar		default	[ ]	

Figura 4.60: Attributi di Maintenance Item

Nome	Attribute	Proprietario	Tipo
mitem_pk	—	mitem	PRIMARY KEY
project	project	—	—
mitem_id	mitem_id	—	—
mitem_un	—	mitem	UNIQUE KEY
project	project	—	—
mitem_code	mitem_code	—	—

Figura 4.61: Chiavi di Maintenance Item

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
mitem_mactivitytype	FOREIGN KEY	m_activity_type_pk	m_activity_type	Set NULL	Cascade
mitem_mitemCategory	FOREIGN KEY	mitem_cat_pk	mitem_cat	Restrict	Cascade
mitem_orderType	FOREIGN KEY	order_type_pk	order_type	Set NULL	Cascade
mitem_planngroup	FOREIGN KEY	mplann_group_pk	mplann_group	Restrict	Cascade
mitem_planningPlant	FOREIGN KEY	plant_pk	plant	Restrict	Cascade
mitem_priority	FOREIGN KEY	priority_pk	priority	Set NULL	Cascade
mitem_ref_equip	FOREIGN KEY	equipment_pk	equipment	Restrict	Cascade
mitem_ref_floc	FOREIGN KEY	fun_loc_pk	fun_loc	Restrict	Cascade
mitem_tasklist	FOREIGN KEY	task_list_pk	task_list	Restrict	Cascade
mitem_workcenter	FOREIGN KEY	work_center_pk	work_center	Set NULL	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.62: Chiavi esterne di Maintenance Item

ObjListElement

Per concludere la descrizione delle tabelle principali dello schema, si chiuderà il cerchio unendo l'asset e il maintenance register con la relazione object list element. Questa rappresenta la relationship di tipo molti a molti che, nello schema E/R, congiunge il Maintenance Item e l'Object. In questo modo, puntando a Object, nella Object List potranno essere introdotti assembly, equipment e sedi tecniche in maniera indistinta. Nelle Figure 4.63, 4.64 e 4.65 sono riportati, rispettivamente, gli attributi, le chiavi e le foreign key della relazione.

Nome colonna	#	Data type	Identity	Collation	Not Null	Default
project	1	int4			[v]	
mitem	2	int8			[v]	
objectid	3	int8			[v]	

Figura 4.63: Attributi di object list

Nome	Attribute	Proprietario	Tipo
obj_list_element_pk	—	obj_list_element	PRIMARY KEY
project	project	—	—
mitem	mitem	—	—
objectid	objectid	—	—

Figura 4.64: Chiavi di object list

Nome	Tipo	Referenced Column	Associated Entity	Delete Rule	Update Rule
objlist_mitem	FOREIGN KEY	mitem_pk	mitem	Cascade	Cascade
project_fk	FOREIGN KEY	project_pk	project	Cascade	Cascade

Figura 4.65: Chiavi esterne in object list

4.2.2 Relazioni secondarie

Per ragioni di brevità, dato lo scarso impatto che queste relazioni hanno sull'effettivo significato complessivo dello schema, si è deciso di non approfondirle come fatto con le precedenti; ci si limiterà a fornire, per ognuna di esse, una breve descrizione inerente il suo significato. Se si volesse approfondire la struttura di queste relazioni secondarie, il rimando è alla sezione contenente il DDL dell'intero database. Tali relazioni secondarie sono le seguenti:

- *Catalog Profile*: elenco di configurazioni specifiche, utilizzate per definire il comportamento e le opzioni disponibili nel catalogo di oggetti tecnici.
- *Company*: rappresenta un'azienda/una compagnia rilevante per lo scopo del progetto.
- *Control Key*: insieme di valori finalizzati a definire il comportamento di una transazione o di un ordine di manutenzione.
- *Cost Center*: un ramo aziendale responsabile della gestione dei costi di determinate operazioni.

- *Country*: validation table dei paesi.
- *Criticality*: validation table dei livelli di criticità di determinate operazioni.
- *Effect*: validation table dei livelli di danno di determinati guasti sul sistema.
- *Currency*: validation table delle monete.
- *Equipment Category*: validation table delle categorie di equipment.
- *Floc Category*: validation table delle categorie di sede tecnica.
- *Location*: indica una posizione fisica in un impianto, ad esempio una stanza, un reparto o un armadio.
- *Activity Type*: validation table delle tipologie di attività manutentiva.
- *Maintenance Item Category*: validation table delle diverse tipologie di posizione di manutenzione.
- *Maintenance Plan Category*: validation table delle diverse tipologie di piani di manutenzione.
- *Planning Group*: gruppo responsabile della gestione delle attività manutentive su determinati asset.
- *Order Type*: tipologia di ordine di manutenzione.
- *Plant*: validation table degli impianti in cui sono presenti degli asset rilevanti per il progetto.
- *Priority*: validation table dei livelli di priorità che un'operazione o un elemento di pianificazione può avere.
- *Structure Indicator*: valori di validazione per gli identificatori che specificano come esprimere la codifica delle functional location in rapporto alla loro struttura gerarchica.
- *System Condition*: condizioni di un determinato asset durante lo svolgimento delle attività manutentive.
- *Task List Group*: validation table finalizzata ad elencare tutti i possibili gruppi di cicli accettati dal sistema.
- *Task List Status*: validation table dei diversi stati in cui può trovarsi un ciclo.
- *Task List Usage*: validation table dei diversi fini/usi/scopi che possono essere associati ad un ciclo.
- *Authorization Group*: gruppo di lavoratori responsabile di rilasciare l'autorizzazione a procedere con determinate attività su degli asset.
- *Unit Of Measurement*: validation table delle diverse unità di misura accettate dal sistema.
- *User Status*: elenco di identificatori finalizzati a registrare/monitorare lo stato corrente di un oggetto nel suo ciclo di vita.
- *WBS Element*: elemento della Work Breakdown Structure aziendale incaricato dello svolgimento di determinate operazioni.

- *Work Center*: elenco di entità operative impiegate nella pianificazione e l'esecuzione delle attività di manutenzione. Un Work Center rappresenta una posizione o un'area specifica all'interno di un impianto industriale dove vengono svolte diverse attività di manutenzione e lavori tecnici.

4.2.3 Note sulle soluzioni implementative adottate

In questa sezione, verranno approfondite alcune delle soluzione più caratteristiche adottate durante la fase di sviluppo del database.

Divisione per progetto

Una delle necessità che, fin da subito, il progetto si imponeva di affrontare, era quella relativa alla separazione dei dati in base al progetto di appartenenza. Ciò vuol dire che ogni dato presente nel sistema deve poter essere ricondotto ad un progetto preciso.

Questa divisione verticale, nell'ambito aziendale in cui il progetto è stato portato avanti, può essere utile sotto molti punti di vista. Ad esempio:

- Tenere i dati separati per progetto consente una tracciabilità più chiara delle attività e delle risorse associate a ciascun progetto. Ciò facilita l'analisi delle prestazioni, la valutazione degli impatti e la comprensione dettagliata del progresso di ogni iniziativa.
- I progetti possono coinvolgere risorse diverse, sia umane che materiali. Mantenendo i dati separati, è possibile gestire in modo più efficace l'allocazione delle risorse, garantendo che queste siano assegnate in modo ottimale in base alle esigenze specifiche di ciascun progetto.
- La separazione dei dati per progetto consente un controllo più preciso del budget. Ogni progetto può avere il proprio budget dedicato, facilitando il monitoraggio delle spese, l'identificazione di eventuali variazioni e la gestione finanziaria in generale.
- L'assegnazione di dati specifici a ciascun progetto aiuta a stabilire chiaramente le responsabilità. Ciò è cruciale per garantire che i team siano consapevoli dei propri compiti e che siano in grado di prendere decisioni tempestive e informate all'interno del contesto del progetto.
- Ogni progetto può richiedere processi specifici o adattamenti. Mantenendo i dati separati, è possibile personalizzare i flussi di lavoro, nonché i modelli e le procedure per soddisfare le esigenze particolari di ciascun progetto, migliorando, così, l'efficienza e l'efficacia.
- Nel caso di Pansoinco, si è detto che molti progetti potrebbero essere soggetti a standard rappresentativi particolari. La separazione dei dati consente di aderire più facilmente a tali requisiti, facilitando la gestione della conformità.
- La possibilità di generare report specifici per ogni progetto semplifica la creazione di analisi dettagliate. Questi report possono essere utilizzati per valutare il successo del progetto, identificare aree di miglioramento e supportare la presa di decisioni strategiche.

In generale, mantenere i dati separati per progetto offre un maggiore controllo, una maggiore flessibilità ed una maggiore precisione nella gestione delle attività aziendali. Questo approccio contribuisce a ottimizzare l'utilizzo delle risorse, migliorare l'efficienza operativa e garantire il successo complessivo dei progetti.

Un'altra nota da fare in merito a questa soluzione riguarda il grado di ridondanza; infatti, è facile capire come nel database siano presenti dei concetti che potrebbero essere tranquillamente rappresentati senza nessun riferimento ad un progetto preciso. Ad esempio, basta pensare ai concetti di Country, Currency o UoM; questi elementi hanno un significato non molto influenzato dal concetto di progetto.

Tuttavia, al fine di mantenere una separazione totale, e quindi dividere anche questi aspetti legati alla parte di validazione, si è deciso di introdurre una ridondanza, riproponendo questi elementi per tutti i progetti (quindi, si sarà in grado di distinguere le valute, i paesi e le unità di misura utilizzate nell'ambito di un progetto). Per di più, si tratta di moli di dati tali da non rendere preoccupante il grado di duplicazione introdotto.

Sotto il punto di vista implementativo, questa separazione è stata introdotta creando una relazione *Project*, con all'interno un elenco "chiave valore", rappresentativo di tutti i progetti presenti nel sistema. Fatto ciò, per ogni singola relazione presente nello schema, è stata aggiunta una foreign key a project; per realizzare materialmente la separazione è stato sufficiente includere questi riferimenti anche nelle chiavi primarie di tutte le tabelle. Nella Figura 4.66 si vede come ogni relazione presente nello schema faccia riferimento a project, e come il campo relativo alla foreign key faccia anche parte della chiave primaria.

Un'ultima nota relativa all'implementazione della divisione per progetto riguarda il comportamento previsto a seguito della cancellazione di un intero progetto. Infatti, si può notare come ogni vincolo di chiave esterna verso project sia del tipo "*on delete: CASCADE*"; ciò vuol dire che, nel momento in cui verrà a mancare il progetto a cui il record fa riferimento, anche quest'ultimo verrà cancellato. In questo modo, eliminando un progetto, verranno cancellati dal sistema anche tutti i dati che ad esso fanno riferimento.

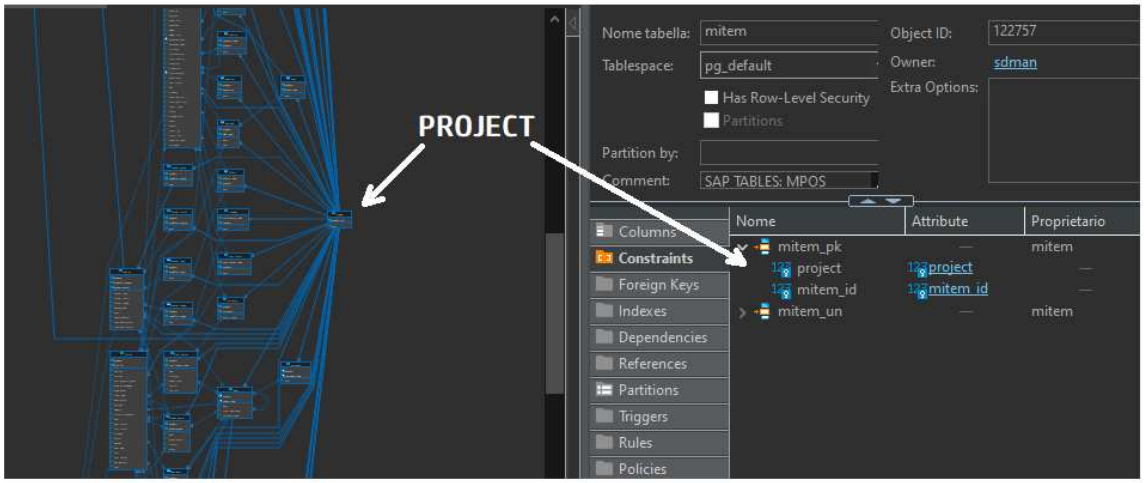


Figura 4.66: Relazione Project

ID Sequenziali

Una seconda soluzione adottata durante lo sviluppo del database è quella relativa all'utilizzo di identificatori sequenziali. In PostgreSQL, una sequenza è un oggetto database progettato per generare valori univoci in modo incrementale, comunemente utilizzato per generare chiavi primarie in tabelle. Queste sequenze forniscono un meccanismo affidabile per assegnare automaticamente valori univoci a colonne, garantendo che ogni valore generato sia diverso da qualsiasi altro valore generato in precedenza.

Caratterizzate da un nome e definite con il tipo di dati serial o bigserial, le sequenze di PostgreSQL sono gestite in modo indipendente dalle tabelle alle quali sono associate.

Una sequenza può essere utilizzata da più tabelle e non è direttamente legata a una tabella specifica. Tuttavia, è spesso associata a una colonna di una tabella come valore predefinito o tramite il comando *nextVal().*

Il valore di una sequenza può essere recuperato utilizzando la funzione *nextVal()* e può anche essere restituito senza incrementare il valore corrente utilizzando la funzione *currVal()*. Inoltre, è possibile ripristinare manualmente il valore corrente della sequenza utilizzando il comando *setVal()*.

Nel caso del database descritto in questo progetto, le sequenze sono state impiegate per tutte quelle tabelle per le quali risulta molto comodo generare una chiave primaria univoca al momento della creazione, senza dover ricorrere a particolari codifiche. In particolare, sono state definite le seguenti sequenze, osservabili anche in Figura 4.67:

- *mitem_id_seq*: sequenza che genera chiavi primarie per la relazione maintenance item.
- *mplan_id_seq*: sequenza che genera chiavi primarie per la relazione maintenance plan.
- *objectid_seq*: sequenza che genera chiavi primarie per la relazione object.
- *project_id_seq*: sequenza che genera chiavi primarie per la relazione project.

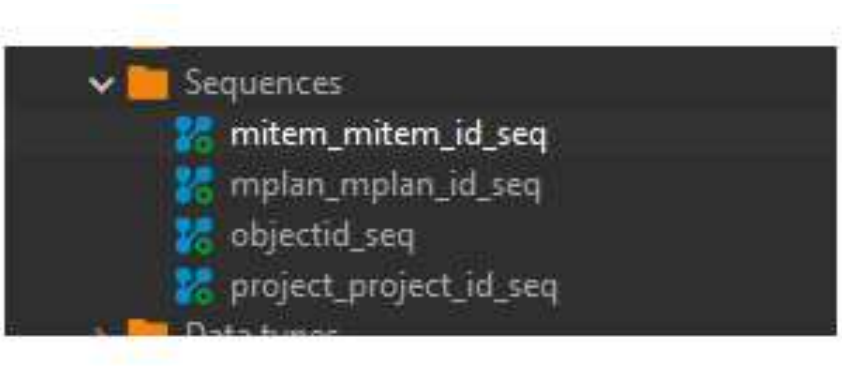


Figura 4.67: Sequenze definite

Come accennato in precedenza, la problematica più ingombrante introdotta dall'utilizzo di queste sequenze per generare chiavi primarie è quella relativa alla discrepanza con i dati storici. Infatti, se si immagina di dover caricare un progetto giunto al termine, verranno rotti tutti i riferimenti interni ad esso che vertono su concetti che nel database vengono etichettati automaticamente dalle sequenze. Ad esempio, se si immagina di caricare il piano di manutenzione identificato nel progetto come "Piano1", quest'ultimo, in fase di caricamento verrà inserito nella relazione maintenance plan con un ID univoco, dettato dal *nextVal()* di *mplan_id_seq*. A questo punto, se si volessero caricare i maintenance item facenti riferimento al progetto caricato, ci sarebbe un problema di perdita dei riferimenti, in quanto questi sono espressi usando gli identificatori definiti durante il progetto, e non quelli nel database.

Questa problematica, come già detto in precedenza, ha portato, durante la fase di caricamento, a dover adottare diverse strategie per rimediare a tutto ciò. Una di queste, prevedeva il mantenimento degli identificatori utilizzati in ambiti esterni per identificare determinati concetti; riguardando la sezione in cui sono state descritte tutte le relazioni, è possibile accorgersi di attributi come *mitem_code*, *mplan_code* o *equip_xls_code*. Tutti questi attributi servono per tener traccia dei vecchi identificatori utilizzati, in modo tale da non perdere traccia dei riferimenti in fase di caricamento.

Queste soluzioni verranno ulteriormente approfondite durante la sezione dedicata alla fase di data exchange.

Un'ultima precisazione da fare in merito alle sequenze riguarda il loro utilizzo nel caso di separazione verticale per progetto. Infatti, come detto in precedenza, non essendo le sequenze legate intrinsecamente a nessuna tabella, esse potrebbero esprimere un concetto di univocità esterna ad ogni progetto. Tutto questo per dire che le tabelle indicizzate tramite identificatori sequenziali non necessiterebbero di avere come parte della chiave l'attributo project. Tuttavia, al fine di mantenere una continuità logica attraverso tutte le relazioni, si è scelto di includere il riferimento a project anche nelle chiavi di queste relazioni, anche se, come detto, ciò non fornisce alcun valore aggiunto.

Trigger

Per concludere questa sezione dedicata all'approfondimento delle soluzioni adottate per l'implementazione del database si parlerà delle funzioni e dei trigger.

Un trigger in PostgreSQL è un oggetto che viene associato a una tabella specifica ed è attivato automaticamente quando si verifica un determinato evento sulla tabella stessa. Gli eventi che possono attivare un trigger includono l'aggiunta, la modifica o l'eliminazione di dati nella tabella di riferimento. I trigger sono utilizzati per eseguire azioni personalizzate, definite dall'utente in risposta a tali eventi. Questi comportamenti personalizzati vengono definiti tramite delle funzioni aventi come tipo di ritorno proprio Trigger.

L'attivazione di questi oggetti può essere caratterizzata sotto vari punti di vista. Ad esempio un trigger può essere attivato in "before" o in "after". Nel primo caso il trigger scatta prima che l'evento abbia effetto sulla tabella; questa azione può essere utilizzata per validare o modificare i dati prima dell'inserimento, dell'aggiornamento o dell'eliminazione. Nel secondo caso, invece, il trigger scatta dopo che l'evento ha avuto effetto sulla tabella; ciò può tornare utile per eseguire azioni successive all'inserimento, all'aggiornamento o all'eliminazione dei dati.

Durante l'implementazione del database, i trigger hanno giocato un ruolo essenziale per la relazione fra le tabelle equipment/assembly/functional location e object. Infatti, quest'ultima è stata definita come una generalizzazione delle prime tre relazioni. Di conseguenza, è stato necessario garantire il mantenimento di una corrispondenza uno ad uno fra questi oggetti. A tal fine, sono stati impiegati i trigger. Ogni volta che si ha un Insert o un Delete su una di queste tabelle, viene fatto partire un trigger che riapplica la stessa azione anche sulla tabella object, la quale, come detto in precedenza, è indicizzata in maniera sequenziale.

Nelle Figure 4.68, 4.69 e 4.70 vengono fornite una rappresentazione dei trigger presenti nella relazione equipment, il codice di uno di essi e la relativa funzione. Da notare che, nel contesto della funzione, con le parole chiave *New* e *Old* si identificano rispettivamente il nuovo e il vecchio valore del record in questione. In questo caso, inserendo un nuovo equipment, il last value della sequenza object verrà incrementato dal valore di default del campo *object_id* di equipment ("valorizzato" al *nextVal()*), il quale verrà inserito nella relazione object.

Name	Timing	Manipulation	Type	Function
equip_objec...	BEFORE	[INSERT]	ROW	object_insert
equip_objec...	AFTER	[DELETE]	ROW	object_delete

Figura 4.68: Trigger in equipment

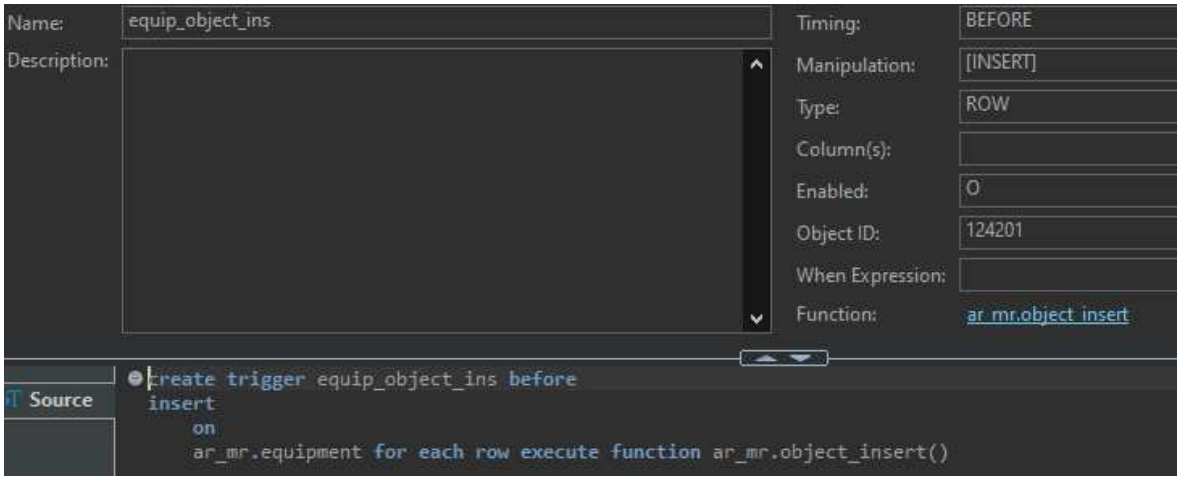


Figura 4.69: Trigger di inserimento in Object

```
-- DROP FUNCTION ar_mr.object_insert();

CREATE OR REPLACE FUNCTION ar_mr.object_insert()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN
    INSERT INTO
        ar_mr.object(objectid)
        VALUES(new.objectid);
           RETURN new;
END;
$function$
;
```

Figura 4.70: Funzione di inserimento in Object

4.2.4 DDL integrale

Al fine di fornire un quadro completo di ogni struttura presente nella base di dati, di seguito, verrà riportato il Data Definition Language in grado di generare una copia identica della struttura dati definita.

```
-- ar_mr."object" definition
CREATE TABLE "object" (
        objectid int8 NOT NULL,
        CONSTRAINT object_pk PRIMARY KEY (objectid)
);

-- ar_mr.project definition
CREATE TABLE project (
        project_id serial4 NOT NULL,
        "desc" varchar(40) NULL,
        CONSTRAINT project_pk PRIMARY KEY (project_id)
);

-- ar_mr.assembly definition
CREATE TABLE assembly (
        project int4 NOT NULL,
        assembly_code varchar(18) NOT NULL,
```

```
        objectid int8 NOT NULL DEFAULT nextval('ar_mr.objectid_seq'::regclass),
        CONSTRAINT assembly_pk PRIMARY KEY (project, objectid),
        CONSTRAINT assembly_un UNIQUE (assembly_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- Table Triggers
create trigger assembly_object_ins before
insert
    on
    ar_mr.assembly for each row execute function ar_mr.object_insert();
create trigger assemvly_object_del after
delete
    on
    ar_mr.assembly for each row execute function ar_mr.object_delete();

-- ar_mr.catalog_profile definition
CREATE TABLE catalog_profile (
        catalog_code varchar(9) NOT NULL,
        project int4 NOT NULL,
        "desc" varchar NULL,
        CONSTRAINT catalog_profile_pk PRIMARY KEY (project, catalog_code),
        CONSTRAINT catalog_profile_fk FOREIGN KEY (project) REFERENCES project(project_id)
        ↪  ON DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr."class" definition
CREATE TABLE "class" (
        project int4 NOT NULL,
        "desc" varchar NULL,
        class_code varchar NOT NULL,
        CONSTRAINT class_pk PRIMARY KEY (project, class_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.company definition
CREATE TABLE company (
        project int4 NOT NULL,
        company_code varchar(4) NOT NULL,
        "desc" varchar(25) NULL,
        CONSTRAINT company_pk PRIMARY KEY (project, company_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.control_key definition
CREATE TABLE control_key (
        project int4 NOT NULL,
        control_key_code varchar NOT NULL,
        extern_processing varchar(1) NULL,
        scheduling bool NULL,
        "desc" varchar(40) NULL,
        complete_conf varchar(1) NULL,
        "cost" bool NULL,
        CONSTRAINT control_key_pk PRIMARY KEY (project, control_key_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.country definition
CREATE TABLE country (
        country_code varchar(3) NOT NULL,
        project int4 NOT NULL,
        "desc" varchar(50) NULL,
        CONSTRAINT country_pk PRIMARY KEY (country_code, project),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.criticality definition
```

```
CREATE TABLE criticality (
        criticality_code varchar(1) NOT NULL,
        "desc" varchar(30) NULL,
        project int4 NOT NULL,
        CONSTRAINT criticality_pk PRIMARY KEY (criticality_code, project),
        CONSTRAINT criticality_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.currency definition
CREATE TABLE currency (
        project int4 NOT NULL,
        currency varchar(5) NOT NULL,
        isocurr_code varchar(3) NOT NULL,
        CONSTRAINT currency_pk PRIMARY KEY (project, currency),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.effect definition
CREATE TABLE effect (
        effect_code varchar(3) NOT NULL,
        project int4 NOT NULL,
        "desc" varchar NULL,
        CONSTRAINT effect_pk PRIMARY KEY (effect_code, project),
        CONSTRAINT effect_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.equip_cat definition
CREATE TABLE equip_cat (
        project int4 NOT NULL,
        equip_cat varchar(1) NOT NULL,
        "desc" varchar NULL,
        CONSTRAINT equip_cat_pk PRIMARY KEY (project, equip_cat),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.floc_cat definition
CREATE TABLE floc_cat (
        project int4 NOT NULL,
        floc_cat varchar(1) NOT NULL,
        "desc" varchar(30) NULL,
        CONSTRAINT floc_cat_pk PRIMARY KEY (project, floc_cat),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.m_activity_type definition
CREATE TABLE m_activity_type (
        project int4 NOT NULL,
        act_type_code varchar(3) NOT NULL,
        "desc" varchar(30) NULL,
        CONSTRAINT m_activity_type_pk PRIMARY KEY (project, act_type_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.mitem_cat definition
CREATE TABLE mitem_cat (
        project int4 NOT NULL,
        mitem_cat varchar(2) NOT NULL,
        "desc" varchar(30) NULL,
        CONSTRAINT mitem_cat_pk PRIMARY KEY (project, mitem_cat),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.mplan_cat definition
CREATE TABLE mplan_cat (
        project int4 NOT NULL,
```

```
        mplan_cat varchar(2) NOT NULL,
        "desc" varchar(40) NULL,
        CONSTRAINT mplan_cat_pk PRIMARY KEY (project, mplan_cat),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.obj_type definition
CREATE TABLE obj_type (
        project int4 NOT NULL,
        obj_type varchar(10) NOT NULL,
        "desc" varchar NULL,
        "class" varchar(18) NULL,
        CONSTRAINT obj_type_pk PRIMARY KEY (project, obj_type),
        CONSTRAINT obj_type_fk FOREIGN KEY (project,"class") REFERENCES
        ↪  "class"(project,class_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.order_type definition
CREATE TABLE order_type (
        project int4 NOT NULL,
        order_type varchar(4) NOT NULL,
        CONSTRAINT order_type_pk PRIMARY KEY (project, order_type),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.plant definition
CREATE TABLE plant (
        project int4 NOT NULL,
        plant_code varchar(4) NOT NULL,
        "desc" varchar(30) NULL,
        maint_plann_plant varchar(4) NULL,
        company_code varchar(4) NULL,
        CONSTRAINT plant_pk PRIMARY KEY (project, plant_code),
        CONSTRAINT plant_company FOREIGN KEY (project,company_code) REFERENCES
        ↪  company(project,company_code) ON DELETE RESTRICT ON UPDATE RESTRICT,
        CONSTRAINT plant_recursive FOREIGN KEY (project,maint_plann_plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.plant_section definition
CREATE TABLE plant_section (
        project int4 NOT NULL,
        plant varchar(4) NOT NULL,
        plant_section varchar(3) NOT NULL,
        person_responsible varchar(14) NULL,
        person_tel_num varchar(12) NULL,
        CONSTRAINT plant_section_pk PRIMARY KEY (project, plant, plant_section),
        CONSTRAINT plant_section_fk FOREIGN KEY (project,plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.priority definition
CREATE TABLE priority (
        project int4 NOT NULL,
        priority_code varchar(5) NOT NULL,
        "desc" varchar NULL,
        CONSTRAINT priority_pk PRIMARY KEY (project, priority_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.std_text definition
CREATE TABLE std_text (
        project int4 NOT NULL,
```

```
        std_text_code varchar(7) NOT NULL,
        lang_key varchar(2) NULL,
        "text" varchar NOT NULL,
        "desc" varchar NULL,
        CONSTRAINT std_text_pk PRIMARY KEY (project, std_text_code),
        CONSTRAINT std_text_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.structure_indicator definition
CREATE TABLE structure_indicator (
        structure_indicator varchar NOT NULL,
        hierarchy_levels varchar(40) NULL,
        edit_mask varchar(40) NULL,
        project int4 NOT NULL,
        CONSTRAINT structure_indicator_pk PRIMARY KEY (project, structure_indicator),
        CONSTRAINT structure_indicator_fk FOREIGN KEY (project) REFERENCES
        ↪  project(project_id) ON DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.system_cond definition
CREATE TABLE system_cond (
        project int4 NOT NULL,
        sys_cond_code varchar NOT NULL,
        "desc" varchar(40) NULL,
        CONSTRAINT system_cond_pk PRIMARY KEY (project, sys_cond_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.tasklist_group definition
CREATE TABLE tasklist_group (
        project int4 NOT NULL,
        tasklist_group varchar(10) NOT NULL,
        "desc" varchar(40) NULL,
        CONSTRAINT tasklist_group_pk PRIMARY KEY (project, tasklist_group),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.tasklist_status definition
CREATE TABLE tasklist_status (
        project int4 NOT NULL,
        tasklist_status varchar(3) NOT NULL,
        "desc" varchar(40) NULL,
        CONSTRAINT tasklist_status_pk PRIMARY KEY (project, tasklist_status),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.tasklist_type definition
CREATE TABLE tasklist_type (
        project int4 NOT NULL,
        tasklist_type varchar(1) NOT NULL,
        "desc" varchar(30) NULL,
        CONSTRAINT tasklist_type_pk PRIMARY KEY (project, tasklist_type),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.tasklist_usage definition
CREATE TABLE tasklist_usage (
        project int4 NOT NULL,
        tasklist_usage varchar(3) NOT NULL,
        "desc" varchar(20) NULL,
        CONSTRAINT tasklist_usage_pk PRIMARY KEY (project, tasklist_usage),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.tech_obj_auth_group definition
CREATE TABLE tech_obj_auth_group (
```

```
        project int4 NOT NULL,
        group_code varchar(4) NOT NULL,
        "desc" varchar(20) NULL,
        CONSTRAINT tech_obj_auth_group_pk PRIMARY KEY (project, group_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.uom definition
CREATE TABLE uom (
        project int4 NOT NULL,
        uom_code varchar NOT NULL,
        "desc" varchar(20) NOT NULL,
        dimension varchar(6) NULL,
        isocode varchar(3) NULL,
        CONSTRAINT uom_pk PRIMARY KEY (project, uom_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.user_status definition
CREATE TABLE user_status (
        user_status_code varchar(4) NOT NULL,
        project int4 NOT NULL,
        "desc" varchar NULL,
        CONSTRAINT user_status_pk PRIMARY KEY (user_status_code, project),
        CONSTRAINT user_status_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.wbs_elem definition
CREATE TABLE wbs_elem (
        project int4 NOT NULL,
        "desc" varchar NULL,
        resp_name varchar(25) NULL,
        plant varchar(4) NULL,
        wbs_code varchar(24) NOT NULL,
        CONSTRAINT wbs_elem_pk PRIMARY KEY (project, wbs_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT "wbsElem_plant" FOREIGN KEY (project,plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE SET NULL ON UPDATE CASCADE
);

-- ar_mr.work_center definition
CREATE TABLE work_center (
        project int4 NOT NULL,
        "desc" varchar(40) NULL,
        work_center_code varchar(8) NOT NULL,
        CONSTRAINT work_center_pk PRIMARY KEY (project, work_center_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.characteristic definition
CREATE TABLE characteristic (
        project int4 NOT NULL,
        char_code varchar(30) NOT NULL,
        data_type varchar(4) NOT NULL,
        characters_num int4 NOT NULL DEFAULT 0,
        decimals_num int4 NOT NULL DEFAULT 0,
        char_uom varchar NULL,
        "desc" varchar(100) NOT NULL,
        CONSTRAINT characteristic_pk PRIMARY KEY (char_code, project),
        CONSTRAINT characteristic_uom FOREIGN KEY (project,char_uom) REFERENCES
        ↪  uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.cost_center definition
CREATE TABLE cost_center (
```

```
        project int4 NOT NULL,
        cost_center_code varchar(10) NOT NULL,
        plant varchar(4) NULL,
        currency varchar(5) NULL,
        department varchar(12) NULL,
        tel_num varchar(16) NULL,
        fax_num varchar(31) NULL,
        CONSTRAINT cost_center_pk PRIMARY KEY (project, cost_center_code),
        CONSTRAINT costcenter_currency FOREIGN KEY (project,currency) REFERENCES
        ↪ currency(project,currency) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT costcenter_plant FOREIGN KEY (project,plant) REFERENCES
        ↪ plant(project,plant_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪ DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr."location" definition
CREATE TABLE "location" (
        project int4 NOT NULL,
        location_code varchar(10) NOT NULL,
        "desc" varchar NULL,
        plant varchar(4) NULL,
        CONSTRAINT location_pk PRIMARY KEY (project, location_code),
        CONSTRAINT location_fk FOREIGN KEY (project,plant) REFERENCES
        ↪ plant(project,plant_code) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪ DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.meas_point definition
CREATE TABLE meas_point (
        project int4 NOT NULL,
        "desc" varchar(40) NULL,
        tech_obj_auth_group varchar(4) NULL,
        objectid int8 NOT NULL,
        "char" varchar(30) NOT NULL,
        meas_point_code varchar(20) NOT NULL,
        CONSTRAINT meas_point_pk PRIMARY KEY (project, meas_point_code),
        CONSTRAINT measpoint_authgroup FOREIGN KEY (project,tech_obj_auth_group)
        ↪ REFERENCES tech_obj_auth_group(project,group_code) ON DELETE SET NULL ON
        ↪ UPDATE CASCADE,
        CONSTRAINT measpoint_char FOREIGN KEY ("char",project) REFERENCES
        ↪ characteristic(char_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪ DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.mplann_group definition
CREATE TABLE mplann_group (
        project int4 NOT NULL,
        plant varchar(4) NULL,
        plann_group varchar(3) NOT NULL,
        group_name varchar(18) NULL,
        tel_num varchar(12) NULL,
        email varchar(40) NULL,
        CONSTRAINT mplann_group_pk PRIMARY KEY (project, plann_group),
        CONSTRAINT "mPlanningGroup_planningPlant" FOREIGN KEY (project,plant) REFERENCES
        ↪ plant(project,plant_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪ DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.strategy definition
CREATE TABLE strategy (
        project int4 NOT NULL,
        strategy_code varchar(6) NOT NULL,
        performance_uom varchar(3) NULL,
        frequency_uom varchar(3) NULL,
        tot_cycle_dur1 float8 NULL,
        tot_cycle_dur2 float8 NULL,
        min_step_mpack float8 NULL,
        min_step_mpack_perfbased float8 NULL,
```

```
        "desc" varchar(30) NOT NULL,
        number_of_pack int2 NULL,
        CONSTRAINT strategy_pk PRIMARY KEY (project, strategy_code),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT "strategy_frequencyUoM" FOREIGN KEY (project,frequency_uom) REFERENCES
        ↪  uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "strategy_performanceUoM" FOREIGN KEY (project,performance_uom)
        ↪  REFERENCES uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE
);

-- ar_mr.task_list definition
CREATE TABLE task_list (
        project int4 NOT NULL,
        tasklist_type varchar(1) NOT NULL,
        tasklist_status varchar(3) NULL,
        tasklist_group varchar NOT NULL,
        tasklist_usage varchar(3) NULL,
        planning_plant varchar(4) NULL,
        "desc" varchar(40) NULL,
        delete_indicator varchar(1) NULL,
        group_counter int2 NOT NULL,
        resp_plann_group varchar(3) NULL,
        main_work_center varchar(8) NOT NULL,
        CONSTRAINT task_list_pk PRIMARY KEY (project, tasklist_group, group_counter),
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT task_list_fk FOREIGN KEY (project,resp_plann_group) REFERENCES
        ↪  mplann_group(project,plann_group) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT tasklist_plant FOREIGN KEY (project,planning_plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "tasklist_tasklistGroup" FOREIGN KEY (project,tasklist_group)
        ↪  REFERENCES tasklist_group(project,tasklist_group) ON DELETE RESTRICT ON UPDATE
        ↪  CASCADE,
        CONSTRAINT "tasklist_tasklistStatus" FOREIGN KEY (project,tasklist_status)
        ↪  REFERENCES tasklist_status(project,tasklist_status) ON DELETE RESTRICT ON
        ↪  UPDATE CASCADE,
        CONSTRAINT "tasklist_tasklistType" FOREIGN KEY (project,tasklist_type) REFERENCES
        ↪  tasklist_type(project,tasklist_type) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "tasklist_tasklistUsage" FOREIGN KEY (project,tasklist_usage)
        ↪  REFERENCES tasklist_usage(project,tasklist_usage) ON DELETE RESTRICT ON UPDATE
        ↪  CASCADE,
        CONSTRAINT tasklist_workcenter FOREIGN KEY (project,main_work_center) REFERENCES
        ↪  work_center(project,work_center_code) ON DELETE RESTRICT ON UPDATE CASCADE
);

-- ar_mr.char_to_class definition
CREATE TABLE char_to_class (
        "class" varchar(18) NOT NULL,
        char_code varchar(30) NOT NULL,
        project int4 NOT NULL,
        CONSTRAINT char_to_class_pk PRIMARY KEY (project, char_code, class),
        CONSTRAINT char_fk FOREIGN KEY (char_code,project) REFERENCES
        ↪  characteristic(char_code,project) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT class_fk FOREIGN KEY (project,"class") REFERENCES
        ↪  "class"(project,class_code) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.char_value definition
CREATE TABLE char_value (
        project int4 NOT NULL,
        value varchar NOT NULL,
        objectid int8 NOT NULL,
        char_code varchar(30) NOT NULL,
        CONSTRAINT char_value_pk PRIMARY KEY (project, objectid, char_code),
        CONSTRAINT char_fk FOREIGN KEY (char_code,project) REFERENCES
        ↪  characteristic(char_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);
```

```
-- ar_mr.fun_loc definition
CREATE TABLE fun_loc (
        project int4 NOT NULL,
        floc_id varchar NOT NULL,
        floc_cat varchar(1) NOT NULL,
        sup_floc varchar NULL,
        tech_obj_auth_group varchar(4) NULL,
        equip_install_allowed bool NOT NULL,
        single_equip bool NOT NULL,
        maint_plant varchar(4) NOT NULL,
        plann_group varchar(3) NULL,
        obj_type varchar(10) NULL,
        objectid int8 NOT NULL DEFAULT nextval('ar_mr.objectid_seq'::regclass),
        structure_indicator varchar NOT NULL,
        "desc" varchar NOT NULL,
        work_center varchar(8) NULL,
        user_status varchar(4) NOT NULL,
        criticality varchar(1) NOT NULL,
        effect varchar(3) NOT NULL,
        "location" varchar(10) NOT NULL,
        plann_plant varchar(4) NOT NULL,
        sce varchar(1) NULL,
        cost_center varchar(10) NULL,
        wbs_element varchar(24) NULL,
        "p&id" varchar NULL,
        CONSTRAINT fun_loc_pk PRIMARY KEY (project, floc_id),
        CONSTRAINT fun_loc_un UNIQUE (project, objectid),
        CONSTRAINT floc_authgroup FOREIGN KEY (project,tech_obj_auth_group) REFERENCES
        ↪  tech_obj_auth_group(project,group_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT floc_costcenter FOREIGN KEY (project,cost_center) REFERENCES
        ↪  cost_center(project,cost_center_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_criticality FOREIGN KEY (criticality,project) REFERENCES
        ↪  criticality(criticality_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_effect FOREIGN KEY (effect,project) REFERENCES
        ↪  effect(effect_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_floccategory FOREIGN KEY (project,floc_cat) REFERENCES
        ↪  floc_cat(project,floc_cat) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_location FOREIGN KEY (project,"location") REFERENCES
        ↪  "location"(project,location_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "floc_maintPlant" FOREIGN KEY (project,maint_plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT floc_objtype FOREIGN KEY (project,obj_type) REFERENCES
        ↪  obj_type(project,obj_type) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT floc_plannplant FOREIGN KEY (project,plann_plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_structureindicator FOREIGN KEY (project,structure_indicator)
        ↪  REFERENCES structure_indicator(project,structure_indicator) ON DELETE RESTRICT
        ↪  ON UPDATE CASCADE,
        CONSTRAINT floc_userstatus FOREIGN KEY (user_status,project) REFERENCES
        ↪  user_status(user_status_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_wbselem FOREIGN KEY (project,wbs_element) REFERENCES
        ↪  wbs_elem(project,wbs_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT floc_workcenter FOREIGN KEY (project,work_center) REFERENCES
        ↪  work_center(project,work_center_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT fun_loc_fk FOREIGN KEY (project,plann_group) REFERENCES
        ↪  mplann_group(project,plann_group) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT objectid_fk FOREIGN KEY (objectid) REFERENCES "object"(objectid) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE DEFERRABLE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT sup_floc FOREIGN KEY (project,sup_floc) REFERENCES
        ↪  fun_loc(project,floc_id) ON DELETE CASCADE ON UPDATE CASCADE DEFERRABLE
        ↪  INITIALLY DEFERRED
);

-- Table Triggers

create trigger floc_object_ins before
insert
    on
    ar_mr.fun_loc for each row execute function ar_mr.object_insert();
```

```
create trigger floc_object_del after
delete
    on
    ar_mr.fun_loc for each row execute function ar_mr.object_delete();

-- ar_mr.mplan definition
CREATE TABLE mplan (
        project int4 NOT NULL,
        "desc" varchar(40) NOT NULL,
        strategy varchar(6) NOT NULL,
        sched_period_uom varchar(3) NOT NULL,
        sched_indicator varchar(1) NULL,
        start_count_read varchar(22) NULL,
        call_confirm bool NULL,
        tech_obj_auth_group varchar(4) NULL,
        mplan_cat varchar(2) NULL,
        mplan_id bigserial NOT NULL,
        sf_latecomp int2 NULL,
        sf_earlycomp int2 NULL,
        tol_latecomp int2 NULL,
        tol_earlycomp int2 NULL,
        cycle_mod_fact float4 NULL,
        sched_period int2 NOT NULL,
        call_horiz int2 NOT NULL,
        start_date date NULL,
        mplan_code varchar NULL,
        CONSTRAINT mplan_pk PRIMARY KEY (project, mplan_id),
        CONSTRAINT mplan_un UNIQUE (project, mplan_code),
        CONSTRAINT "mplan_authGroup" FOREIGN KEY (project,tech_obj_auth_group) REFERENCES
        ↪  tech_obj_auth_group(project,group_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT "mplan_mplanCategory" FOREIGN KEY (project,mplan_cat) REFERENCES
        ↪  mplan_cat(project,mplan_cat) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "mplan_schedPeriodUoM" FOREIGN KEY (project,sched_period_uom)
        ↪  REFERENCES uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT mplan_strategy FOREIGN KEY (project,strategy) REFERENCES
        ↪  strategy(project,strategy_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.operation definition
CREATE TABLE operation (
        project int4 NOT NULL,
        op_num int2 NOT NULL,
        "desc" varchar(40) NOT NULL,
        control_key_code varchar(4) NOT NULL,
        sys_condition varchar(1) NOT NULL,
        exec_fact int2 NULL,
        fix_lot_ext_proces bool NULL,
        duration float4 NOT NULL,
        duration_uom varchar(3) NOT NULL,
        work_percentage int2 NULL,
        tasklist_group varchar NOT NULL,
        group_counter int2 NOT NULL,
        work_man_h int4 NULL,
        work_center varchar(8) NOT NULL,
        std_text_code varchar(7) NULL,
        scmop varchar(1) NULL,
        CONSTRAINT operation_pk PRIMARY KEY (project, op_num, tasklist_group,
        ↪  group_counter),
        CONSTRAINT "operation_controlKey" FOREIGN KEY (project,control_key_code)
        ↪  REFERENCES control_key(project,control_key_code) ON DELETE RESTRICT ON UPDATE
        ↪  CASCADE,
        CONSTRAINT operation_stdtext FOREIGN KEY (project,std_text_code) REFERENCES
        ↪  std_text(project,std_text_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "operation_sysCondition" FOREIGN KEY (project,sys_condition) REFERENCES
        ↪  system_cond(project,sys_cond_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT operation_tasklist FOREIGN KEY (project,tasklist_group,group_counter)
        ↪  REFERENCES task_list(project,tasklist_group,group_counter) ON DELETE CASCADE
        ↪  ON UPDATE CASCADE,
        CONSTRAINT "operation_workUoM" FOREIGN KEY (project,duration_uom) REFERENCES
        ↪  uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
```

```
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.package definition
CREATE TABLE package (
        project int4 NOT NULL,
        strategy varchar(6) NOT NULL,
        cycle_freq float8 NULL,
        count_read float8 NULL,
        offset_time float8 NULL,
        offset_perf float8 NULL,
        "desc" varchar(30) NOT NULL,
        counter_uom varchar(10) NULL,
        frequency_uom varchar(10) NULL,
        package_code varchar(2) NOT NULL,
        meas_point varchar(20) NULL,
        CONSTRAINT package_pk PRIMARY KEY (project, strategy, package_code),
        CONSTRAINT couteruom_fk FOREIGN KEY (project,counter_uom) REFERENCES
        ↪  uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT frequencyuom_fk FOREIGN KEY (project,frequency_uom) REFERENCES
        ↪  uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT package_measpoint FOREIGN KEY (project,meas_point) REFERENCES
        ↪  meas_point(project,meas_point_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT package_strategy FOREIGN KEY (project,strategy) REFERENCES
        ↪  strategy(project,strategy_code) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.equipment definition
CREATE TABLE equipment (
        project int4 NOT NULL,
        "desc" varchar NOT NULL,
        tech_id_num varchar(25) NULL,
        tech_obj_auth_group varchar(4) NULL,
        equip_cat varchar(1) NOT NULL,
        obj_type varchar(10) NOT NULL,
        invent_num varchar(25) NULL,
        dimension varchar(18) NULL,
        weight float8 NULL,
        weight_uom varchar(3) NULL,
        acquisition_date date NULL,
        acquisition_value float8 NULL,
        currency varchar(5) NULL,
        manufacturer varchar(30) NOT NULL,
        manu_country varchar(3) NULL,
        serial_num varchar(30) NOT NULL,
        model_num varchar(20) NOT NULL,
        startup_date date NULL,
        license_num varchar(20) NULL,
        user_status varchar(4) NOT NULL,
        floc varchar NULL,
        criticality varchar(1) NOT NULL,
        equip_tag_num varchar(30) NOT NULL,
        manu_part_num varchar(30) NULL,
        constr_type varchar(18) NULL,
        effect varchar(3) NOT NULL,
        catalog_prof varchar(9) NULL,
        asset varchar(12) NULL,
        vendor varchar(20) NULL,
        constr_yy int2 NULL,
        constr_mm int2 NULL,
        objectid int8 NOT NULL DEFAULT nextval('ar_mr.objectid_seq'::regclass),
        equip_xlsx_code varchar NOT NULL,
        sup_equip int8 NULL,
        CONSTRAINT equipment_pk PRIMARY KEY (project, objectid),
        CONSTRAINT equipment_un UNIQUE (equip_xlsx_code, project),
        CONSTRAINT "equipment_acquisitionCurrency" FOREIGN KEY (project,currency)
        ↪  REFERENCES currency(project,currency) ON DELETE RESTRICT ON UPDATE CASCADE,
```

```
        CONSTRAINT equipment_authgroup FOREIGN KEY (project,tech_obj_auth_group)
        ↪  REFERENCES tech_obj_auth_group(project,group_code) ON DELETE SET NULL ON
        ↪  UPDATE CASCADE,
        CONSTRAINT equipment_catalogprof FOREIGN KEY (project,catalog_prof) REFERENCES
        ↪  catalog_profile(project,catalog_code) ON DELETE RESTRICT ON UPDATE RESTRICT,
        CONSTRAINT equipment_criticality FOREIGN KEY (criticality,project) REFERENCES
        ↪  criticality(criticality_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT equipment_effect FOREIGN KEY (effect,project) REFERENCES
        ↪  effect(effect_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT equipment_equipcategory FOREIGN KEY (project,equip_cat) REFERENCES
        ↪  equip_cat(project,equip_cat) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT equipment_floc FOREIGN KEY (project,floc) REFERENCES
        ↪  fun_loc(project,floc_id) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT equipment_manucountry FOREIGN KEY (manu_country,project) REFERENCES
        ↪  country(country_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT equipment_objtype FOREIGN KEY (project,obj_type) REFERENCES
        ↪  obj_type(project,obj_type) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT equipment_supequip FOREIGN KEY (project,sup_equip) REFERENCES
        ↪  equipment(project,objectid) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT equipment_userstatus FOREIGN KEY (user_status,project) REFERENCES
        ↪  user_status(user_status_code,project) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "equipment_weightUoM" FOREIGN KEY (project,weight_uom) REFERENCES
        ↪  uom(project,uom_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- Table Triggers

create trigger equip_object_ins before
insert
    on
    ar_mr.equipment for each row execute function ar_mr.object_insert();
create trigger equip_object_del after
delete
    on
    ar_mr.equipment for each row execute function ar_mr.object_delete();

-- ar_mr.mitem definition
CREATE TABLE mitem (
        project int4 NOT NULL,
        "desc" varchar(40) NOT NULL,
        status varchar(1) NULL,
        plant varchar(4) NOT NULL,
        order_type varchar(4) NOT NULL,
        m_activity_type varchar(3) NULL,
        mitem_cat varchar(2) NULL,
        priority varchar(5) NOT NULL,
        not_release_imm bool NULL,
        task_list_fact int2 NULL,
        mitem_id bigserial NOT NULL,
        mplan int8 NOT NULL,
        tasklist_group varchar NOT NULL,
        group_counter int2 NOT NULL,
        work_center varchar(8) NULL,
        plann_group varchar(3) NULL,
        mitem_code varchar NULL,
        ref_equip int8 NULL,
        ref_floc varchar NULL,
        CONSTRAINT mitem_pk PRIMARY KEY (project, mitem_id),
        CONSTRAINT mitem_un UNIQUE (project, mitem_code),
        CONSTRAINT mitem_mactivitytype FOREIGN KEY (project,m_activity_type) REFERENCES
        ↪  m_activity_type(project,act_type_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT "mitem_mitemCategory" FOREIGN KEY (project,mitem_cat) REFERENCES
        ↪  mitem_cat(project,mitem_cat) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "mitem_orderType" FOREIGN KEY (project,order_type) REFERENCES
        ↪  order_type(project,order_type) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT mitem_planngroup FOREIGN KEY (project,plann_group) REFERENCES
        ↪  mplann_group(project,plann_group) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT "mitem_planningPlant" FOREIGN KEY (project,plant) REFERENCES
        ↪  plant(project,plant_code) ON DELETE RESTRICT ON UPDATE CASCADE,
```

```
        CONSTRAINT mitem_priority FOREIGN KEY (project,priority) REFERENCES
        ↪  priority(project,priority_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT mitem_ref_equip FOREIGN KEY (project,ref_equip) REFERENCES
        ↪  equipment(project,objectid) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT mitem_ref_floc FOREIGN KEY (project,ref_floc) REFERENCES
        ↪  fun_loc(project,floc_id) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT mitem_tasklist FOREIGN KEY (project,tasklist_group,group_counter)
        ↪  REFERENCES task_list(project,tasklist_group,group_counter) ON DELETE RESTRICT
        ↪  ON UPDATE CASCADE,
        CONSTRAINT mitem_workcenter FOREIGN KEY (project,work_center) REFERENCES
        ↪  work_center(project,work_center_code) ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.obj_list_element definition
CREATE TABLE obj_list_element (
        project int4 NOT NULL,
        mitem int8 NOT NULL,
        objectid int8 NOT NULL,
        CONSTRAINT obj_list_element_pk PRIMARY KEY (project, mitem, objectid),
        CONSTRAINT objlist_mitem FOREIGN KEY (project,mitem) REFERENCES
        ↪  mitem(project,mitem_id) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.operation_package definition
CREATE TABLE operation_package (
        project int4 NOT NULL,
        op_num int2 NOT NULL,
        strategy varchar(6) NOT NULL,
        tasklist_group varchar NOT NULL,
        group_counter int2 NOT NULL,
        package_code varchar(2) NOT NULL,
        CONSTRAINT operation_package_pk PRIMARY KEY (project, op_num, strategy,
        ↪  tasklist_group, group_counter, package_code),
        CONSTRAINT operation_fk FOREIGN KEY (project,op_num,tasklist_group,group_counter)
        ↪  REFERENCES operation(project,op_num,tasklist_group,group_counter) ON DELETE
        ↪  CASCADE ON UPDATE CASCADE,
        CONSTRAINT package_fk FOREIGN KEY (project,strategy,package_code) REFERENCES
        ↪  package(project,strategy,package_code) ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);

-- ar_mr.assembly_equip definition
CREATE TABLE assembly_equip (
        project int4 NOT NULL,
        assemblyid int8 NOT NULL,
        equipid int8 NOT NULL,
        CONSTRAINT assembly_equip_pk PRIMARY KEY (project, assemblyid, equipid),
        CONSTRAINT assembly_equip FOREIGN KEY (project,equipid) REFERENCES
        ↪  equipment(project,objectid) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT assembly_fk FOREIGN KEY (project,assemblyid) REFERENCES
        ↪  assembly(project,objectid) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT project_fk FOREIGN KEY (project) REFERENCES project(project_id) ON
        ↪  DELETE CASCADE ON UPDATE CASCADE
);
```

CAPITOLO 5

Sviluppo dell'interfaccia CRUD

In questo capitolo verrà analizzato il risultato dello sviluppo dell'interfaccia CRUD. In un primo momento, verrà fatto un riassunto delle ragioni che stanno alla base della realizzazione di un software di questo tipo, spiegando, quindi, quali sono i principali vantaggi che questo tool apporterà. Successivamente, si dedicherà una sezione all'approfondimento del software utilizzato per lo sviluppo dell'interfaccia, ovvero SQL Maestro PHP Generator. Una volta illustrate le principali features di questo strumento, si passerà a parlare del risultato prodotto dalla fase di sviluppo; in particolare, verranno descritte la struttura portante e le pagine principali che compongono l'interfaccia CRUD. Per concludere il capitolo, verranno descritti alcuni possibili sviluppi futuri che si prevede verranno implementati.

5.1 Introduzione allo sviluppo

Nel precedente capitolo della tesi si è parlato in modo molto approfondito, di come, partendo dalla mappatura interna di SAP PM, si è arrivati alla forma finale del database; come già detto in molte occasioni, questa struttura dati sarà la backbone di tutto il progetto, fornendo una rappresentazione stabile e strutturalmente solida per i dati aziendali.

Detto ciò, nelle prossime pagine, si passerà ad introdurre il *sistema CRUD* sviluppato sulla base del database appena descritto. Come già detto durante l'introduzione della tesi, un sistema CRUD costituisce la base per la gestione dei dati relativi alle applicazioni, fornendo un modo organizzato e strutturato per creare, leggere, aggiornare e cancellare informazioni.

Nel descrivere il sistema CRUD si procederà seguendo lo stesso approccio usato per lo studio del database; quindi, in primis, verranno descritti i principali obiettivi da far raggiungere al sistema, per poi passare ad illustrare gli strumenti software utilizzati durante lo sviluppo. Infine, si analizzerà la struttura dell'interfaccia prodotta, cercando di motivare le scelte fatte in fase di progettazione.

5.1.1 Obiettivi del sistema CRUD

Un sistema CRUD rappresenta, di fatto, un punto di accesso per la gestione delle informazioni presenti in una base di dati. Come già detto precedentemente, la sigla CRUD indica le quattro operazioni fondamentali, ovvero creazione (Create), lettura (Read), aggiornamento (Update) ed eliminazione (Delete), che un sistema di questo tipo deve garantire agli utenti finali del database.

Più nel dettaglio, gli obiettivi principali da raggiungere, sono i seguenti:

- *Facilitare la creazione di dati*: semplificare il processo di aggiunta di nuovi dati al sistema, consentendo agli utenti di creare record o inserire informazioni senza difficoltà, ma soprattutto, in modo intuitivo.
- *Garantire un accesso rapido ed efficiente*: fornire un meccanismo veloce ed efficiente per recuperare informazioni esistenti dal sistema. La lettura dei dati dovrebbe essere ottimizzata per garantire risposte tempestive alle richieste degli utenti.
- *Semplicità di aggiornamento*: consentire agli utenti di apportare modifiche ai dati esistenti in modo semplice e sicuro. L'aggiornamento dovrebbe essere gestito in modo da mantenere l'integrità dei dati e garantire che le modifiche siano riflesse correttamente nel sistema. Tutto ciò, sarà, in gran parte, garantito dalla struttura sottostante del database.
- *Gestione dei dati obsoleti*: fornire strumenti per la cancellazione o l'archiviazione sicura di dati obsoleti o non più necessari. Ciò contribuisce a mantenere pulito e ordinato il database, migliorando le prestazioni complessive del sistema.
- *Integrità dei dati*: garantire che i dati siano accurati, coerenti e conformi a eventuali vincoli o regole definite nel sistema. La gestione dell'integrità dei dati è cruciale per evitare errori e garantire la fiducia negli output del sistema. Ancora una volta, questo obiettivo è gia stato, in parte, raggiunto dalla presenza di un database ben strutturato.

Al fine di comprendere pienamente i miglioramenti che un CRUD di questo tipo potrebbe apportare a livello di gestione interna delle informazioni, è necessario porre questa soluzione a confronto con il metodo utilizzato attualmente per interagire con i dati in Pansoinco.

Come detto nella sezione riguardante lo stato dell'arte aziendale, attualmente, la maggior parte delle operazioni sui dati passa per Excel. Rispetto a quest'ultimo approccio, il vantaggio principale che l'uso di un sistema CRUD potrebbe fornire, è relativo alla gestione strutturata dei dati; infatti, un'interfaccia CRUD è progettata per gestire i dati in modo strutturato, organizzato e relazionato, il che risulta ancora più rilevante, quando ci si trova ad operare con dati complessi o relazionati fra loro. Per di più, il sistema CRUD può essere sfruttato in un contesto multiutente, consentendo a più persone di accedere contemporaneamente ai dati in modo sicuro e controllato.

D'altra parte, abbandonando Excel, si rinuncia anche ad una serie di facilitazioni. In particolare, Excel è noto per la sua facilità d'uso e familiarità per molti utenti; la sua interfaccia spreadsheet è intuitiva e richiede poca formazione, rendendolo uno strumento estremamente flessibile e adatto per una vasta gamma di compiti. Fra questi si trovano l'analisi dei dati, la creazione di grafici e la realizzazione di calcoli complessi; per questa ragione, il CRUD dovrà essere in grado di sopperire a queste eventuali mancanze con i vantaggi precedentemente introdotti.

5.2 Sviluppo

Ora che sono stati descritti gli obiettivi chiave e le principali ragioni che hanno condotto alla decisione di includere un sistema CRUD come parte del progetto, il focus della discussione si sposterà sullo sviluppo vero e proprio di questo software. Come già accennato, in un primo momento ci si concentrerà sugli strumenti utilizzati per lo sviluppo, per poi passare all'analisi del risultato ottenuto a seguito delle fasi di progettazione e realizzazione.

5.2.1 Strumenti utilizzati

Nel caso del sistema CRUD in questione, questa sezione, riguardante gli strumenti utilizzati, assume un'importanza ancora maggiore. Questo perché il software scelto per realizzare l'applicazione ha facilitato enormemente lo sviluppo dell'interfaccia.

Il programma di cui si sta parlando è *SQL Maestro PHP Generator*. Si tratta di uno strumento software progettato per semplificare lo sviluppo di applicazioni web PHP basate su database. Il software è progettato principalmente per interagire con database relazionali e offre una serie di funzionalità per automatizzare la creazione di codice PHP per l'accesso ai dati, la loro manipolazione e la loro visualizzazione. Il software è progettato per lavorare con una varietà di DBMS relazionali, tra cui MySQL, PostgreSQL, SQLite, Microsoft SQL Server e altri; tutto ciò consente di utilizzare il generatore con diversi sistemi di gestione dei database anche nel caso in cui si abbia a che fare con schemi complessi con molte tabelle correlate. SQL Maestro PHP Generator offre un'interfaccia utente intuitiva che semplifica il processo di progettazione delle pagine web e delle interazioni con il database. Agli utenti è fornita un'ampia gamma di metodi finalizzati a personalizzare l'aspetto ed il comportamento delle pagine generate. Oltre a ciò, SQL Maestro PHP Generator include funzionalità di sicurezza per controllare l'accesso alle pagine generate, dando la possibilità agli utenti di definire ruoli e autorizzazioni per garantire che solo chi è autorizzato possa accedere a determinate funzionalità.

Per lo sviluppo del sistema CRUD in questione, l'azienda ha messo a disposizione la licenza "Pro" del programma, dando la possibilità di accedere ad un numero ancora maggiore di funzionalità.

Appena avviato, il software chiede di collegare un database dal quale accedere alle tabelle sulle quali poi verranno basate le pagine dell'applicazione; come DBeaver, anche PHP Generator consente il collegamento al database tramite SSH Tunneling. Quindi, una volta forniti tutti i parametri di connessione, la prima schermata che il programma presenta è quella riportata in Figura 5.1.

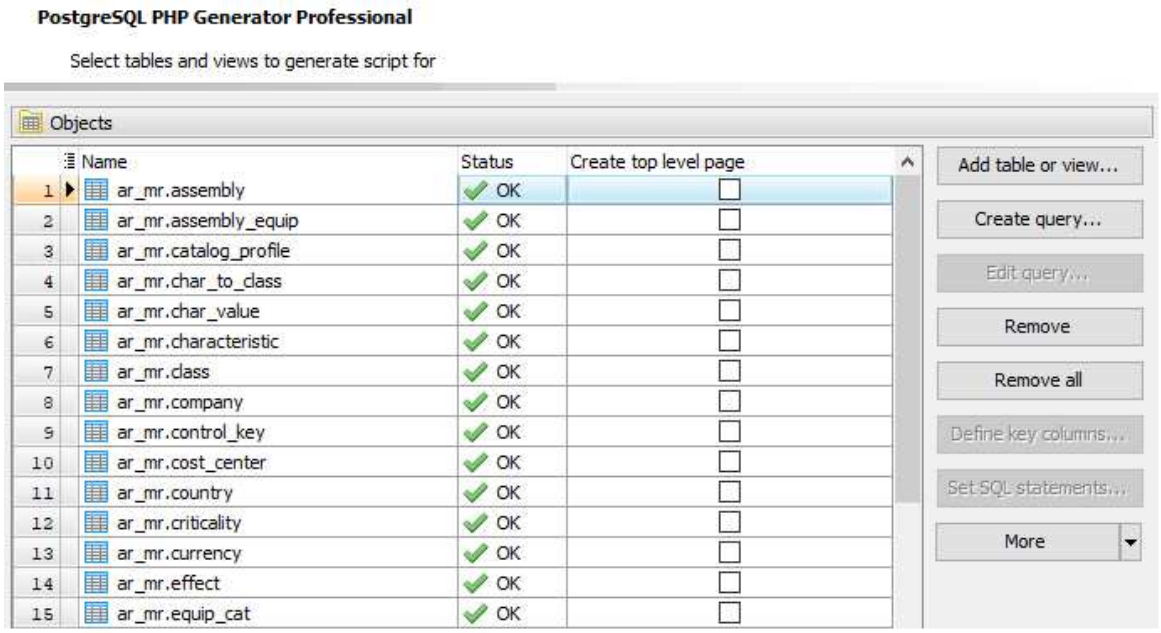
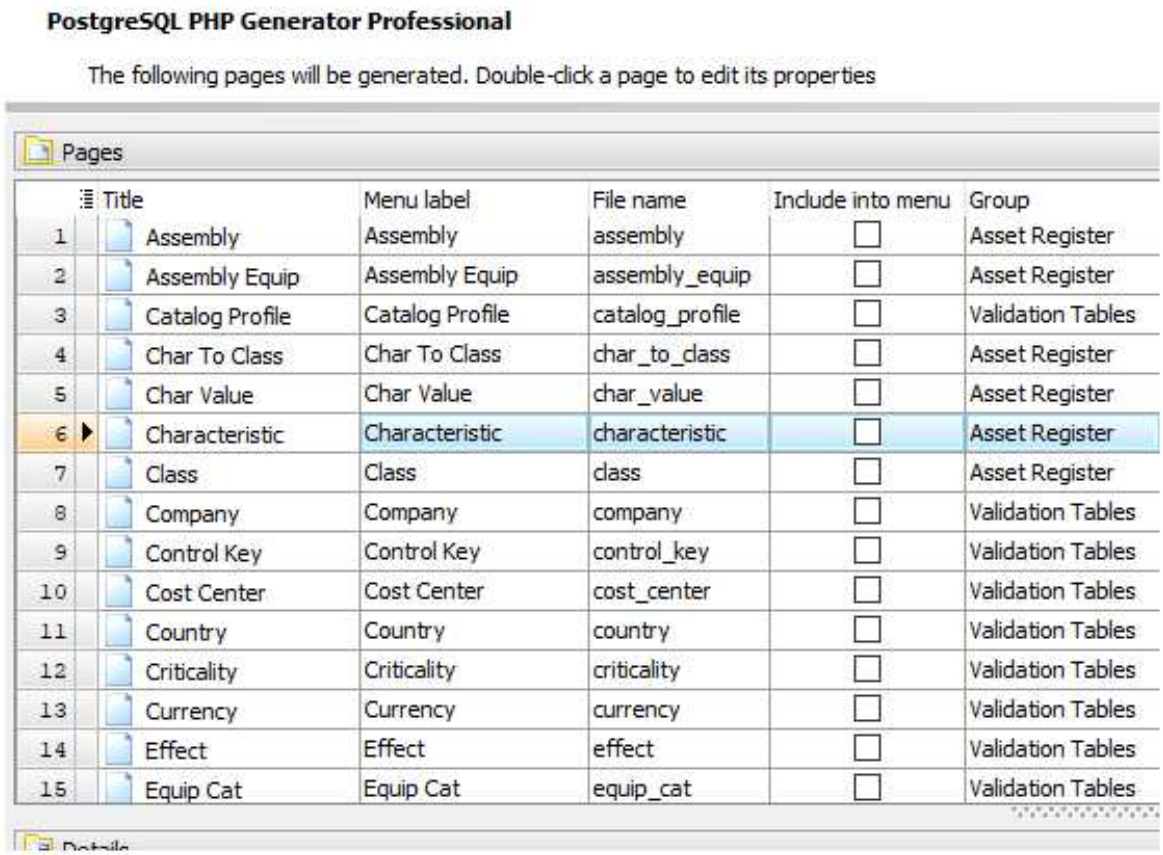


Figura 5.1: PHP Generator - Selezione delle tabelle

In questa finestra, il software chiede di scegliere quali sono fra le tabelle presenti nello schema, quelle per cui dovranno essere generate delle pagine PHP. Una volta selezionate

tutte le tabelle (tutte le informazioni dovranno poter essere accedute), e una volta dato l'ok, la schermata successiva (Figura 5.2) già permette di entrare nello specifico delle configurazioni di ogni pagina. Per ognuna di esse si può scegliere se includerla nel menù iniziale dell'applicazione e se assegnare un gruppo logico di appartenenza. All'avvio, quindi, l'applicazione mostrerà un menù di accesso alla pagine (tabelle) per le quali è stata spuntata la checkbox "Include into menu", divise per gruppi. Seguendo la logica alla base del database, sono stati definiti tre gruppi logici a cui ricondurre ogni tabella, ovvero*Asset Register*, *Maintenance Register* e *Validation Tables*.



	Title	Menu label	File name	Include into menu	Group
1	Assembly	Assembly	assembly	☐	Asset Register
2	Assembly Equip	Assembly Equip	assembly_equip	☐	Asset Register
3	Catalog Profile	Catalog Profile	catalog_profile	☐	Validation Tables
4	Char To Class	Char To Class	char_to_class	☐	Asset Register
5	Char Value	Char Value	char_value	☐	Asset Register
6	Characteristic	Characteristic	characteristic	☐	Asset Register
7	Class	Class	class	☐	Asset Register
8	Company	Company	company	☐	Validation Tables
9	Control Key	Control Key	control_key	☐	Validation Tables
10	Cost Center	Cost Center	cost_center	☐	Validation Tables
11	Country	Country	country	☐	Validation Tables
12	Criticality	Criticality	criticality	☐	Validation Tables
13	Currency	Currency	currency	☐	Validation Tables
14	Effect	Effect	effect	☐	Validation Tables
15	Equip Cat	Equip Cat	equip_cat	☐	Validation Tables

Figura 5.2: PHP Generator - Vista relativa alle pagine

Entrando nelle proprietà di ogni pagina verranno mostrati tutti i campi della tabella associata e una serie di checkbox per ognuno di essi (Figura 5.3). Queste caselle rappresentano le azioni da poter associare ad ogni colonna (campo) della tabella; queste ultime includono, ad esempio, l'inserimento, la modifica, la visualizzazione, il filtraggio, e altri ancora.

La strutturazione dell'applicazione in pagine e sottopagine è garantita dalla possibilità di definire, per ognuna di esse, una sezione "*Details*".

In poche parole, associando una tabella B nella sezione details di una tabella A, per ogni record di A mostrato sul video, sarà possibile visualizzare tutti gli elementi di B associati (tramite chiave esterne). Ad esempio, se si pensa alla relazione fra task list ed operation, associando la tabella operation nei details della tabella task list, per ogni ciclo sarà possibile visualizzare le operazioni ad esso collegate. In Figura 5.4 è possibile osservare il collegamento fra la tabella principale (task list) e la tabella dettaglio (operation) tramite chiave esterna, in aggiunta al risultato sulla pagina dell'applicazione.

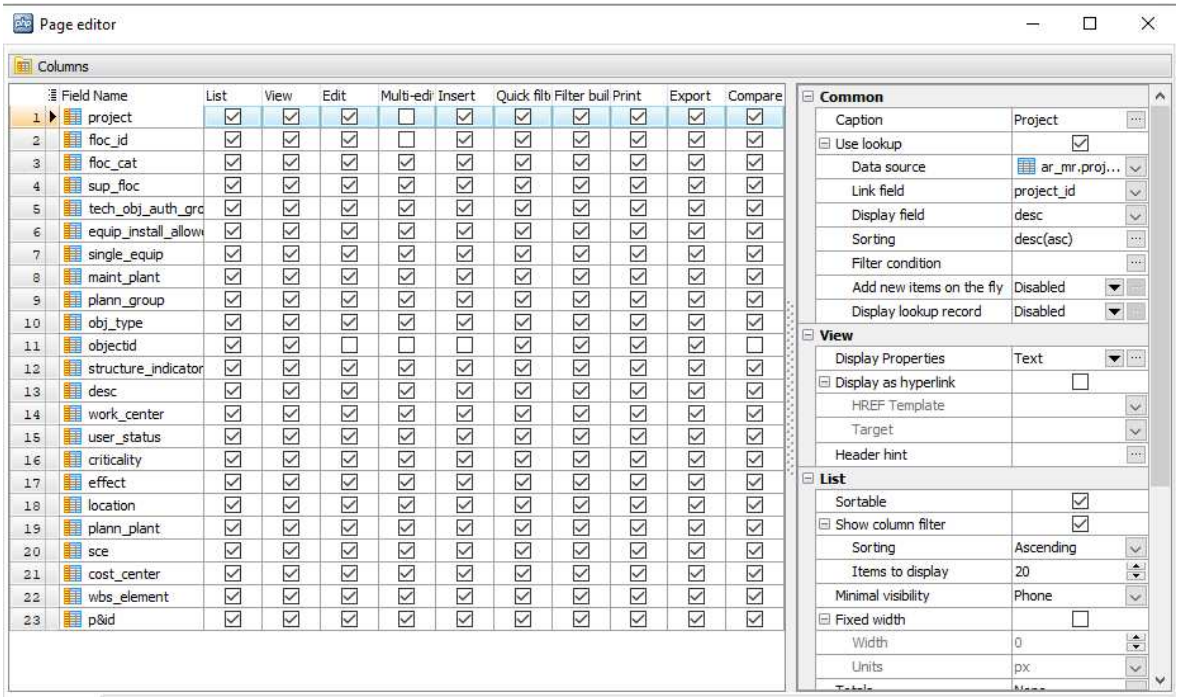
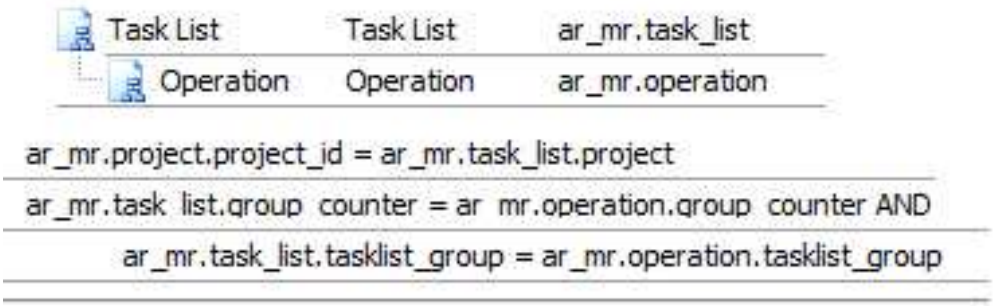


Figura 5.3: PHP Generator - Proprietà della pagina



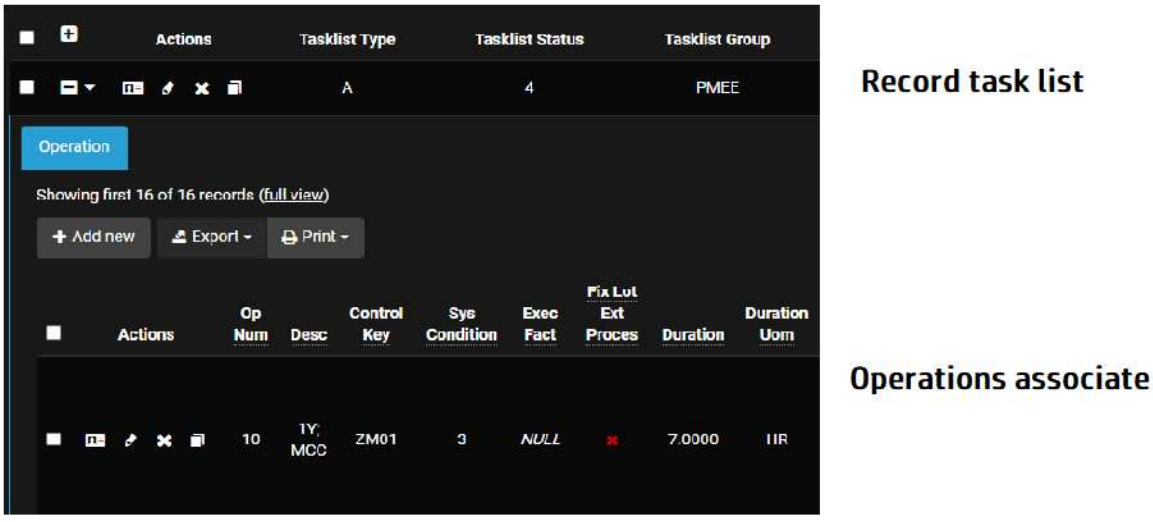


Figura 5.4: PHP Generator - Details

5.2.2 Struttura e pagine

Si passerà, ora, a fornire una descrizione dettagliata dell'interfaccia CRUD realizzata. Il focus verrà posto sulla struttura del sistema e sulle varie pagine che lo compongono. Il sistema CRUD è raggiungibile collegandosi al server aziendale, dove si trova la web application creata usando PHP Generator.

La struttura del sistema CRUD ricalca la divisione logica in Asset e Maintenance register, mantenendo, a livello più alto, la ripartizione per progetto. Infatti, la prima pagina che si presenta, una volta che ci si collega all'applicazione, è quella relativa alla scelta del progetto (Figura 5.5). In essa verranno mostrati tutti i progetti presenti nel sistema, accompagnati da una breve descrizione. Da questa prima pagina è anche possibile eseguire operazioni di modifica, inserimento ed eliminazione per ogni progetto.

Per consentire l'accesso alle informazioni relative ai vari progetti, è stata sfruttata la funzionalità "Details". Scelto un progetto, accedendo a questa sezione, verrà presentato l'elenco degli elementi principali che lo caratterizzano (Figura 5.6), ovvero:

- assembly;
- equipment;
- functional location;
- measuring point;
- maintenance item;
- maintenance plan;
- strategy;
- task list.

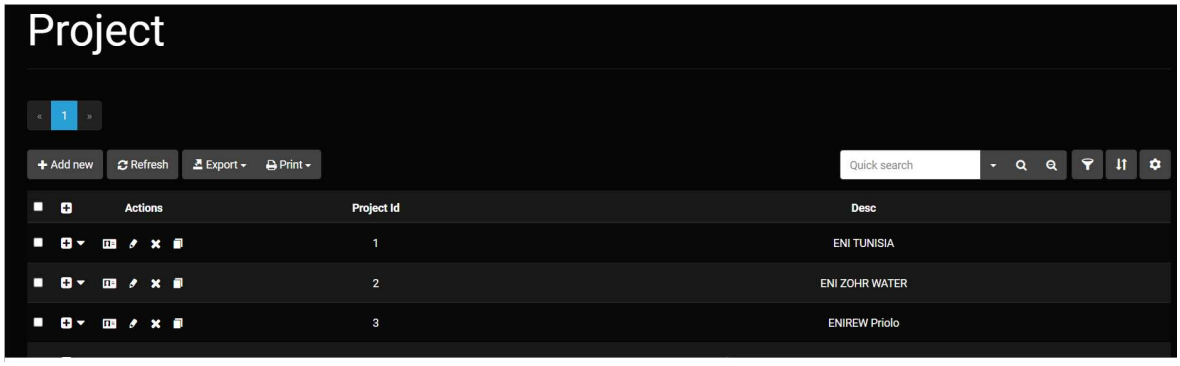


Figura 5.5: Pagina di selezione dei progetti

Accedendo a queste sezioni si potrà interagire con i record relativi all'elemento in questione. Ad esempio, accedendo ad equipment (Figura 5.7), potranno essere eseguite operazioni di visualizzazione, modifica, aggiunta ed eliminazione sui valori dei campi. Nel caso di campi chiave autogenerati dal sistema (ad esempio, equipment, maintenance plan, maintenance item, etc.) questi non potranno essere modificati dall'utente; questo perché, così facendo, si rischierebbe di generare conflitti in fase di generazione delle nuove chiavi.

Le tabelle che, in fase di generazione dell'applicazione, sono state classificate come validation, non appaiono fra gli elementi caratteristici del progetto. Infatti, essendo esse

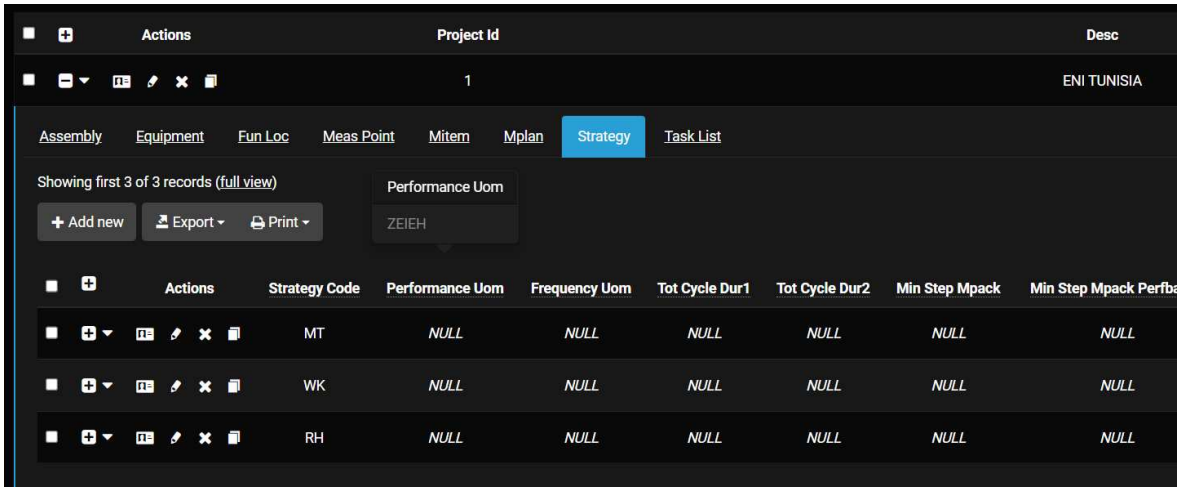


Figura 5.6: Pagina details relativa ad un progetto

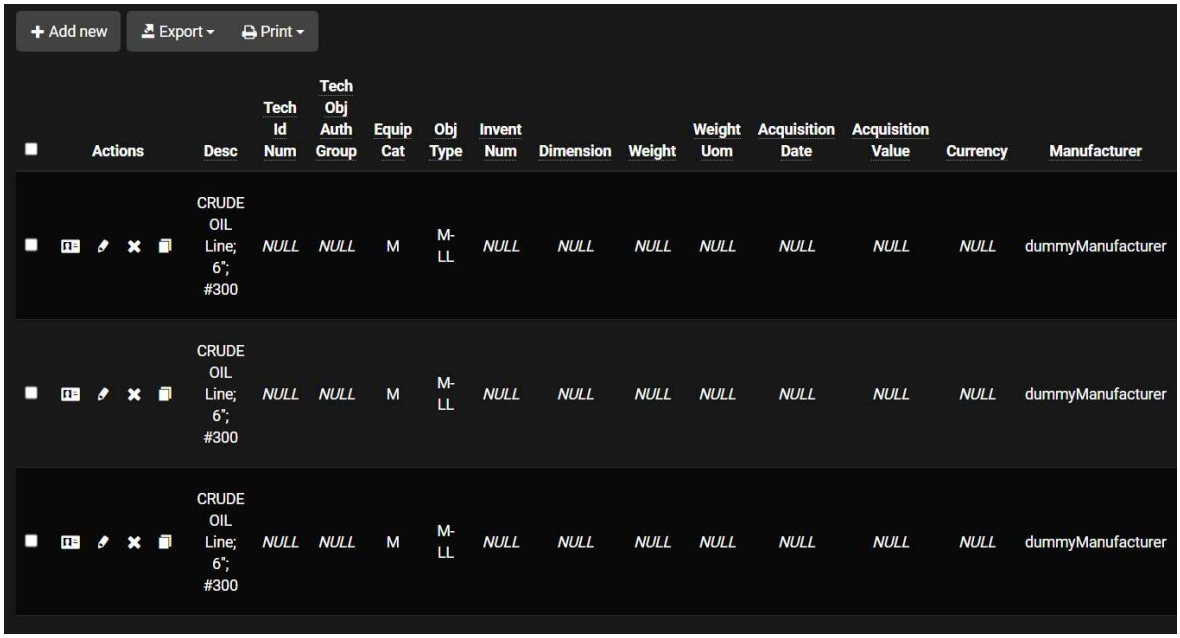


Figura 5.7: Pagina relativa agli equipment

delle tabelle di verifica, servono solo per restringere i possibili valori dei campi associati ad un determinato range. In fase di generazione del codice, è possibile associare ad ogni campo una tabella di *look-up*; facendo questo, si restringeranno i valori associabili al campo in questione ai soli presenti nella tabella indicata. Sempre prendendo come esempio equipment, se si prova ad interagire con il campo "Object Type", si noterà subito che si ha un insieme ristretto di valori fra cui scegliere (Figura 5.8). Lo stesso vale per tutti gli altri campi ai quali è associabile una validation table (ad esempio UoM, country, plant, etc.)

Per quanto riguarda le sezioni details delle tabelle assembly e functional location, in esse è possibile trovare, rispettivamente, gli equipment che compongono l'assembly e quelli che sono ricollegabili alla sede tecnica in questione.

Spostando, ora, il focus sulle interfacce relative agli elementi del maintenance register, si inizierà la discussione concentrandosi su Strategy. Accedendo a questa sezione, per un dato progetto, si potranno inserire, modificare ed eliminare informazioni relative alle strategie di progetto. Parlando dei campi presenti in strategy, a livello di database, in futuro, dovrebbe

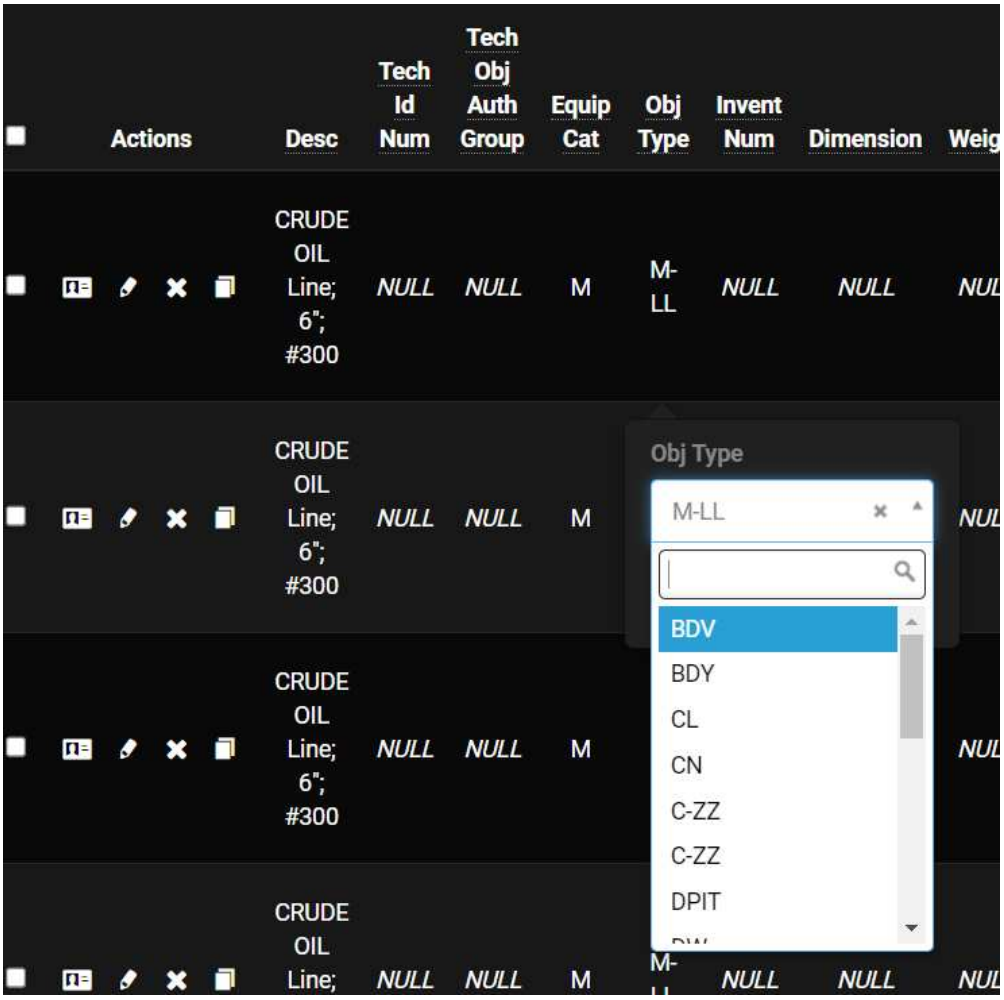


Figura 5.8: Editing del campo object type relativo all'equipment

essere implementato un automatismo in grado di eseguire dei controlli di validazione e delle assegnazioni automatiche sui campi riguardanti il numero di pacchetti nella strategia e lo step minimo fra le frequenze di due pacchetti. Una volta implementato un codice in grado di svolgere automaticamente queste azioni, i campi in questione potrebbero essere modificati in modo da non poter essere modificati manualmente. In Figura 5.9 è possibile osservare la visualizzazione di una strategia ed alcuni suoi pacchetti associati (presenti fra i details).

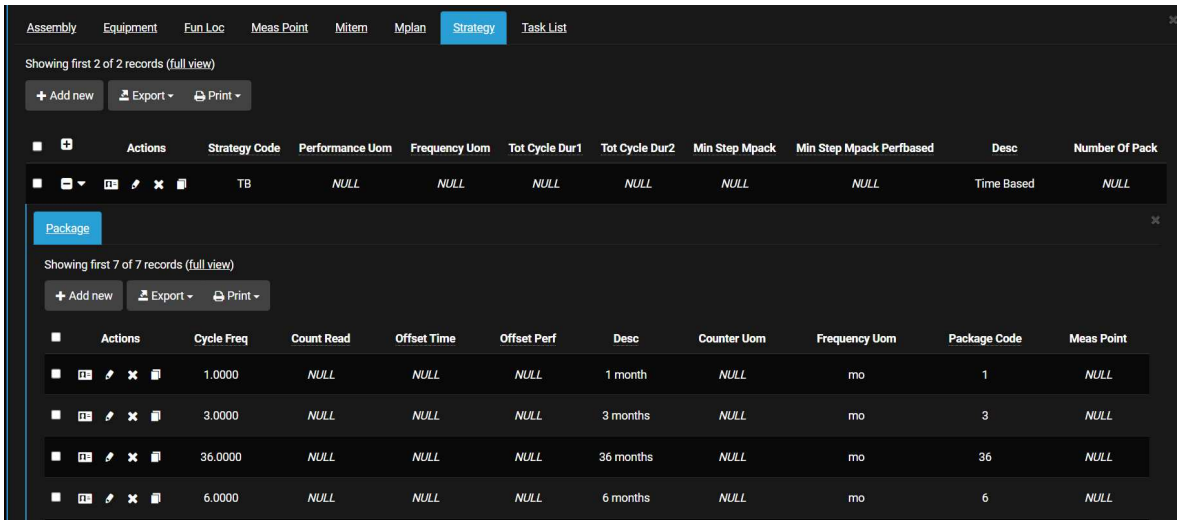


Figura 5.9: Interfaccia con le strategie di manutenzione

Spostando, ora, l'attenzione sulle interfacce riguardanti i maintenance items e i maintenance plan, come per tutti gli altri elementi, anche qui è possibile effettuare operazioni di modifica, eliminazione e creazione. È stato scelto di non rappresentare le posizioni di manutenzione solo come sottoelemento (nella sezione details) dell'interfaccia relativa ai piani; questo perché, data l'importanza associata alle posizioni di manutenzione, è stato reputato necessario includere un elenco a se stante; detto ciò, nella sezione details dei vari piani di manutenzione, è comunque possibile osservare le posizioni legate al piano specifico.

Nell'interfaccia dedicata ai maintenance item, accedendo alla sezione details, è possibile osservare sia le task list ad essi associati, sia la lista completa degli oggetti facenti parte della object list (Figure 5.10 e 5.11).

Focalizzando, ora, l'attenzione sull'interfaccia dedicata ai maintenance plan, come accennato in precedenza, oltre a poter effettuare le generiche operazioni CRUD, è possibile, tramite la sezione details, consultare le posizioni di manutenzione associate ad ogni piano presente nel progetto (Figura 5.12).

Per concludere la trattazione inerente alle diverse interfacce presenti sistema CRUD, si descriverà la sezione dedicata alle task list. Come nel caso di maintenance item e plan, anche qui si è deciso di dedicare una sezione a sé stante atta ad elencare tutti i cicli definiti nell'ambito di un dato progetto. Come al solito, sono garantite le operazioni di modifica, eliminazione e inserimento dati sui campi caratteristici dei vari cicli; oltre a questo, accedendo alla sezione details della pagina, è possibile trovare l'elenco di tutte le operazioni di manutenzione definite all'interno della tasklist in questione (Figura 5.13). Da notare che, per il campo standard text relativo ad operation, è stato deciso di riportare direttamento il valore del campo text, e non l'id.

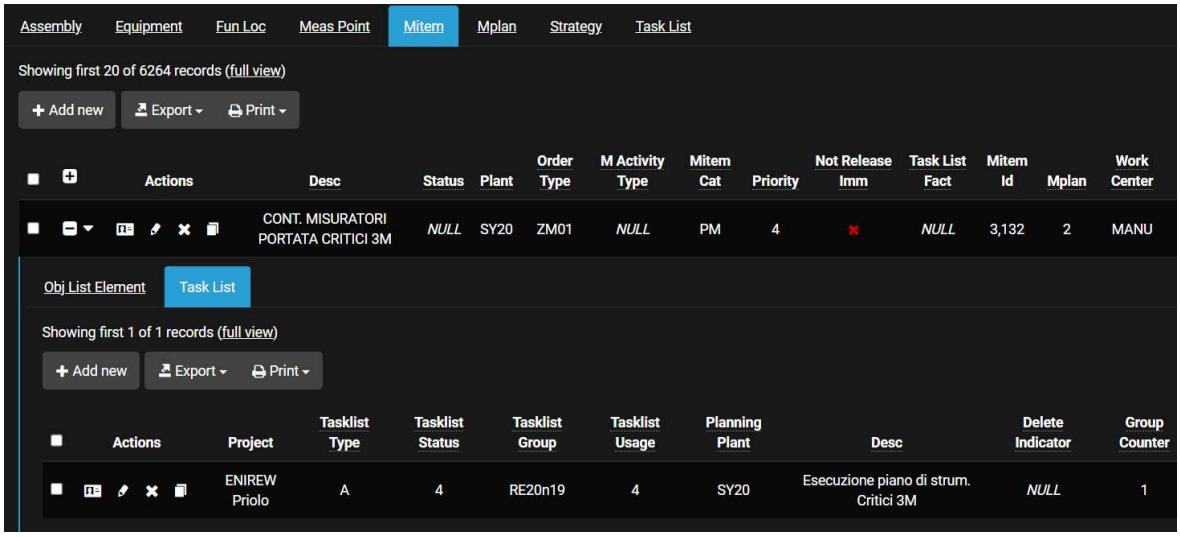


Figura 5.10: Interfaccia con le TaskList

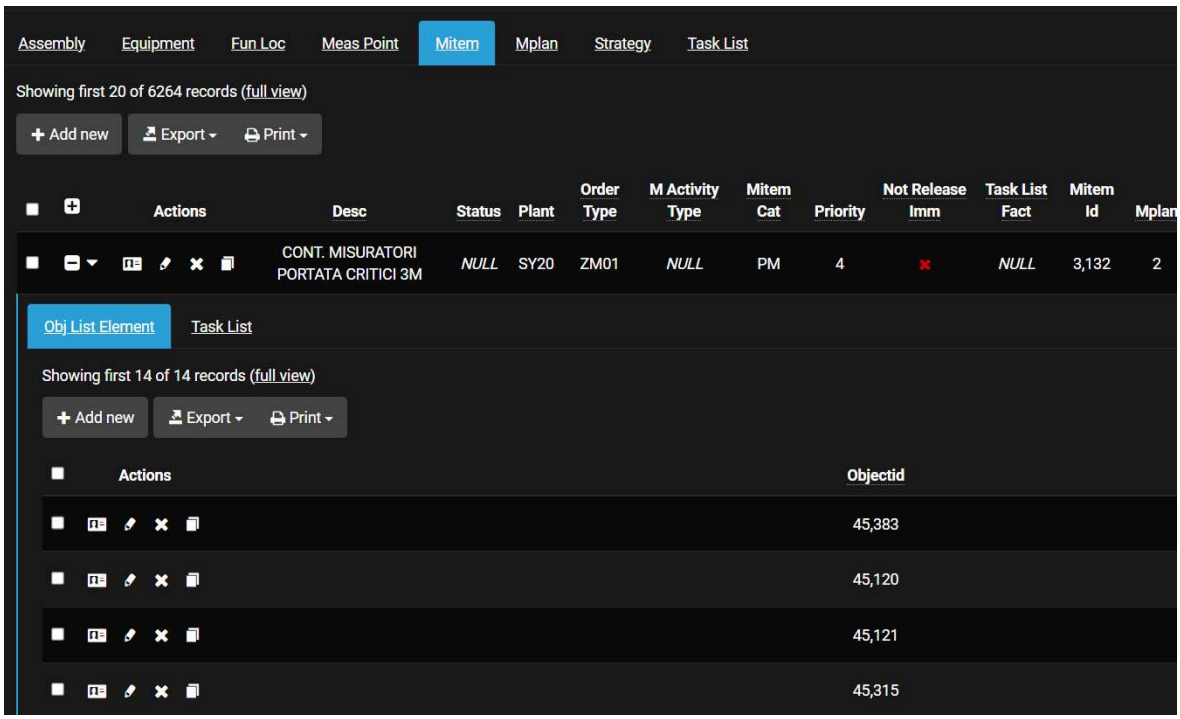


Figura 5.11: Interfaccia con i Maintenance Items

5.3 Possibili sviluppi futuri

Per concludere l'approfondimento della sezione riguardante l'interfaccia CRUD sviluppata, si descriveranno alcuni degli sviluppi futuri che potrebbero essere inclusi nelle prossime versioni del software.

Già da ora, è prevista l'implementazione di un *sistema di autenticazione*, atto a potenziare la sicurezza e proteggere i dati sensibili aziendali. Conoscendo l'importanza delle informazioni presenti nel sistema, è stato chiaro, fin da subito, che si sarebbe presentata la necessità di adottare misure di sicurezza più avanzate.

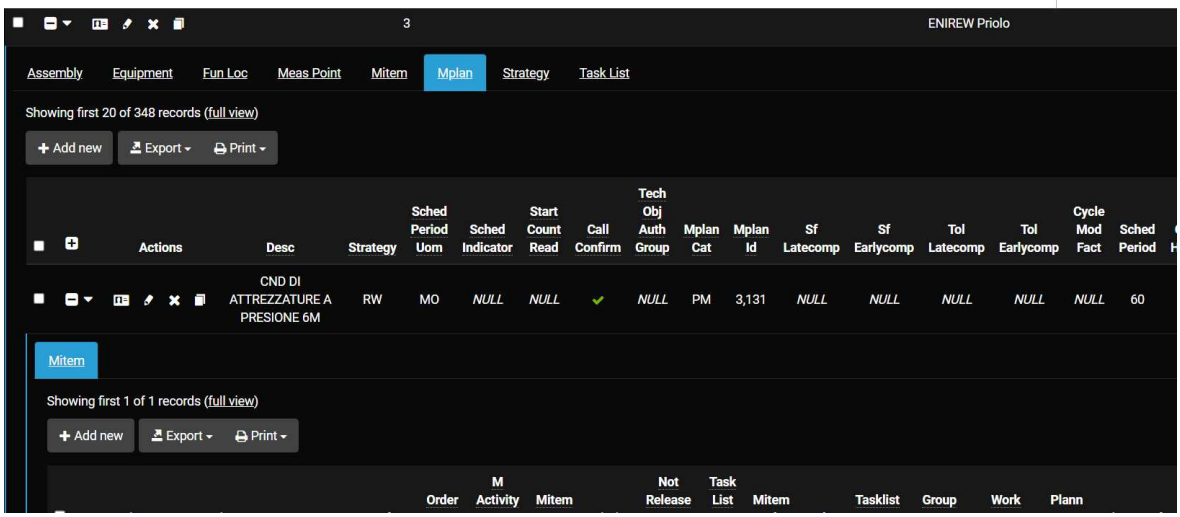


Figura 5.12: Interfaccia con i Maintenance Plan

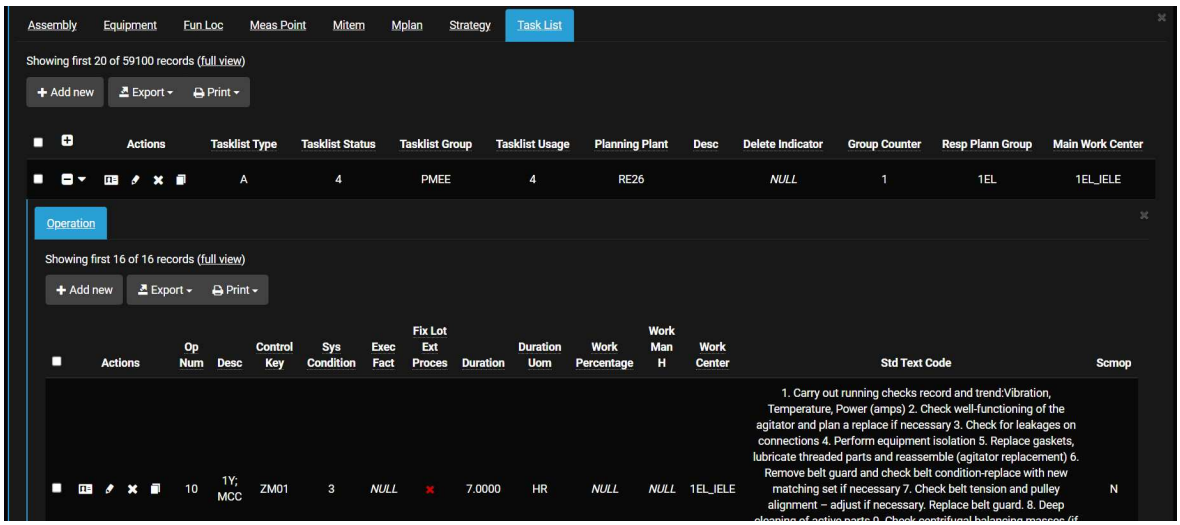


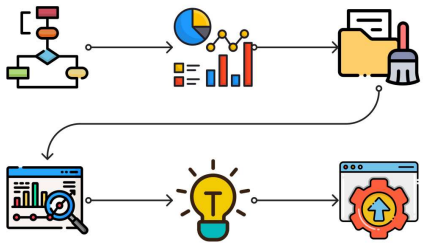
Figura 5.13: Interfaccia con le operazioni relative ad una TaskList

L'eventuale sistema di autenticazione sarà progettato per consentire agli utenti un controllo dettagliato sull'accesso alle informazioni attraverso l'uso di credenziali personalizzate, quali username e password. L'implementazione di regole di complessità per le password contribuirà a incrementare la resistenza contro eventuali tentativi di forza bruta. Tuttavia, questa strategia di sicurezza si estende oltre l'autenticazione tradizionale, contemplando l'integrazione di metodi multifattoriali, come, ad esempio, l'autenticazione tramite token, finalizzata a rafforzare la sicurezza complessiva.

L'autenticazione multifattoriale agirà come strato aggiuntivo di sicurezza, richiedendo più di un elemento di verifica prima di consentire l'accesso alle risorse critiche del sistema. Ad esempio, si potrebbe considerare l'invio di un codice di verifica al dispositivo mobile dell'utente insieme alle consuete credenziali di accesso. Questo approccio mira a ridurre il rischio di accessi non autorizzati, anche nel caso di compromissione delle credenziali

principali. L'integrazione di un sistema di autenticazione avanzato non solo accrescerà la sicurezza del sistema CRUD, ma dimostrerà anche l'impegno nell'offrire un ambiente digitale sicuro e affidabile.

Un'ulteriore possibilità di espansione di questo sistema riguarda l'implementazione di un *sistema di reportistica ed analisi di dati*; quest'ultimo, sarebbe mirato a fornire agli utenti un approccio più completo e illuminante alla gestione delle informazioni. Questo avanzamento si rivela cruciale nell'era digitale attuale, poiché l'accesso e l'interpretazione rapida dei dati sono diventati imperativi per le decisioni aziendali informate.



L'eventuale modulo di reportistica avrebbe come obiettivo quello di offrire agli utenti la possibilità di estrarre informazioni significative attraverso una varietà di metriche e indicatori chiave di performance. Questo strumento avanzato non solo semplifica l'analisi dei dati ma introduce anche la capacità di generare report personalizzati, adattati alle esigenze specifiche di ciascun utente o reparto. Tale flessibilità rappresenta un salto qualitativo nella capacità di comprendere i dati, trasformandoli in insight strategici.

Con la nuova piattaforma di analisi dei dati, gli utenti sarebbero in grado di esplorare le tendenze emergenti, individuare correlazioni nascoste e identificare pattern che potrebbero essere fondamentali per il successo dell'organizzazione. Il sistema potrebbe essere alimentato da algoritmi di Intelligenza Artificiale, e quindi, in grado di suggerire automaticamente analisi approfondite e fornire raccomandazioni basate sui dati. Ad esempio, analizzando i dati relativi alle operazioni solitamente associate ad ogni object type, e dato un oggetto, si potrebbe essere in grado di suggerire automaticamente le possibili operazioni di manutenzione da svolgere su di esso.

Questa evoluzione non solo migliora l'efficienza operativa, ma apre anche la strada a un livello superiore di adattabilità. Il sistema sarà in grado di anticipare le esigenze degli utenti, adattandosi dinamicamente alle richieste in continua evoluzione. L'aggiunta del sistema di reportistica ed analisi dati al sistema CRUD, per l'azienda, non rappresenta solo un passo avanti, ma una rivoluzione nell'approccio alla gestione delle informazioni.

Sempre nell'ambito degli sviluppi futuri, emerge la prospettiva di arricchire il sistema con l'implementazione di un sofisticato *sistema di logging* dedicato a monitorare e registrare ogni modifica effettuata ai dati presenti nel database. Questa innovazione rappresenterebbe un gran passo in avanti, con il quale si migliorerà di molto la tracciabilità delle operazioni; il sistema sarebbe in grado di fornire un resoconto dettagliato di ogni inserimento, aggiornamento o cancellazione di record. Esso permetterebbe agli utenti di accedere a un registro cronologico esauriente delle modifiche, identificando chi ha effettuato le operazioni, quando queste sono state eseguite e quali modifiche specifiche sono state apportate.

Tale approccio non solo incrementa la trasparenza operativa e la responsabilità nell'utilizzo del database, ma costituisce, anche, una preziosa risorsa per l'analisi di errori, la risoluzione di problematiche e la gestione delle versioni dei dati. Inoltre, un robusto sistema di logging contribuirà significativamente a rafforzare la sicurezza del sistema, rilevando tempestivamente attività sospette o non autorizzate, consolidando così l'integrità complessiva del sistema informativo.

CAPITOLO 6

Componente di Data Exchange

In questo capitolo della tesi, verranno illustrate nel dettaglio le fasi di progettazione e sviluppo che hanno portato alla realizzazione di un software di data exchange legato al database. Inizialmente, verranno fornite alcune delle ragioni che stanno alla base della realizzazione di questa componente, ponendo il focus sui vantaggi principali che possono essere ricondotti ad essa.

Successivamente, si sposterà il focus sul processo che ha portato alla definizione di un template univoco di caricamento, elencandone i principali pregi. Fatto ciò, sipasserà ad illustrare le fasi di progettazione e sviluppo del software, dedicando maggiore attenzione alla struttura portante, al funzionamento e agli strumenti utilizzati durante la programmazione.

Il capitolo si concluderà illustrando due esempi di data loading, prendendo in analisi il caricamento di due progetti caratterizzati da due standard rappresentativi differenti (standard ENI e standard JobPlan).

6.1 Introduzione al software di data exchange

Nella sezione precedente della tesi, si è parlato, in maniera estesa, di come l'interfaccia CRUD sviluppata rappresenti un punto di accesso ai dati presenti nel sistema; essa, infatti, permette di svolgere le operazioni basilari di gestione delle informazioni relative agli asset e ai piani di manutenzione.

In particolare, si tratta di operazioni "puntuali", ovvero eseguite in maniera mirata verso un numero ristretto di record. Quando si vuole agire in maniera massiva sulle informazioni, l'interfaccia CRUD sviluppata presenta delle forti limitazioni; queste ultime riguardano, soprattutto, il tempo necessario a svolgere gli inserimenti e le modifiche. È, infatti, facile capire che, per eseguire tali operazioni, bisogna agire record per record e, di conseguenza, quando si ha a che fare con moli di dati importanti, il tempo necessario per completare questi task cresce a dismisura.

Per queste ragioni, fin dalle prime fasi del progetto, è stato chiaro che si sarebbe dovuto sviluppare un software in grado di eseguire tutte quelle operazioni che sarebbero risultate troppo ingombranti se eseguite attraverso l'interfaccia CRUD. Questa componente di *data exchange* riuscirebbe a superare i limiti del CRUD riguardanti il caricamento massivo di dati nel sistema, tramite il raggiungimento di alcuni obiettivi prefissati.

In primis, lo sviluppo di un software di questo tipo sarebbe mirato ad ottenere un caricamento ed un'estrazione automatizzata di informazioni dal database. Ciò, vuol dire che, in fase di caricamento, il programma dovrebbe essere in grado di recuperare i dati da una

fonte e, accedendo alle varie tabelle, caricare tutti i record. Al contrario, in fase di estrazione, esso dovrebbe essere in grado di recuperare i record dalle strutture dati nel DB per poi mandare tutto ciò in output, usando un formato prestabilito.

Un secondo obiettivo chiave di questo software, riguarda le operazioni di validazione sui dati. Infatti, quando si parla di caricamento massivo di dati, una volta iniziata l'operazione di loading, si deve avere la certezza che non si incapperà in errori legati ai valori presenti nei record. Nel DB sono intrinsecamente presenti dei controlli di validazione, i quali sono rappresentati dai vincoli relazionali definiti in fase di sviluppo del sistema (ad esempio, vincoli di valore, vincoli di chiave primaria, vincoli di chiave esterna, etc.). Tuttavia, essendo questo software un componente esterno al database, andando a svolgere le operazioni di caricamento, qualora, per qualche ragione, si dovesse interrompere il flusso di informazioni (ad esempio, per la violazione di un vincolo nel DB) si rischierebbe di lasciare il sistema in uno stato inconsistente. Si immagini, ad esempio, di voler svolgere un caricamento di equipment; senza svolgere a monte una fase di validazione, si inizierebbe direttamente il caricamento. A questo punto nel DB ad ogni upload verrebbe incrementata la sequenza corrispondente agli ID degli equipment caricati. Se, nel mezzo dell'upload, dovesse essere violato qualche vincolo (ad esempio, se ci fosse un campo testuale più lungo del max lenght), l'operazione verrebbe interrotta, lasciando solo parte degli equipment nel sistema. Questo esempio fa capire il motivo dietro la necessità di includere una componente di validazioni in questo modulo di data exchange.

Oltre a tutto ciò, questo software, dovrebbe essere anche in grado di avvisare gli utenti dell'eventuale presenza di errori riscontrati durante le operazioni. Di conseguenza in questo programma verrà anche inclusa una componente di logging, in grado di comunicare in maniera alquanto più esaustiva possibile con gli utenti, segnalando tutte le problematiche necessarie di attenzione.

Per concludere questa sezione introduttiva, di seguito verranno riassunti alcuni degli obiettivi principali che si vogliono raggiungere con lo sviluppo di questa componente di data exchange:

- *Interoperabilità*: consentire la comunicazione efficace e lo scambio di dati tra sistemi diversi, che possono utilizzare formati di dati differenti o seguire standard diversi (ad esempio, standard ENI e standard job plan). L'obiettivo è garantire che i dati possano fluire senza intoppi.
- *Integrazione dei sistemi*: favorire l'integrazione senza problemi dei dati tra sistemi aziendali, eliminando silos di informazioni e consentendo un flusso continuo di dati tra applicazioni, database e processi aziendali.
- *Automazione dei processi di caricamento/estrazione*: semplificare e automatizzare i processi di scambio dei dati per ridurre la necessità di intervento manuale. Ciò può contribuire a migliorare l'efficienza operativa e a ridurre gli errori umani.
- *Consistenza e qualità dei dati:* garantire la coerenza, l'integrità e la qualità dei dati durante il processo di scambio. Ciò significa che i dati dovrebbero essere accurati, completi e conformi agli standard stabiliti.
- *Monitoraggio*: fornire uno strumento per il monitoraggio delle attività di scambio dei dati, la registrazione di eventi inaspettati e la generazione di report. Ciò aiuta a tracciare e comprendere il flusso dei dati e ad identificare eventuali problematiche o violazioni di vincoli.
- *Conformità*: garantire che il software di data exchange sia in conformità con gli standard di settore.

- *Velocità ed Efficienza*: ridurre i tempi di elaborazione dei dati e garantire che lo scambio di informazioni avvenga in modo rapido ed efficiente.

6.2 Sviluppo del template di caricamento

Come accennato nella sezione introduttiva, uno degli scopi principali legati allo sviluppo del modulo di data exchange, è quello di fornire uno strumento che consenta di interagire con il DB in maniera massiva ed automatizzata; come già detto, tutto ciò porterebbe ad enormi vantaggi in termini di tempo risparmiato e robustezza delle operazioni.

Nei capitoli precedenti di questa tesi è già stato spiegato come, durante la definizione dei piani di manutenzione, in base al cliente per il quale l'azienda sta lavorando, vengono usati dei formati di rappresentazione dei dati differenti. Un primo approccio allo sviluppo di questa componente di data exchange è stato quello di sviluppare il codice sulla base dello standard rappresentativo più vicino a SAP, ovvero lo *Standard ENI*. Quest'ultimo si presenta come un insieme di template chiamati Annex; questi sono:

- *Annex 01 - Functional Locations* (Figura 6.1);
- *Annex 02 - Equipments* (Figura 6.2);
- *Annex 02c - Classes and Characteristics* (Figura 6.3);
- *Annex 03 - Materials*;
- *Annex 04 - Bill of Materials*;
- *Annex 05 - Measuring Points*;
- *Annex 06 - Work Centres*;
- *Annex 07 - Standard Texts*;
- *Annex 08 - Task list* (Figura 6.4);
- *Annex 09 - Maintenance Items* (Figura 6.5);
- *Annex 10 - Maintenance Plans* (Figura 6.6).

Inizialmente, quindi, lo sviluppo della componente di scambio dati era ritagliata su questo standard. Di conseguenza, tutti i controlli e la validazione venivano effettuati tenendo come punto fermo i campi presenti nei file Annex. Tuttavia, nonostante la grande somiglianza fra lo standard ENI e il modo in cui i dati sono rappresentati nel DB, questo approccio si è rivelato inapplicabile fin da subito. Questo perchè, nella realtà operativa aziendale, anche fra progetti che condividono lo stesso standard alla base (in questo caso, lo standard ENI), esistono delle non conformità. Per di più, per quanto lo standard ENI sia vicino al modo di rappresentare i dati nel DB, esistono delle differenze; ad esempio, una delle più ingombranti è rappresentata dal fatto che nello standard ENI si fa riferimento alla posizione di manutenzione a partire dal piano; questo rappresenta un collegamento di tipo completamente diverso rispetto a quello previsto dal database (dove si fa riferimento al piano di appartenenza dal maintenance item).

Per tutte queste ragioni, è stato deciso di basare lo sviluppo del modulo di data exchange su un *template di caricamento*. Quest'ultimo rappresenta un vero e proprio standard di rappresentazione che i dati devono rispettare al fine di essere caricati nel DB. L'introduzione di questa costante nello sviluppo, ha permesso di poter fare affidamento su un formato standard, sul quale basare tutta la progettazione e la successiva codifica.

	A	B	C	D	E	F
1	**Field Label**	**Description**	**Data type**	**Length**	**Status**	**Constraints**
2	Structure Indicator	The structure indicator of a functional location determines the structure of the functional location label	Alphanumeric	5	Mandatory	ZFORM if Functional Location code is up to 30 characters long, ZFORM2 if Functional Location code has a length between 31 and 40 chars.
3	Functional Location	Alphanumeric code identifying the technical object site at a Company level. It is assigned manually.Coding is guided by structure Indicator attribute. The pattern is (XXXX-SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS) where: "X" both numbers and letters can be entered "S" numbers, letters, and special characters (only "-") can be entered	Alphanumeric	40	Mandatory	'Item numbering' (ref. document [1])
4	Functional Location Category	Key distinguishing the most suitable maintenance discipline related to the Functional Location according to its use.	Alphanumeric	1	Mandatory	Validation Table 44
5	Description	Description of Functional Location managed through a Short text	Alphanumeric	40	Mandatory	
6	Class	Name used to uniquely identify a class within a class type. More than one class can be linked to every Functional Location. (PER ORA UNA)	Alphanumeric	18	Mandatory	Validation Table 1
7	User status	Individual status of an object (short form).	Alphanumeric	4	Mandatory	Validation Table 33
8	Object type	Code identifying the equipment type code	Alphanumeric	10	Mandatory	Validation Tables 2-32
9	Authorization Group	Authorization Group identifies the users allowed to use	Alphanumeric	4	Optional	
10	Installation allowed	It indicates whether the installation of an Equipment is allowed or not.The choice is highlighted through a flag. The installation of Equipments will be allowed under the Functional Unit Level	Alphanumeric	1	Mandatory	Flag=X
11	Single Installation	It indicates if the installation of single Equipment is allowed.The choice is highlighted through a flag	Alphanumeric	1	Mandatory	Flag=X
12	Superior Functional Location	Functional location which is hierarchically directly above the functional location that you are creating, changing or displaying. The field is not mandatory only for first level functional location	Alphanumeric	30	Mandatory	
13	Maintenance Plant	It identifies Maintenance Plant Code.	Alphanumeric	4	Mandatory	Validation Table 34
14	Location	It indicates the "Area Code".	Alphanumeric	10	Mandatory	Validation Table 35
15	Criticality	It is used to highlight the criticality of Functional Locations	Alphanumeric	1	Mandatory	Validation Table 36
16	Effect	It is used to define consequences that a breackdown or a drowback could generate (effect on Production - Environment - Safety).	Alphanumeric	3	Mandatory	Validation Table 37
17	SCE	Safety Criticality Element	Alphanumeric	1	Mandatory	Validation Table 45
18	Company Code	It is the organizational unit of accounting	Alphanumeric	4	Mandatory	Validation Table 38
19	Cost Center	It identifies the Functional Location Cost Center.	Alphanumeric	10	Optional	Validation table 39
20	WBS Element	The WBS elements are SAP PS entities which represent Budget elements.	Alphanumeric	24	Mandatory	Validation table 40
21	Planning Plant	It identifies the Maintenance Planning Plant.	Alphanumeric	4	Mandatory	Validation Table 41
22	Maintenance Planner Group	It uniquely identifies the group of people responsible for Maintenance planning of an object inside a Maintenance Planning Plant.	Alphanumeric	3	Optional	Validation Table 42
23	Main Work Center	It uniquely indicates the work Center responsible of performing maintenance on a certain object. It is recorded in Notifications and Work Order.	Alphanumeric	8	Optional	Validation Table 43 (examples used in NAOC)

Figura 6.1: Annex 01 - Functional Locations

Uno dei principali vantaggi introdotti dall'utilizzo di un template per il caricamento massimo delle informazioni, è che esso permette di recuperare tutti quei vantaggi associati all'utilizzo di software di data entry come Excel. Infatti, operando sul template sarà possibile operare su un numero molto alto di record; una volta eseguite le modiche sul template, esso verrà dato in input al software di caricamento che eseguirà l'upload sul DB. Il template definito si basa su nove contenitori, ognuno dei quali fa riferimento ad uno o più elementi chiave. I file sono i seguenti:

- *Validation*: si tratta di un template per il caricamento delle informazioni relative alle tabelle di validazione; all'interno di questo file sono presenti tutti quei dati di tipo key-value che sono stati usati per rappresentare i tagli e le semplificazioni fatte sulla struttura SAP di partenza. Avere tutta la parte di validazione in un unico file (a differenza dello standard ENI dove ogni Annex aveva la sua validazione) ha permesso di svolgere tutti i controlli sui valori dei dati in maniera centralizzata e monolitica.

	A	B	C	D	E	F
1	**Field Name**	**Description**	**Data type**	**Length**	**Status**	**Constraints**
2	Equipment	Number identifying the Equipment. This will be assigned externally for conversion purpose.	Numeric	10	Mandatory	
3	Category	Key distinguishing individual technical objects according to their use	Alphanumeric	1	Mandatory	Validation Table 32
4	Description	Description of Equipment managed through a Short text	Alphanumeric	40	Mandatory	
5	User status	Individual status of an object (short form):	Alphanumeric	4	Mandatory	Validation Table 33
6	Object Type	Code identifying the "equipment item code".	Alphanumeric	10	Mandatory	Validation Tables 2-31
7	Authorization Group	Authorization Group identifies the users allowed to use Equipments	Alphanumeric	4	Optional	Validation Table 34
8	Start-up date	Time at which the technical object came into operation	Date	YYYYMMD	Optional	
9	Manufacturer	Code identifying the manufacturer	Alphanumeric	30	Mandatory	
10	Manufacturer Country	Production country ID	Alphabetic	2	Optional	Validation Table 35
11	Construction Year	Construction Year	Numeric	4	Optional	
12	Construction Month	Construction Month	Numeric	2	Optional	
13	Model Number	Code by which constructor identifies material model used to group technical object of the same type.	Alphanumeric	20	Mandatory	
14	Manufacturer Serial number	The serial number is the number that is assigned by the manufacturer for an individual technical object	Alphanumeric	30	Mandatory	
15	Vendor	Code identifying the vendor of the equipment.	Alphanumeric	10	Optional	Validation Table 36
16	Location	It indicates the "Area Code". For now this value is to be define.	Alphanumeric	10	Optional	Validation Table 37
17	Criticality	It is used to highlight the criticality of Equipment. Use "X".	Alphanumeric	1	Mandatory	Validation Table 38
18	Effect	It is used to define consequences that a breackdown or a drowback could generate (i.e: Production - Environment - Safety).	Alphanumeric	3	Mandatory	Validation table 39
19	Asset	It uniquely identifies the asset code. Assets are managed in FI module only (Finance & Accounting)	Alphanumeric	12	Optional	Validation Table 40
20	Cost Center	It identifies the Functional Location Cost Center	Alphanumeric	10	Optional	Validation Table 41
21	WBS element	The WBS elements are SAP CO-PS (Cost Controlling-Project System) entities which represent Budget elements. If it is filled, it is automatically proposed during work order management	Alphanumeric	24	Mandatory	Validation Table 42
22	Planning Plant	It identifies Maintenance Planning Plant.	Alphanumeric	4	Mandatory	Validation Table 43
23	Maintenance Planner Group	It identifies the group of people responsible for maintenance Planning inside Maintenance Planning Plant (Departments)	Alphanumeric	3	Optional	Validation Table 44
24	Main Work Center	It identifies the unique Work Center responsible of performing maintenance for a certain object.It is copied in Notifications and Work Orders	Alphanumeric	8	Optional	Validation Table 45
25	Catalog Profile	It identifies a group of code groups for certain catalogs, referred to specific equipments	Alphanumeric	9	Optional	Validation Table 46
26	Functional Location	Label that uniquely identifies the functional location where the Equipment is installed. Select values from template 5	Alphanumeric	30	Mandatory	
27	Superior Equipment	Number that identifies the superior equipment associated with this piece of equipment.	Alphanumeric	10	Optional	
28	Construction Type	Material number (Number that uniquely identifies a Material in the system) under which pieces of Equipment of the same construction are grouped.It addresses Material BOMs. It should be an IBAU material type. A unique assignment is allowed for every Equipment. Select value from template 3.	Alphanumeric	18	Optional	
29	Equipment Tag Number	It is the number of the equipment as referenced on company technical documents (TAG n.).	Alphanumeric	30	Mandatory	
30	Manufacturer Part number	Number which is given to the technical object by the manufacturer and which allows him to manage objects of the same category	Alphanumeric	30	Optional	

Figura 6.2: Annex 02 - Equipments

- *Equipments*: questo file conterrà tutte le informazioni relative alle attrezzature; di fatto, si tratta di una copia dei campi relativi agli equipment presenti nel database.
- *Functional locations*: questo file conterrà tutte le informazioni relative alle sedi tecniche; come nel caso precedente, si tratta di una copia dei campi relativi alle functional location nel DB.
- *Assemblies*: contiene tutte le informazioni relative agli oggetti che fanno parte di un assembly.

	A	B	C	D	E	F
1	**Field Name**	**Description**	**Data type**	**Length**	**Status**	**Constraints**
2	Equipment	Number identifying the Equipment. This will be assigned externally for conversion purpose. A number range for equipment master data will be defined for each country. Select data from template 6	Numeric	10	Mandatory	
3	Class	Name used to uniquely identify a class within a class type. More than one class can be linked to every equipment [this information is taken from 01 Classes and Characteristics values]	Alphanumeric	18	Mandatory	Validation Table 1
4	Characteristic	They are in relation with the Class, see the combination between Class & Characteristics.	Alphanumeric	30	Mandatory	Validation Tables 5-31. They are in relation with the Class, see the combination between Class & Characteristics .
5	Value	They are defined for some characteristics (see validation table). If validation table does not exist, it is possible to assign values. (possible dimensions are displayed in page "Characteristics Dimension")	Alphanumeric	30	Mandatory if value exists for a given characteristic	Validation Tables 32-74

Figura 6.3: Annex 02c - Classes and Characteristics

- *Classes and Characteristics*: contiene tutte le informazioni relative ai valori delle caratteristiche che descrivono ogni oggetto.
- *Task Lists*: questo template è suddiviso in tre sezioni. La prima è relativa alla definizione dei cicli di manutenzione, mentre la seconda elenca le operazioni associate ad ogni ciclo. All'interno della terza sezione sono presenti le informazioni relative ai pacchetti di frequenza associabili ad ogni operazione.
- *Maintenance Plans*: in questo file sono presenti le colonne che descrivono i campi relativi ai piani di manutenzione.
- *Maintenance items*: in questo file sono presenti le colonne che descrivono i campi relativi alle posizioni di manutenzione (per le quali viene definito anche il piano di appartenenza).
- *Object List*: in questo ultimo file viene rappresentata l'associazione fra gli oggetti definiti nell'ambito del progetto e le relative posizioni di manutenzione.

6.3 Progettazione componente data exchange

Si passerà, ora, a fornire un approfondimento del processo di progettazione della componente di scambio dati. Gli obiettivi principali da raggiungere con la progettazione di una componente software di questo tipo sono i seguenti:

- *Controllo del formato*: controllare il formato dei dati prima di immeterli in un database è fondamentale per diversi motivi. Innanzitutto, questo processo garantisce che i dati immessi rispettino la struttura e il formato previsti dallo schema del database. Ciò contribuisce a mantenere l'integrità dei dati, evitando errori e garantendo che i dati siano coerenti con le regole definite. Si hanno, anche, dei benefici per quanto riguarda la consistenza dei dati; è necessario assicurarsi che i dati siano uniformi e coesi all'interno del database. Se i dati hanno formati diversi o seguono regole diverse (standard differenti), potrebbe essere difficile eseguire query o analisi coerenti e accurate.

  In secondo luogo, ci sono anche delle ragioni legate al risparmio di spazio e alla prevenzione di errori. Ovvero, eseguendo dei controlli automatizzati sul formato dei dati in ingresso, si limita l'uso inefficiente dello spazio di archiviazione. Se i dati

Field (SAP name)	Description	Data type	Length	Status	Constraints
Task List type	Key which classifies task lists according to their functionality	Alphabetic	1	Mandatory	Validation Table 1
Equipment	Number identifying the Equipment. Use equipment defined in Annex 02.	Numeric	10	Mandatory only for Task List Type E	
Functional Location	Alphanumeric code identifying the technical object site at a Company level. Use Functional Location defined in Annex 01.	Alphanumeric	30	Mandatory for Task List Type T	
Group	Code to be considered as a folder in which several Task Lists regarding the same criteria can be grouped. Number is automatically assigned for EQ and FL task lists. The group is user defined for General Task Lists. Raccomandations: use a talking code for task list group in order to easily manage them	Alphanumeric	8	Mandatory	validation table 15
Group counter	Code that uniquely identifies a Task List within a Group	Alphanumeric	2	Mandatory for Task List Type A	
Short Description	Short text description of the task list (used for the name of the operations)	Alphanumeric	40	Optional	
Main Work Center	It is a reference Work Center for the predominant activity of the task list and is recorded as default work center in task list operations. Use data from template 1.	Alphanumeric	8	Mandatory	
Planning Plant	It identifies the Maintenance Planning Plant.	Alphanumeric	4	Mandatory	Validation Table 2
Usage	It defines the area in which task lists are used. Choose: 4 - Plant Maintenance	Alphanumeric	3	Mandatory	Validation Table 3
Responsible Planner group	Group responsible for Planning. Validation list is different from those one used into Work Orders and Preventive Maintenance Items	Alphanumeric	3	Mandatory	Validation Table 4
Status	It defines the status of the Task List. Default: 4 - Released (General)	Alphanumeric	3	Mandatory	Validation Table 5
Maintenance Strategy	It defines maintenance strategy.	Alphanumeric	6	Mandatory	Validation Table 6
System Condition	It defines the overall condition of the technical system	Numeric	1	Optional	Validation Table 7
Assembly	In task list general, it defines the BOM material necessary to select spare parts needed for the operations. Use data defined in Annex 03.	Alphanumeric	18	Optional	
Long Description	Long text description of the task list (used for the activities)	Alphanumeric	72	Optional	
Operation	Code identifying the sequence of operations. Each operation is linked to a discipline/package involved. Numbering is 10, 20, 30...	Numeric	4	Mandatory	
Sub-operation	Determines in which order the sub-operations of an operation are listed	Numeric	D	Optional	
Work Center	Code identifying the Work Center in charge of the operation. Use values from Annex 06.	Alphanumeric	8	Mandatory	
Control Key	Determines whether the operation is performed by an internal or external work center.	Alphanumeric	4	Mandatory	Validation Table 8
Plant	It defines the Plant for Work center	Alphanumeric	4	Mandatory	Validation Table 9
Standard text key	Code that uniquely identifies the key for standard text. Use data from Annex 07.	Alphanumeric	7	Optional	
Description	Description of the Maintenance Operation to be carried out	Alphanumeric	40	Mandatory	
Duration	The unitary time of operation for manpower.	Numeric	7	Mandatory	
Work (Man Hours)	Overall time necessary for the execution of a single operation. It will be calculated by the system if calculation criteria is set to '2'.	Numeric	9	Optional	
UM	Unit of measure of work (days, hours, minutes)	Alphanumeric	3	Mandatory	Validation Table 10
No	Number of Human Resources necessary to carry out the activity.	Numeric	3	Optional	
Calculation Criteria	Choose: "2 - calculate work" for automatic calculation	Numeric	1	Optional	Validation Table 11
Execution Factor	It indicates how many times the activity is to be performed	Alphanumeric	3	Optional	
Long Text Operation	Long text for operation description	Alphanumeric	72	Optional	
Packages (1st, 2nd, 3th...)	This field identifies the Maintenance packages available for each strategy.	Alphanumeric	2	Optional	Validation Table 12-14
Key date	Date on which the task list is edited or displayed.	Char	8	Mandatory	
Indicator for reports	User-defined field: Indicator for reports	Char	1	Optional	
SCMOp	is the indicator for *Safety Critical Maintenance for Operations within Work Orders*	flag	1	Optional	

Figura 6.4: Annex 08 - Task list

vengono memorizzati in formati non ottimali o con una struttura errata, potrebbe essere sprecato spazio prezioso nel database.

Come accennato in precedenza, si riduce anche il rischio di errori durante l'inserimento dei dati. Se essi non rispettano il formato richiesto, ciò potrebbe portare a errori durante

	A	B	C	D	E	F
1	**Field (SAP name)**	**Description**	**Data type**	**Length**	**Status**	**Constraints**
2	Maintenance Item	External number range assigned: from 200000000000 to 299999999999	Numeric	12	Mandatory	
3	Maintenance Plan category	The value will always be Preventive Maintenance Plan (PM)	Alphanumeric	2	Mandatory	Default: **PM**
4	Maintenance Strategy	This attribute is used when a Maintenance Plan is to be linked to a specific Strategy. The task list associated to Maintenance Items (and therefore to Maintenance Plan) must have the same Maintenance Plan strategy.If Maintenance Item (and therefore Maintenance Plan) is not Strategy based, the field "strategy" must be empty for the Task List also	Alphanumeric	6	Mandatory	Validation Table 1
5	Item Short Text	Maintenance Item description (short text)	Alphanumeric	40	Mandatory	
6	Planning Plant	Planning Plant in which Maintenance tasks are planned	Alphanumeric	4	Mandatory	Validation Table 2
7	Maintenance Planner Group	Group or person responsible for Planning Work Order.	Alphanumeric	3	Mandatory	Validation Table 3
8	Main Work Center	Work Center in charge of Maintenance tasks. Use data from template 1	Alphanumeric	8	Mandatory	
9	Order Type	It identifies the work order type. For Preventive Maintenance work order the value should be set to "PM02".	Alphanumeric	4	Mandatory	**PM02**
10	Maintenance activity type	Define the type of maintenance work	Alphanumeric	3	Mandatory	Validation Table 4
11	Priority	Defines the priority of Work Order that will be issued	Alphanumeric	1	Mandatory	Validation Table 5
12	Task List type	Key which classifies task lists according to their functionality.	Alphabetic	1	Mandatory	Validation Table 6
13	Group	Code to be considered as a folder in which several Task Lists regarding the same criteria can be grouped. Number is automatically assigned for EQ and FL task lists, and can be user defined for General Task Lists	Alphanumeric	8	Mandatory	
14	Functional Location	the functional location where the technical object is installed. It can be the object of preventive maintenance activities itself.	Alphanumeric	30	Mandatory	
15	Equipment	Technical Object on which maintenance activities should be performed. It must be installed on the Functional Location selected, in order to ensure consistency of data.	Alphanumeric	10	Optional	
16	Group counter	This key and the task list group uniquely identify a task list	Numeric	2	Mandatory	
17	**Task List Factor**	It indicates how many times the activities of the task list have to be performed. If the maintenance item has an object list, the task list factor will be equal to the number of object of the list.	Numeric	3	Optional	
18	**SCM (Safety Critical Maintenance)**	It is the indicator for Safety Critical Maintenance for Work Orders.	Alphanumeric	1	Optional	

Figura 6.5: Annex 09 - Maintenance Items

	A	B	C	D	E	F
1	**Field (SAP name)**	**Description**	**Data type**	**Length**	**Status**	**Constraints**
2	Maintenance Plan	Code internally assigned. Since Maintenance Plan name will be use to identify plans for scheduling, a proper talking code it's compulsory. Please choose a name that can identify key objects for plan such as locations, planner group or planning plant (so locations, responsable department for planning....)	Alphanumeric	12	Mandatory	validation table 3
3	Plan Description	Maintenance Plan description (short text)	Alphanumeric	40	Mandatory	
4	Long text	Maintenance Plan description (long text)	Alphanumeric	72	Default: blank	
5	Measuring Point-Equipment	It Is the number of the Equipment where the Measuring Point is installed. Use data from Annex 02	Numeric	10	Mandatory if the plan is counter-	
6	Measuring Point	Measuring point linked to the plan. In case of counter based plans, a Counter must be indicated. This information will be retrieved in SAP after conversion through association of two fields: "equipment" and "measuring position". Use data from Annex 05.	Numeric	12	Mandatory if the plan is counter-based	
7	Start Counter Reading	It is the starting number of the counter reading	Numeric	22	Mandatory if the plan is counter-	
8	Start of cycle	Date from which the system starts scheduling work orders	Date	YYYYMMGG	Optional	
9	Scheduling Period	It determines the length of time for which the system creates maintenance calls during maintenance plan scheduling	Numeric	3	Mandatory	
10	Unit Of Measure	Unit of measure of work (days, hours, minutes, etc.)	Alphanumeric	3	Mandatory	Validation Table 1
11	Shift factor for late completion	The shift factor in the event of late completion of a maintenance task defines the percentage of the shift to be applied to the calculation of the next date. It is only applied when the variance between target and actual date is outside the tolerance range.	Numeric	3	Optional	from 1 to 100
12	Tolerance for late completion	The tolerance in the case of late completion determines the time span in which positive variances between actual and planned date do not influence subsequent scheduling. The tolerance is defined as a percentage rate of the smallest interval between the maintenance cycles of the maintenance strategy. The smallest interval is used so that a date shift does not automatically lead to a planned date	Numeric	3	Optional	from 1 to 100
13	Shift factor for early completion	The shift factor in the event of early completion of a maintenance task defines the percentage of the shift to be applied to the calculation of the next date. It is only applied when the variance between target and actual date is outside the tolerance range.	Numeric	3	Optional	
14	Tolerance for early completion	The tolerance in the case of late completion determines the time span in which positive variances between actual and planned date do not influence subsequent scheduling. The tolerance is defined as a percentage rate of the smallest interval between the maintenance cycles of the maintenance strategy. The smallest interval is used so that a date shift does not automatically lead to a planned date	Numeric	3	Optional	from 0 to 100
15	Cycle Modification Factor	It allows to change the cycle times of a maintenance strategy individually for each maintenance	Alphanumeric	4	Optional	
16	Call horizon	The call horizon determines when a maintenance call object, for example a maintenance order should be generated for a maintenance call. It's a percentage of the time between two calls. The time length depends on the package that have been assigned to the plan.	Numeric	3	Mandatory	From 0 to 100
17	Completion Requirement	If you set this indicator, the system only generates the next call object once the preceding call object has been confirmed.	Alphanumeric	1	Optional.	Flag X
18	Maintenance item	Code identifying the Maintenance Item (or items) to be assigned. Must be the same declared in	Alphanumeric	12	Mandatory	
19	Strategy	This attribute is used when a Maintenance Plan is to be linked to a specific Strategy.	Alphanumeric	6	Optional. Mandatory in	Validation Table 2
20	Measuring Point	Measuring point or counter code number. It is the code number generated by the system	Numeric	12	Mandatory if the plan is counter-	

Figura 6.6: Annex 10 - Maintenance Plans

le operazioni di inserimento o durante l'esecuzione di query.

- *Controllo di errori di riferimento*: l'importanza dietro questo obiettivo riguarda principalmente l'integrità dei dati. Se i riferimenti sono errati, potrebbero verificarsi problemi nel reperimento delle informazioni, portando a conclusioni errate o a decisioni basate su dati inaccurati. Un'altra ragione si trova nell'affidabilità delle query. Il database in questione viene utilizzato per l'esecuzione di query finalizzate ad ottenere informazioni specifiche. Se ci sono errori di riferimento nei dati, le query potrebbero restituire risultati imprecisi o non riuscire a restituire i dati desiderati, influenzando la qualità delle informazioni ottenute. Ogni database dovrebbe sempre mantenere la consistenza dei dati in tutte le sue tabelle e relazioni, in quanto eventuali errori di riferimento potrebbero portare a situazioni di incoerenza tra i dati. Altre motivazioni dietro questo obiettivo sono legate alla necessità di evitare duplicazioni e ridurre il rischio di errori umani durante le correzioni. Gli errori di riferimento possono contribuire alla presenza di dati duplicati o orfani all'interno del database. Ciò può rendere difficile la gestione e la manutenzione del database nel tempo. Per di più, controllare i riferimenti prima di immettere i dati nel database aiuta a prevenire errori umani, garantendo che le relazioni e le chiavi esterne siano corrette fin dall'inizio. Ciò riduce la probabilità di dover affrontare correzioni complesse in futuro (durante le quali potrebbero essere introdotte ulteriori imprecisioni).

- *Correzione ed adattamento dei dati in ingresso*: idealmente, il software di data exchange dovrebbe anche essere in grado di correggere autonomamente parte di questi errori; ciò vuol dire che si dovrebbe riuscire, dove possibile, a svolgere delle operazioni di adattamento e correzione atte ad eliminare eventuali imprecisioni nella forma e nei valori dei dati. Un esempio potrebbe essere quello di troncare le stringhe che risultano troppo lunghe per il campo al quale sono destinate; se, da una parte, questa operazione andrebbe ad inserire dei dati parziali nel DB, dall'altra permetterebbe di non bloccare tutto il caricamento per dei campi testuali troppo lunghi.

- *Gestione automatizzata dell'output*: un'altra sotto-componente di questo software dovrebbe essere dedicata all'output dei dati relativi ad un progetto su un template prestabilito. Questa operazione sarebbe molto utile per diverse ragioni:

  - Creare file di output in formati specifici (come CSV, Excel, JSON, XML) per l'analisi dei dati, la condivisione con altri sistemi o la generazione di report. Ciò consente una maggiore flessibilità nell'utilizzo dei dati al di fuori del database.
  - Creare copie di backup dei dati presenti nel database per scopi di sicurezza. In caso di perdita di dati o guasti del sistema, il software può essere utilizzato per ripristinare i dati nel loro stato originale.
  - Trasferire i dati da un sistema di database a un altro, ad esempio durante l'aggiornamento del database o la sostituzione di un sistema con un altro. Un software di output dei dati può aiutare a garantire una migrazione senza problemi e a preservare l'integrità dei dati durante il processo.
  - Un possibile sviluppo futuro, da associare a questa sottocomponente, che sarebbe di enorme aiuto all'interno dell'azienda, sarebbe quello relativo alla capacità di eseguire l'output dei dati di processo direttamente nei template utilizzati per eseguire il caricamento dei piani di manutenzione e degli asset in SAP. Ciò rappresenterebbe un passaggio diretto fra le informazioni rappresentate con lo standard interno aziendale ed il formato dei dati che costituiscono l'output per il cliente finale (SAP PM).

- *Logging*: per concludere, come accennato in precedenza, si vuole che il programma abbia la capacità di segnalare errori e warning in un linguaggio facilmente interpretabile anche dai non addetti.

  Il software di logging deve essere in grado di identificare e registrare ogni errore che si verifica nel sistema. Ciò include errori di sintassi SQL, violazioni di vincoli di integrità, errori di connessione al database e altri problemi operativi. Per di più, questa componente deve facilitare la diagnosi rapida dei problemi. Fornendo dettagli specifici sugli errori, come messaggi di errore, timestamp e dettagli delle query coinvolte, il personale di supporto può individuare e risolvere i problemi in modo più efficiente.

Fornita questa prima panoramica sugli obiettivi principali che si vorrebbero raggiungere con questo software, si passerà, ora, ad illustrare la struttura sulla quale si baserà il funzionamento di tutto il software. In questa sezione ci si concentrerà sulla parte di data entry, quindi solo sul caricamento automatizzato dei dati (tralasciando, per ora, la parte legata all'output).

Durante la fase di progettazione di questa sotto-componente è stato deciso di seguire un approccio modulare. Ciò sta a significare che l'intero software sarà suddiviso in una serie di moduli, ognuno dei quali sarà relegato ad uno scopo preciso. Si passerà, ora, a fornire una rapida descrizione di queste componenti, le quali, nel complesso, svolgeranno le operazioni di controllo e caricamento.

- *FormatErrorChecking*: questa componente conterrà tutti i metodi finalizzati al controllo dei dati presenti nel template di caricamento. Essa svolgerà controlli sulla loro forma e sui valori assunti; essa eseguirà, anche, tutte le operazioni di validazione riguardanti i riferimenti fra record. Si tratta di un modulo pensato per agire su ogni "gruppo" di dati presenti nel sistema (eseguirà controlli sulle tabelle di validazione, sulle tabelle relative all'asset register e sulle tabelle del maintenance register).

- *ValidationDataLoading*: questo modulo presenterà al suo interno tutti gli script di caricamento riguardanti le tabelle di validazione.

- *AssetDataLoading*: questa componente avrà al suo interno tutti i metodi e gli oggetti necessari a svolgere il caricamento delle tabelle relative a dati di asset. Ovviamente, il codice presente in questa componente darà per scontato che i dati siano privi di errori.

- *MaintenanceDataLoading*: questa quarta sotto-componente verrà sviluppata con l'obiettivo di caricare tutti i dati relativi ai piani di manutenzione. Come per i due moduli precedenti, anche qui si darà per scontato che nei dati non saranno presenti error; di conseguenza, non verranno effettuati ulteriori controlli sui loro valori.

L'idea di base, quindi, è quella di avere un software completamente modulare in cui ogni componente mira ad un obiettivo ben preciso. Un'altra caratteristica fondamentale alla base della progettazione riguarda la sequenzialità nell'esecuzione del codice relativo alle diverse componenti. È, infatti, facile capire come non è possibile avviare la fase di caricamento se prima non è stato svolto l'Error Checking da parte del primo modulo.

I vantaggi che si ottengono impostando un software in questo modo sono innumerevoli; di seguito ne verranno elencati alcuni, ritenuti i più interessanti:

- *Flessibilità*: un software modulare consente di aggiungere, rimuovere o sostituire moduli in modo relativamente semplice. Ciò rende il sistema flessibile e adattabile a cambiamenti nei requisiti, facilitando l'evoluzione del software nel tempo.

- *Riutilizzo del codice*: i moduli possono essere progettati in modo indipendente e riutilizzati in diverse parti del sistema. Ciò riduce il tempo e lo sforzo necessari per lo sviluppo

di nuove funzionalità, in quanto è possibile utilizzare i moduli esistenti senza dover riscrivere tutto il codice da zero.

- *Manutenibilità semplificata*: la suddivisione del software in moduli facilita la manutenzione in quanto è possibile concentrarsi su singoli moduli senza dover considerare l'intero sistema. Ciò semplifica la risoluzione di bug, l'implementazione di miglioramenti e l'applicazione di correzioni senza influire su altre parti del software.
- *Sviluppo parallelo*: in futuro, se dovesse essere necessario, i moduli potrebbero essere sviluppati in modo indipendente e parallelamente da team diversi. Ciò accelera il processo di sviluppo, consentendo una maggiore distribuzione del lavoro e una riduzione dei tempi complessivi.
- *Facilità di testing*: si tratta, forse, del vantaggio più incisivo; moduli più piccoli e indipendenti sono più facili da testare rispetto a un sistema monolitico. Ciò semplifica l'individuazione e la correzione di errori, migliorando la qualità complessiva del software.
- *Scalabilità*: la modularità facilita l'aggiunta di nuove funzionalità o l'espansione del sistema senza dover riscrivere l'intero codice. Questo rende il software più scalabile, in grado di crescere e adattarsi alle esigenze che cambiano nel tempo.
- *Riduzione dei rischi*: nel caso di problemi o cambiamenti nei requisiti, la sostituzione di un singolo modulo è meno rischiosa e più gestibile rispetto alla modifica di un sistema monolitico. Ciò può ridurre i rischi associati agli aggiornamenti o alle modifiche del software.

Si passera, ora, a definire più nello specifico le relazioni che intercorrono fra i vari moduli del programma; per far ciò verrà rappresentato il flusso di chiamate a metodi, portato avanti dalle componenti durante l'operazione di caricamento dei dati relativi ad un progetto. Al fine di fornire una rappresentazione quanto più chiara possibile, si farà affidamento su di un diagramma di sequenza (Figura 6.7).

Come è possibile osservare in Figura 6.7, l'operazione di caricamento è portata avanti tramite uno scambio di chiamate che ha come punto fermo centrale il modulo di controllo degli errori. Infatti, come accennato in precedenza, non potendo procedere alla fase di loading senza aver prima scongiurato la presenza di errori, ogni modulo di caricamento, prima di partire con la propria routine, esegue una chiamata verso il modulo FormatErrorChecking; in questo modo, ogni fase di caricamento è preceduta da una fase di validazione, finalizzata a far terminare lo step che succederà a valle senza intoppi.

Ognuno dei metodi presenti nelle componenti, nel caso in cui dovessero verificarsi errori, dovrà essere in grado di segnalare le problematiche riscontrate eseguendo l'output in un file testuale chiamato *ErrorLog.txt*. Il linguaggio usato in questi avvisi dovrà essere facilemente interpretabile, anche dai non addetti al settore; in questo modo, chiunque si troverà a svolgere il caricamento, nel caso di errori, sarà in grado di capire con rapidità cosa modificare al fine di poter procedere con l'operazione.

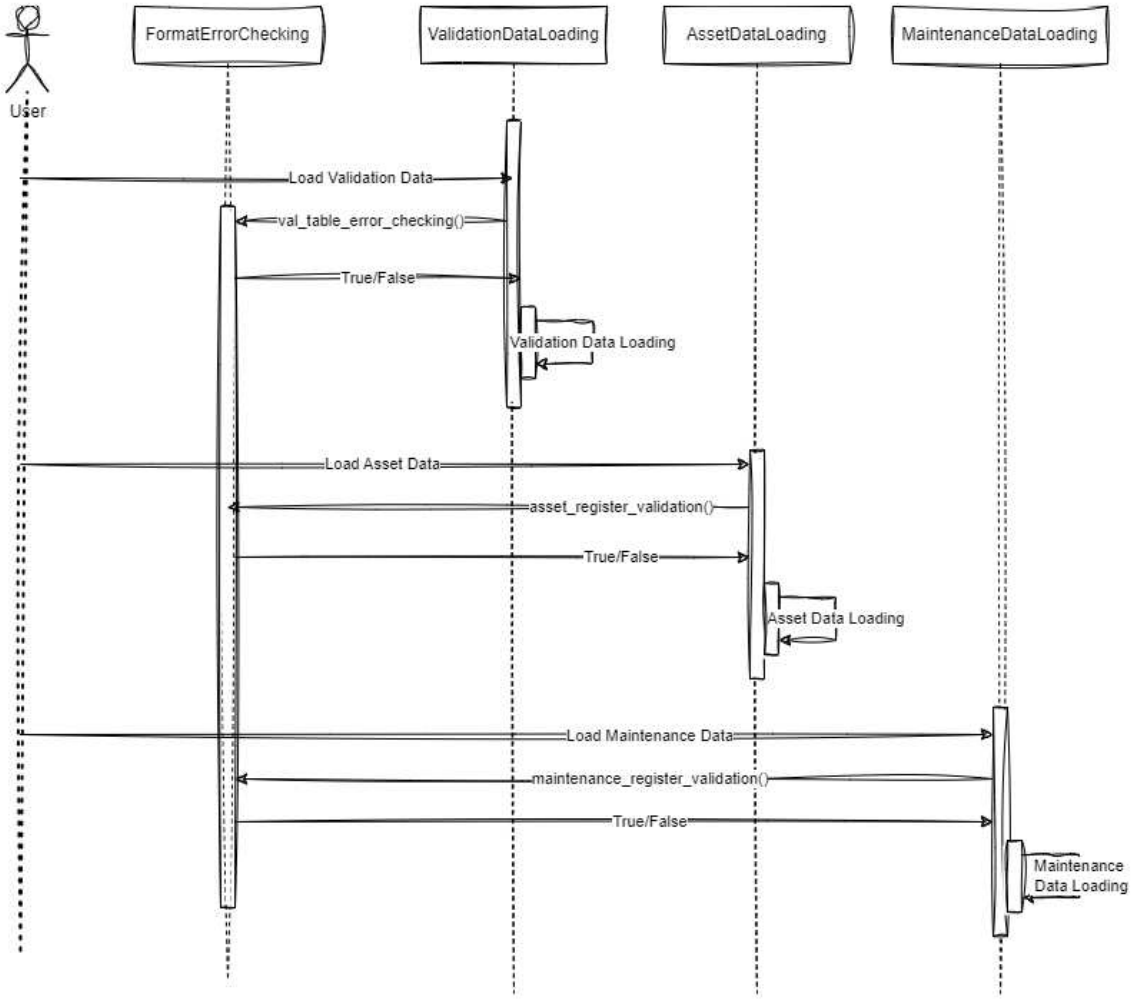


Figura 6.7: Diagramma di sequenza del software di data exchange (componente di loading)

6.4 Sviluppo della componente di data exchange

Ora che si ha una visione più chiara in merito alla struttura e alle componenti che formeranno il programma, si passerà a fornire una panoramica dettagliata sul processo di sviluppo il quale, basandosi sulla fase di progettazione appena descritta, ha portato alla realizzazione del sistema finale.

Per prima cosa verranno descritti gli strumenti software utilizzati durante la fase di programmazione.

L'intero software è stato sviluppato utilizzando come linguaggio *Python*, il quale offre un gran numero di vantaggi, soprattutto quando si parla di applicazioni legate ai database. Di seguito verranno illustrati alcuni dei pro che hanno portato alla scelta di questo linguaggio:

- *Semplicità di sintassi*: Python ha una sintassi chiara e leggibile, che facilita lo sviluppo e la manutenzione del codice.
- *Ampio supporto ai database*: Python offre supporto per una vasta gamma di database relazionali e non relazionali attraverso librerie come SQLAlchemy, Django ORM, PyMongo (per MongoDB), e molti altri. Ciò consente agli sviluppatori di lavorare con diversi tipi di database senza dover imparare nuove sintassi o paradigmi di programmazione per ciascuno di essi.
- *Librerie ORM (Object-Relational Mapping)*: Python offre librerie ORM potenti, le quali semplificano l'interazione con i database relazionali. Queste librerie consentono agli sviluppatori di utilizzare oggetti Python per rappresentare le entità del database, facilitando la gestione dei dati in modo più intuitivo.
- *Ricchezza di librerie e framework*: Python è supportato da una vasta gamma di librerie e framework che semplificano lo sviluppo di applicazioni database-driven. Ad esempio, Django e Flask sono framework web che semplificano la creazione di applicazioni web interattive che utilizzano database.
- *Community attiva e documentazione completa*: Python ha una community molto attiva e una vasta documentazione, il che facilita la risoluzione di problemi e l'apprendimento di nuove tecniche. Durante lo sviluppo, è stato, infatti, molto facile trovare risorse online (forum di discussione e tutorial) che hanno aiutato a superare svariati ostacoli.
- *Cross-platform*: Python è un linguaggio di programmazione cross-platform, il che significa che i programmi sviluppati in Python possono essere eseguiti su diverse piattaforme senza la necessità di modifiche significative al codice sorgente.
- *Integrazione con altri linguaggi e tecnologie*: Python può essere facilmente integrato con altri linguaggi di programmazione e tecnologie, consentendo di utilizzare le migliori risorse disponibili per soddisfare le esigenze specifiche del progetto.

Come accennato in precedenza, Python mette a disposizione un gran numero di librerie open source finalizzate ad agevolare lo sviluppo di applicazioni database-driven. In questo progetto i due tool di cui si è fatto maggiormente utilizzo sono stati *Pandas* e *SQLAlchemy*.

Pandas è una libreria open-source per la manipolazione e l'analisi dei dati. La sua utilità si concentra principalmente sull'elaborazione di dati strutturati, come tabelle e serie temporali. Pandas introduce due principali strutture dati, chiamate DataFrame e Series. Un DataFrame è una tabella bidimensionale con etichette sugli assi, simile a un foglio di calcolo, mentre una Series è una struttura dati unidimensionale. Queste strutture dati permettono di organizzare e manipolare facilmente i dati in modo intuitivo. Nello sviluppo del software l'utilizzo dei dataframe è stato di fondamentale importanza.

Pandas offre, anche, una serie di metodi per leggere/scrivere dati da e verso una vasta gamma di formati (tra cui CSV, Excel, SQL, HDF5, e molti altri). Questa flessibilità semplifica l'importazione e l'esportazione di dati da diverse fonti. La libreria supporta una vasta gamma di operazioni di manipolazione dei dati, come la selezione, l'indicizzazione, il raggruppamento, l'ordinamento e la fusione dei dati. Queste operazioni sono spesso eseguite in modo efficiente, consentendo a chi sviluppa il software di lavorare su grandi set di dati senza preoccuparsi eccessivamente delle prestazioni.

Pandas semplifica, anche, le operazioni di aggregazione, trasformazione e visualizzazione dei dati; ad esempio, è possibile calcolare la somma, la media, il massimo, il minimo, etc., di colonne specifiche in un DataFrame; questo tool è integrato con librerie di visualizzazione (come Matplotlib e Seaborn), le quali semplificano la creazione di grafici e visualizzazioni per esplorare i dati in modo più approfondito. Oltre a ciò, sono anche presenti diversi strumenti finalizzati alla gestione dei dati mancanti.

Il secondo strumento largamente utilizzato durante lo sviluppo è stato SQLAlchemy; si tratta di una libreria ORM (Object-Relational Mapping) la cui principale utilità risiede nella facilitazione dell'interazione tra applicazioni Python e database relazionali. Un ORM consente agli sviluppatori di lavorare con oggetti Python invece di scrivere query SQL. Questa astrazione semplifica il processo di gestione e manipolazione dei dati in un database relazionale. Di seguito, verranno elencati alcuni dei principali vantaggi che si hanno utilizzando questa libreria:

- *Object-Relational Mapping (ORM)*: SQLAlchemy offre un sistema di ORM che permette agli sviluppatori di rappresentare le tabelle del database come classi Python. Questa mappatura oggetto-relazionale semplifica notevolmente la gestione dei dati, poiché gli sviluppatori possono interagire con gli oggetti Python invece di scrivere direttamente le query SQL.
- *Astrazione del DB*: SQLAlchemy fornisce un'astrazione del database, il che significa che gli sviluppatori possono scrivere codice Python senza doversi preoccupare dei dettagli specifici del database sottostante. Questo rende il codice più portabile e meno dipendente dal tipo di database utilizzato.
- *Ricco set di funzionalità*: SQLAlchemy offre un ricco set di funzionalità che va oltre la semplice mappatura oggetto-relazionale. Esso include strumenti per gestire le transazioni, le relazioni tra tabelle, l'indicizzazione e altro ancora, semplificando la gestione avanzata dei dati in un'applicazione.
- *Ottimizzazione delle performance*: nonostante l'astrazione aggiuntiva, SQLAlchemy è progettato per offrire buone prestazioni. Gli sviluppatori possono ancora ottimizzare le query e sfruttare le caratteristiche specifiche del database, se necessario, ma possono farlo senza compromettere la chiarezza e la manutenibilità del codice.
- *Interoperabilità*: SQLAlchemy è progettato per essere utilizzato in combinazione con altri strumenti e framework Python. Ad esempio, può essere integrato facilmente

con framework web come Flask e Django, consentendo agli sviluppatori di costruire applicazioni web basate su dati in modo efficiente.

Per concludere la discussione relativa agli strumenti utilizzati e al contesto in cui è stato portato avanti lo sviuppo, bisogna specificare che l'intero codice è stato sviluppato e testato direttamente sul server Linux aziendale; ciò è stato fatto per eliminare il collo di bottiglia rappresentato dalla rete. Infatti, eseguendo il codice in locale, ed avendo il database installato nel server, la maggior parte del tempo di esecuzione sarebbe stato destinato alla comunicazione dei dati. In Figura 6.9 è possibile osservare l'insieme dei file Python caricati sul server.

Figura 6.8: File python presenti nel server

Nel codice, al fine di centralizzare la gestione dell'accesso al DB, tutti i parametri di connessione (hostname, database name, password, port, etc.) sono stati salvati in un file chiamato *params.py*, nel quale, per ognuno di essi, è associata una variabile globalmente accessibile. Oltre a ciò, in questo file, sono presenti anche tutti quei valori che sono costanti globali per la fase di caricamento (ad esempio, l'id del progetto o le cartelle in cui sono salvati i template, etc.)

Si passerà, ora, a dare una descrizione sommaria della sequenza di operazioni eseguita da ognuna delle componenti.

FormatErrorChecking.py

Come accennato in precedenza, i metodi presenti in questo codice sono finalizzati alla scansione dei valori presenti nei template di caricamento; per ogni situazione di errore, verrà generato un output sul file *ErrorLog.txt*.

In questo file, sono presenti tre metodi, ovvero uno per ognuna delle altre componenti; come è facile capire, ciascuno di essi sarà relegato ai controlli da eseguire sul gruppo di dati in questione (dati di validation, dati di asset register e dati di manutenzione).

Il primo metodo è chiamato *val_table_error_checking* ed il suo scopo principale è quello di verificare la presenza di tutti i template necessari alle operazioni di caricamento. Una volta verficata l'esistenza di tutti i file (la cui posizione è specificata in *params.py*) il metodo restituisce il valore True, indicando che tutti i controlli sono stati eseguiti con successo. Nel caso in cui uno o più file dovessero mancare all'appello, il tutto verrà segnalato con un output sul file di log ed il metodo restituirà False. In Figura 6.9 è possibile osservare due controlli eseguiti nello script.

Il secondo metodo presente in *FormatErrorChecking.py* è chiamato *asset_register_validation* e, al suo interno, vengono svolti tutti i controlli sui valori relativi ai template di Functional Location, Equipments, Assemblies e Characteristics. A questo punto, si dà per scontato che i valori relativi nelle tabelle di validazione siano già presenti nel DB (si ricorda che l'esecuzione della componente di asset data loading viene eseguita dopo il corretto caricamento delle validation table) e, di conseguenza, tutti i controlli sui valori verranno eseguiti accedendo

```
##ALL FILES EXISTENCE
error_flag = False
try:
    validation_df=pd.read_excel(params.validation_file_path, index_col=0,header=0,sheet_name=params.validation_sheet_name)
except:
    print(now.strftime("%Y-%m-%d %H:%M")+ ": Can't read validation file '"+params.validation_file_path+
          "', sheet '"+params.validation_sheet_name+"'")
    f = open("ErrorLog.txt", "a")
    f.write(now.strftime("%Y-%m-%d %H:%M")+ ": Can't read validation file '"+params.validation_file_path+
            "', sheet '"+params.validation_sheet_name+"'\n")
    f.close()
    error_flag = True
try:
    floc_data_df=pd.read_excel(params.root_dir+params.floc_file, index_col=0,header=0,sheet_name=params.data_sheet_name)
except:
    print(now.strftime("%Y-%m-%d %H:%M")+ ": Can't read functional location data file '"+params.floc_file+"', sheet '"
          +params.data_sheet_name+"'")
    f = open("ErrorLog.txt", "a")
    f.write(now.strftime("%Y-%m-%d %H:%M")+ ": Can't read functional location data file '"+params.floc_file+"', sheet '"
            +params.data_sheet_name+"'\n")
    f.close()
    error_flag = True
```

Figura 6.9: Esempio di due controlli relativi all'esistenza dei file

direttamente alla base di dati, senza dover consultare i template (il che sarebbe molto più oneroso). Per la connessione al database, in aggiunta a SQLAlchemy, viene utilizzata la libreria *Psycopg2*, la quale fornisce un'interfaccia per comunicare con database PostgreSQL, gestendo anche la connessione al database, l'esecuzione di query SQL, il recupero dei risultati e la gestione delle transazioni.

Per ricapitolare, per la gestione della comunicazione con il DB, SQLAlchemy viene usato principalmente per facilitare il trasferimento degli oggetti Python (Pandas datafram) da e verso il DB (ORM), mentre Psycopg2 viene utilizzato in quanto permette un'esecuzione di query string più semplificata.

La prima operazione svolta dal metodo *asset_register_validation* è, quindi, quella di recuperare dal DB tutti i valori necessari alle operazioni di validazione. In Figura 6.10 è possibile osservare sia la connessione al database tramite connection string, che alcune query per il fetch dei valori di validazione.

```
def asset_register_validation():
    error_flag = False


    infile_error_str = ""
    db_connection_str = f'postgresql://{params.db_params["user"]}:{params.db_params["password"]}@localhost:{params.db_params["port"]}/

    conn = psycopg2.connect(**params.db_params)
    curs = conn.cursor()

    #### VALIDATION TABLES FROM DB
    curs.execute("SELECT equip_cat FROM equip_cat where project=%s;",(params.project,))
    res = list(curs.fetchall())
    equip_cat_list = [row[0] for row in res]

    curs.execute("SELECT obj_type FROM obj_type where project=%s;",(params.project,))
    res = list(curs.fetchall())
    obj_type_list = [row[0] for row in res]
```

Figura 6.10: Connessione al DB e lettura di alcuni valori di validazione

Una volta letti questi valori dal DB, il metodo, accedendo al template di caricamento, verifica che:

- non siano presenti valori Null nei campi che, nel DB, sono not nullable;
- non siano presenti valori duplicati per i campi chiave;

- ogni campo associato ad una tabella di validazione abbia solo valori che ricadono nel set presente nel database;

Due controlli molto importanti sono quelli relativi ai campi superior functional location e superior equipment; la loro validazione, infatti, non può fare affidamento su valori presenti nel database, e quindi viene eseguita verificando che tutti i valori non nulli facciano riferimento ad identificatori esistenti. Un ulteriore controllo chiave è quello dell'esistenza di tutte le functional location a cui gli equipment fanno riferimento; in questa fase, non essendo le sedi tecniche ancora caricate nel DB, tutta la validazione deve essere effettuata consultando la colonna del dataframe contenente i dati delle functional location. Questi controlli evitano che, in fase di caricamento, si faccia riferimento ad attrezzature o sedi tecniche non esistenti. Il codice relativo a queste verifiche è illustrato in Figura 6.11.

```
#Superior functional location
invalid_sup_flocs = df[~df['TPLMA'].isin(df['TPLNR'])]['TPLMA'].dropna()
if len(invalid_sup_flocs.values)>0:
    error_flag = True
    print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid superior functional locations found: "+str(invalid_sup_flocs.values)+" \n ")
    infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ": Invalid superior functional locations found: "+str(invalid_su
#Checking all superior equipments exist
if not(df['HEQUI'].dropna().isin(df['EQUNR'].unique()).all()):
    error_flag = True
    invalid_supeq = df[~df['HEQUI'].isin(df['EQUNR'].unique())]['HEQUI'].dropna()
    if len(invalid_supeq.values)>0:
        print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid superior_equipment found: "+str(invalid_supeq.values)+" \n ")
        infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ": Invalid superior_equipment found: "+str(invalid_supeq.valu
```

```
#Checking all equipment flocs exist in the floc file
floc_df=pd.read_excel(params.root_dir+params.floc_file, index_col=1,header=2,sheet_name=params.data_sheet_name)
floc_df[floc_df.index.name] = floc_df.index
if not(df['TPLNR'].isin(floc_df['TPLNR'].unique().tolist()).all()):
    error_flag = True
    invalid_flocs = df[~df['TPLNR'].isin(floc_df['TPLNR'].unique().tolist())]['TPLNR']
    if len(invalid_flocs.values)>0:
        print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid equipment_functional_locations found: "+str(invalid_flocs.values)+" \n ")
        infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ": Invalid equipment_functional_locations found: "+str(inval
```

Figura 6.11: Validazione dei campi superior functional location/superior equipment e functional location di riferimento

Gli ultimi controlli di questo metodo riguardano il template delle caratteristiche; le validazioni eseguite in questa parte del codice servono per assicurarsi che non ci siano coppie attrezzatura-caratteristica duplicate e che tutte le attrezzature a cui si fa riferimento siano presenti nei rispettivi template.

Il terzo metodo presente in questo primo file è chiamato *maintenance_register_validation* e, come è possibile capire dal nome, è finalizzato a svolgere tutti i controlli sui valori presenti nei template riconducibili ad elementi del maintenance register.

È importante tenere in considerazione che, arrivati a questo punto, si dà per scontato che sia i dati di validazione che quelli di asset sono stati caricati con successo nel database; di conseguenza tutti i controlli basati su questi valori sfrutteranno i dati presenti nel DB. Proprio per questo, la prima operazione che il metodo svolge, come nel caso precedente, è quella di leggere dalla base di dati tutti i valori necessari a svolgere i controlli.

Il primo gruppo di controlli riguarda il template contenente i dati relativi alle task list, alle operazioni e ai package. Per quanto riguarda la validazione sui cicli di manutenzione, oltre ai soliti controlli inerenti i valori nulli e i duplicati nei campi chiave, non ci sono particolarità degne di essere approfondite. Passando alle verifiche fatte sui valori nel template delle operazioni, due controlli importanti in questa sezione riguardano l'esistenza delle task list e dei package associati ad ogni operazione (Figura 6.12). Da notare come viene eseguito il controllo sui pacchetti di frequenza. È presente un ciclo for che scorre le colonne del template contenenti gli identificatori dei pacchetti (colonne che vanno da MARK01 a MARK33) e,

per ognuno di essi, verifica l'esistenza consultando l'elenco dei package (ripresi dal relativo template).

```
#Checking that all reference to group-groupcounter exist
invalid_values = []
tasklist_list = tasklist_data_df[['PLNNR','PLNAL']].drop_duplicates().values.tolist()
for i, row in enumerate(operation_data_df.iterrows()):
    tasklist = row[1][['PLNNR','PLNAL']].values.tolist()
    if not(tasklist in tasklist_list):
        invalid_values.append(row[1])
if(len(invalid_values)>0):
    error_flag = True
    invalid_values_df = pd.DataFrame(columns=['PLNNR','PLNAL'], data=invalid_values)
    invalid_values = invalid_values_df.drop_duplicates()
    print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid operation task_lists found : \n "+str(invalid_values.values)+" \n ")
    infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ":  Invalid operation items task_lists found : "+s
```

```
#Checking that all reference to packages exist
package_data_df=pd.read_excel(params.root_dir+params.tasklist_oper_file, index_col=0,header=1,sheet_name="PACKAGES")
package_data_df[package_data_df.index.name] = package_data_df.index
packages_list = package_data_df[['MARK','STRAT']].drop_duplicates().values.tolist()
invalid_values = []
for i, row in enumerate(operation_data_df.iterrows()):
    for j in range(1,33):
        pack_col_name = ""
        if j<10:
            pack_col_name = 'MARK0'+str(j)
        else:
            pack_col_name = 'MARK'+str(j)
        if not(pd.isnull(row[1][pack_col_name])):
                package = row[1][[pack_col_name,'STRAT']].values.tolist()
                if not(package in packages_list):
                    invalid_values.append(package)
if(len(invalid_values)>0):
    error_flag = True
    invalid_values_df = pd.DataFrame(columns=['MARK','STRAT'], data=invalid_values)
    invalid_values = invalid_values_df.drop_duplicates()
    print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid package reference found : \n "+str(invalid_values.values)+" \n ")
    infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ":  Invalid package reference found : "+str(in
```

Figura 6.12: Validazione dei campi package e task list nel template relativo alle operazioni

Il secondo gruppo di controlli eseguiti da questo metodo riguarda i measuring point; in questo caso i controlli più importanti riguardano l'esistenza dell'attrezzatura fisica l'esistenza della caratteristica misurata.

Il terzo insieme di controlli è inerente ai template di maintenance items, maintenance plans ed object list. Per cioò che concerne la validazione dei valori legati alle posizioni di manutenzione, gli step più rilevanti riguardano i seguenti aspetti:

- controllo dell'esistenza dei reference equipment e reference functional location;
- controllo dell'esistenza del ciclo di manutenzione associato;
- controllo dell'esistenza del piano di manutenzione associato.

Parlando, ora, della validazione svolta sul template della object list, come è facile immaginare, gli unici controlli da eseguire riguardano l'esistenza di tutti gli oggetti e di tutte le posizioni di manutenzione (Figura 6.13).

Con questo si può concludere la discussione legata alla componente FormatErrorChecking. Nelle prossime sezioni, verranno illustrati i metodi che eseguono materialmente il caricamento dei dati nel DB.

ValidationDataloading.py

Come illustrato nel diagramma di sequenza, questo script, prima di poter iniziare il caricamento dei dati, deve verificare che non siano presenti errori nei valori, e, per far ciò,

```
#Checking all equipment and mitems exist
if not(obj_list_data_df['EQUNR'].dropna().isin(equips_list).all()):
    error_flag = True
    invalid_values = obj_list_data_df[~obj_list_data_df['EQUNR'].isin(equips_list)]['EQUNR'].dropna()
    if len(invalid_values.values)>0:
        print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid object list equipment found: "+str(invalid_values.values)+" \n ")
        infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ": Invalid object list equipment found: "+str(invalid
if not(obj_list_data_df['WAPOS'].dropna().isin(mitem_data_df['WAPOS'].values.tolist()).all()):
    error_flag = True
    invalid_values = obj_list_data_df[~obj_list_data_df['WAPOS'].isin(mitem_data_df['WAPOS'].values.tolist())]['WAPOS'].dropna()
    if len(invalid_values.values)>0:
        print(now.strftime("%Y-%m-%d %H:%M")+ ": Invalid object list maintenance item found: "+str(invalid_values.values)+" \n ")
        infile_error_str = infile_error_str+now.strftime("%Y-%m-%d %H:%M")+ ": Invalid object list maintenance item found: "+str(
```

Figura 6.13: Validazione dei campi relativi al template della object list

richiama il metodo *val_table_error_checking* del modulo *FormatErrorChecking* appena descritto (Figura 6.14). Se questa serie di controlli va a buon fine (ovvero, viene restituito valore True), si potrà procedere con il caricamento.

```
if not(FormatErrorChecking.val_table_error_checking()):
    if input('Validation tables have been checked! \nDo you want to load the Validation Tables into the CMMS? (y to continue) ') == 'y':

        #DATA LOADING
```

Figura 6.14: Chiamata al metodo di validazione in FormatErrorChecking.py

In questi programmi di caricamento è stato di fondamentale importanza, rispettare un ordinamento preciso nelle operazioni di inserimento nel DB. Questo perché, se dovessero essere inserite delle tabelle che fanno riferimento a dati non ancora caricati, l'intera operazione sarebbe arrestata a causa della violazione dei vincoli di chiave esterna. Di conseguenza, per ogni tabella che si vuole caricare, bisogna essere certi che tutti i dati a cui questa fa riferimento siano già presenti nel DB. In verità, come vedremo più avanti, nel caso del caricamento del template di validazione, questa stretta necessità viene meno. La sequenza di caricamento è stata la seguente:

- Plants;
- Locations;
- Catalog Profile;
- Order Type;
- Company;
- Control Key;
- Cost Center;
- Criticality;
- Effect;
- UoM;
- User Status;
- WBS Element;

- Work Center;
- Floc Category;
- Equipment Category;
- System Condition;
- Priority;
- Structure Indicator;
- Task List Type;
- Task List Usage;
- Task List Status;
- Maintenance Strategy;
- Maintenance Plan Category;
- Maintenance Item Category;
- Standard Text Key;
- Classes;
- Characteristics;
- Characteristics to Classes;
- Object Type;
- Maintenance Activity Type.

Una nota degna di attenzione riguarda il comportamento di questo modulo quando vengono riscontrati degli errori; infatti, trattandosi di dati finalizzati alla sola validazione, si è deciso di non interrompere il caricamento qualora qualche operazione di inserimento dovesse restituire degli errori. In questo modo, qualora si dovessero aggiungere dei valori in un secondo momento, sarà possibile eseguire più volte l'operazione di caricamento, modificando ogni volta i valori nel template. La consistenza del database sarà garantita ogni volta dai vincoli di chiave, i quali non permetteranno di inserire valori duplicati (ma solo i record nuovi). In Figura 6.15 si può notare la presenza del comando *pass* nel blocco di gestione delle eccezioni; esso permette di passare al caricamento del record successivo (senza bloccare l'esecuzione) ogniqualvolta ci si trova in condizione di errore.

AssetDataLoading.py

Dopo aver caricato con successo i dati relativi alle validation table, lo step successivo nell'esecuzione è quello di caricare i dati relativi agli elementi dell'asset register. Per far ciò deve essere eseguito il codice presente in *AssetDataLoading.py*. Come per il modulo precedente (ed il successivo) anche in questo caso, per procedere al caricamento, bisogna che prima siano superati i controlli presenti nel metodo *asset_register_validation*; una volta che quest'ultimo avrà ritornato il valore true, indicando l'assenza di errori, la sequenza che verrà seguita nel caricamento è la seguente:

- Functional Locations;

```
#Criticality
transfer_df = validation_df[['Criticality','Criticality desc','project']]
transfer_df = transfer_df[pd.notna(transfer_df["Criticality"])]
transfer_df.rename(columns={'Criticality':'criticality_code', 'Criticality desc':'desc'}, inplace=True)
print(transfer_df)
for i in range(len(transfer_df)):
    try:
        transfer_df.iloc[i:i+1].to_sql('criticality',engine, schema='ar_mr', if_exists='append', index=False)
    except Exception as e:
        print(e)
        f = open("ErrorLog.txt", "a")
        f.write(now.strftime("%Y-%m-%d %H:%M")+" : "+str(e)+"'\n \n")
        f.close()
        pass

#Effect
transfer_df = validation_df[['Effect','Effect desc','project']]
transfer_df = transfer_df[pd.notna(transfer_df["Effect"])]
transfer_df.rename(columns={'Effect':'effect_code', 'Effect desc':'desc'}, inplace=True)
print(transfer_df)
for i in range(len(transfer_df)):
    try:
        transfer_df.iloc[i:i+1].to_sql('effect',engine, schema='ar_mr', if_exists='append', index=False)
    except Exception as e:
        print(e)
        f = open("ErrorLog.txt", "a")
        f.write(now.strftime("%Y-%m-%d %H:%M")+" : "+str(e)+"'\n \n")
        f.close()
        pass
```

Figura 6.15: Esempi di caricamento di valori relativi al template di validazione

- Equipments;
- Assemblies;
- Characteristics Values.

Di notevole interesse sono i due frammenti di codice che realizzano il caricamento delle functional location e degli equipment; questo perché, nel capire come ottenere il comportamento desiderato, sono stati incontrate due problematiche principali.

La prima riguarda la gestione dei campi superior equipment e superior functional location; infatti, non si ha la certezza che, nel template, i record rispettino un ordine di inserimento corretto. Ad esempio, se si immagina di inserire un equipment nel database, ma il campo relativo al suo superiore fa riferimento ad un'attrezzatura non ancora inserita (in quando presente solo più avanti nell'elenco), allora si avrà una condizione di errore (violazione della foreign key). Di conseguenza è stato necessario implementare una logica finalizzata alla gestione automatizzata di questi potenziali errori. L'approccio seguito, nel caso delle functional location, è il seguente (Figura 6.17):

- Viene avviato un ciclo while che itera finché nel dataframe relativo alle functional location sono presenti record.
- Ad ogni iterazione del while viene fatto partire un ciclo for che scorre tutto il dataframe relativo alle functional location.
- Per ogni functional location viene tentato l'inserimento; questo comando andrà ad aumentare il last_value della object sequence definita nel database.
- Se l'inserimento va a buon fine, allora il record viene eliminato dal dataframe, mentre, se si ha un errore per violazione della foreign key, la object sequence viene riportata al valore precedente.

In questo modo, ad ogni iterazione del while esterno, verranno inserite alcune sedi tecniche; quelle rimanenti nel dataframe sono le functional location per le quali non è ancora stata inserita la superiore. Onde evitare situazioni di stallo, ad ogni iterazione del while si controlla che la dimensione del dataframe sia diminuita (il che vuol dire che alcune sedi tecniche sono state caricate, permettendo all'iterazione successiva di caricare tutte le "figlie" non ancora inserite).

Un meccanismo analogo viene applicato durante il caricamento degli equipment, con la differenza che, in questo caso, è stato necessario, come accennato in precedenza, risolvere anche una seconda problematica. Questo scoglio ha a che fare con la discrepanza fra gli identificatori usati nell'ambito del progetto (presenti nei template) e quelli usati nel DB (generati tramite la sequenza di object). In poche parole, quando alcuni tipi di oggetti (ad esempio, equipment, maintenance plan etc.) vengono caricati nel DB, ad essi viene associato un identificatore sequenziale che dovrà essere ripreso al fine di rispettare i riferimenti fra record.

Nel caso del caricamento degli equipment, questa problematica si è palesata nel momento in cui è stato necessario capire se un superior equipment era presente nel DB o meno; per far ciò, infatti, bisogna eseguire una query che vada a ritrovare l'identificatore originale (per i quali sono previsti dei campi dedicati, come descritto nella sezione relativa alla strauttura delle tabelle presenti nel DB), ritornando il valore della chiave generata automaticamente dal sistema. In Figura 6.16 si può osservare il meccanismo di caricamento degli equipment; la logica è la stessa usata nel caso delle functional location, con la differenza che, in questo caso, prima di inserire un equipment (e quindi eliminarlo dal dataframe degli oggetti rimanenti), è necessario controllare che il superior equipment sia già nel database; per far ciò, ad ogni iterazione del ciclo while esterno, viene eseguita una query finalizzata a leggere gli ID generati e gli ID originali degli equipment caricati. Nel momento in cui dovrà essere inserito un equipment (nel ciclo for interno), al campo superior dovrà essere associato il valore dell'identificatore generato nel DB e ritrovato grazie all'interrogazione.

Il metodo di caricamento dei dati di asset si conclude con l'inserimento dei valori relativi alle caratteristiche delle attrezzature. In merito al modo in cui questa operazione viene svolta, l'unico punto di interesse riguarda il rintracciamento degli ID sequenziali degli equipment; infatti, come nei due script precedenti, anche in questo caso i riferimenti sono basati sugli identificatori originali, il che porta alla necessità di eseguire interrogazioni finalizzate a rintracciare a quali ID sequenziali corrispondono i vari ID originali. In Figura 6.18 si può osservare il codice di caricamento di questi valori.

MaintenanceDataLoading.py

La componente che chiude l'insieme di script legati alla parte di caricamento del software di data exchange è chiamata *MaintenanceDataLoading*; come è facile capire dal nome, in questo caso si avrà a che fare con il caricamento dei dati presenti nei template riconducibili ad elementi del maintenance register. Come nelle due componenti precedentemente descritte, è necessario fare attenzione alla sequenza seguita per il caricamento, onde evitare riferimenti a dati di cui non è ancora stato effettuato l'upload. L'ordine seguito è il seguente:

- Measuring Points;
- Task List;
- Packages;
- Operation;
- Operation to Package;

```
#Pushing EQUIPMENT data in the database
last_valid_obj = 0
last_df_dim = -1
index_to_remove = []
while(df.shape[0]>0): #While loop that stops only when all rows are inserted (We know from the FormatErrorChecking that all the
    print(df)
    curs.execute("SELECT equip_xlsx_code FROM equipment WHERE project=%s",(project,))
    res = list(curs.fetchall())
    equi_code_list = [row[0] for row in res]
    curs.execute("SELECT equip_xlsx_code, objectid FROM equipment WHERE project=%s",(project,))
    equip_list = list(curs.fetchall())

    for i in range(len(df)): #For loop that iterate over the rows for wich the superior equip was not found
        trasfer_df = df.iloc[i:i+1]
        if not(pd.isna(df.iloc[i:i+1]['sup_equip'].values[0])) and not(df.iloc[i:i+1]['sup_equip'].isin(equi_code_list).all()):
            continue
        if not(pd.isna(df.iloc[i:i+1]['sup_equip'].values[0])) and (df.iloc[i:i+1]['sup_equip'].isin(equi_code_list).all()):
            for equip_row in equip_list: #finding the generated ID
                if df.iloc[i:i+1]['sup_equip'].values[0] == equip_row[0]:
                    trasfer_df['sup_equip'] = equip_row[1]
        try:
            #Insert row and remove it from the df
            trasfer_df.to_sql('equipment',engine, schema='ar_mr', if_exists='append', index=False)
            curs.execute("SELECT last_value FROM objectid_seq;")
            last_valid_obj = curs.fetchone()
            index_to_remove.append(df.iloc[i:i+1].index.values[0])
            conn.commit()
        except IntegrityError as e:
            #Skip
            curs.execute("ALTER sequence objectid_seq RESTART with %s",(last_valid_obj[0]+1,))
            conn.commit()
            continue
    for idx in index_to_remove:
        df.drop(index = idx, axis=1, inplace = True)

    if df.shape[0] == last_df_dim:
        f = open("ErrorLog.txt", "a")
        f.write(now.strftime("%Y-%m-%d %H:%M")+ ": Equipment data loading was stuck \n ")
        f.close()
        sys.exit(now.strftime("%Y-%m-%d %H:%M")+ ": Equipment data loading was stuck \n ")


    index_to_remove=[]
```

Figura 6.16: Script di caricamento degli equipment

- Maintenance Plan;
- Maintenance Item;
- Object List.

I caricamenti presenti in questa componente sono stati implementati in maniera più diretta rispetto a quelli descritti nella sezione dell'asset register; questo perchè, in questo caso, non è stato necessario preoccuparsi dei riferimenti ciclici (ad esempio, equipment-superior equipment o functional location-superior functional location etc.), e, di conseguenza, l'unica problematica necessaria di particolare attenzione è stata quella relativa alla discrepanza fra gli identificatori progettuali e quelli generati dal DB in fase di loading. In particolare, questa criticità è stata riscontrata nei seguenti riferimenti:

- Measuring Point - Equipment (object id dell'attrezzatura relativa al punto di misura);
- Measuring Point - Functional location (object id della sede tecnica relativa al punto di misura);

```
#Pushing FUNCTIONAL LOCATION data in the database
last_valid_obj = 0
last_df_dim = -1
index_to_remove = []
while(df.shape[0]>0): #While loop that stops only when all rows are inserted (We know from the Format
    print(df)
    for i in range(len(df)): #For loop that iterate over the rows for wich the superior floc was not
        try:
            #Insert row and remove it from the df
            df.iloc[i:i+1].to_sql('fun_loc',engine, schema='ar_mr', if_exists='append', index=False)
            curs.execute("SELECT last_value FROM objectid_seq;")
            last_valid_obj = curs.fetchone()
            index_to_remove.append(df.iloc[i:i+1].index.values[0])
            conn.commit()
        except IntegrityError as e:
            #Skip
            curs.execute("ALTER sequence objectid_seq RESTART with %s",(last_valid_obj[0]+1,))
            conn.commit()
            continue
    for idx in index_to_remove:
        df.drop(index = idx, axis=1, inplace = True)

    if df.shape[0] == last_df_dim:
        f = open("ErrorLog.txt", "a")
        f.write(now.strftime("%Y-%m-%d %H:%M")+ ": Functional location data loading was stuck \n ")
        f.close()
        sys.exit(now.strftime("%Y-%m-%d %H:%M")+ ": Functional location data loading was stuck \n ")
    index_to_remove=[] #refreshing the indexes to remove
```

Figura 6.17: Script di caricamento delle functional location

```
#CHARACTERISTICS VALUES LOADING

df=pd.read_excel(char_file_path, index_col=0,header=2,sheet_name=data_sheet_name)
df[df.index.name] = df.index
#char value data loading
curs.execute("SELECT * FROM equipment where project=%s",(project,)) #Equipments must have been loaded at
equipments = curs.fetchall()

values_df = df[['EQUNR','MNAME','MWERT']]
values_df['project'] = project
values_df.rename(columns={'MWERT':'value','MNAME':'char_code'},inplace=True)

for i, row in enumerate(values_df.iterrows()):
    for equip in equipments:
        if str(equip[32]) == str(row[1]['EQUNR']):
            transfer_df = values_df.iloc[i:i+1]
            transfer_df['objectid'] = equip[31]
            transfer_df.drop('EQUNR',axis=1,inplace=True)
            try:
                transfer_df.to_sql('char_value',engine, schema='ar_mr', if_exists='append', index=False)
            except IntegrityError:
                pass
            break

print("Characteristics data loaded")
```

Figura 6.18: Script di caricamento dei valori delle caratteristiche legate agli equipment

- Maintenance Plan - Maintenance Item (id sequenziale del piano di riferimento per la posizione di manutenzione);
- Object List - Maintenance Item (id sequenziale della posizione a cui fa riferimento

l'elemento della object list);

- Object List - Equipment/Functional Location (object id dell'attrezzatura/sede tecnica da associare alla posizione di manutenzione in questione).

6.5 Esempio di caricamento

A conclusione di questa sezione dedicata alla componente di data exchange, verrà illustrato un esempio relativo al caricamento di un progetto nel database; i dati che verranno caricati seguono lo standard JobPlan, il quale si distacca notevolmente dalla rappresentazione dei dati in SAP (sulla quale il sistema descritto in questa tesi si basa).

In questi casi, dove sono presenti notevoli differenze fra le rappresentazioni dei dati è necessario sfruttare il template di caricamento per adattare i dati allo standard dettato dalle tabelle del DB. Nel caso dello standard aziendale JobPlan le principale discrepanze sono riscontrabili nei dati relativi al maintenance register; al contrario, per quanto riguarda l'asset register, si assiste ad una buona sovrapposizione dei formati.

Verranno, ora, analizzati gli adattamenti effettuati sulla forma dei dati di progetto, finalizzati al corretto caricamento di tali informazioni nel database. La compilazione del template di validazione è avvenuta in parallelo alla compilazione degli altri; ciò vuol dire che, man mano che i template di caricamento venivano riempiti, si aggiungevano nella scheda di validazione tutti quei valori ancora non presenti.

Si andrà, ora, ad illustrare il procedimento seguito passo per passo. Come accennato in precedenza, per quanto riguarda i dati di asset, non ci sono state particolari difficoltà di adattamento; i primi due template ad essere compilati sono stati quelli relativi agli equipment e alle functional location. In Figura 6.19 è possibile osservare alcune righe del template di caricamento degli equipment; come è facile notare, sono molte le colonne che non contengono dati. Ciò avviene perché, non avendo una sovrapposizione perfetta delle informazioni, alcuni campi risultano mancanti.

Fortunatamente, nel caso del progetto in questione, nessuno dei campi con vincolo di Not Null presentava valori nulli; nel caso in cui, invece, tali colonne avessero presentato delle mancanze, l'unica possibilità sarebbe stata quella di riempirle con dei valori fittizi. In queste circostanze, è sempre importante tenere a mente che, al fine di mantenere chiarezza nel DB, è necessario scegliere dei valori fittizi facili da rintracciare e che non rischino di confondersi con dei valori reali.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Equipment	Category	Description	User Status	Object Type	Authorization group	Start-up Date [YYYYMMDD]	Manufacturer	Manufacturer Country	Construction Year	Construction Month	Manufacturer model number	Manufacturer serial number	Vendor	Criticality	Effect	Asset	Catalog Profile	Functional Location	Superior Equipment	Construction type	Equipment Tag Number
2		Validation Table 32		Validation Table 33	Validation Tables	Validation Table 34			Validation Table 35					Validation Table 36	Validation Table	Validation table 39	Validation Table	Validation Table 46				
3	10	1	40	4	10	4	8	30	2	4	2	20	30	10	1	3	12	9	30	10	18	31
4	EQUNR	EQTY	EQKTX	STAT	EQAI	BEGRU	INBDT	HERST	HERLD	BAUJJ	BAUMM	TYPBZ	SERGE	ELIEF	ABCK	ZEFC	ANLN	RBNR	TPLNR	HEQU	SUBMT	EQFNR
1152	10000442	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000442
1153	10000443	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000443
1154	10000444	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000444
1155	10000445	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000445
1156	10000446	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000446
1157	10000447	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000447
1158	10000448	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000448
1159	10000449	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000449
1160	10000450	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000450
1161	10000451	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000451
1162	10000452	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000452
1163	10000453	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000453
1164	10000454	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000454
1165	10000455	E	Sezione 10	MONT	SY06			FINDER POMPE				not defined	not defined		N	N			PR-TAFS-10-10			10000455
1166	10000705	E	Sezione 10	MONT	SY30			DI&CI				not defined	not defined		N	N			PR-TAFS-10-10			10000705
1167	10000706	E	Sezione 10	MONT	SY30			not defined				not defined	not defined		N	N			PR-TAFS-10-10			10000706
1168	10000782	E	Sezione 10	MONT	SY06			NETZSCH				not defined	not defined		N	N			PR-TAFS-10-10			10000782
1169	10000783	E	Sezione 10	MONT	SY03			not defined				not defined	not defined		N	N			PR-TAFS-10-10			10000783
1170	10012708	E	Sezione 10	MONT	SY36			not defined				not defined	not defined		K	N			PR-TAFS-10-10			10012708
1171	10012709	E	Sezione 10	MONT	SY36			not defined				not defined	not defined		K	N			PR-TAFS-10-10			10012709
1172	10012710	E	Sezione 10	MONT	SY36			not defined				not defined	not defined		K	N			PR-TAFS-10-10			10012710
1173	10012711	E	Sezione 10	MONT	SY36			not defined				not defined	not defined		K	N			PR-TAFS-10-10			10012711
1174	10012712	E	Sezione 10	MONT	SY36			not defined				not defined	not defined		K	N			PR-TAFS-10-10			10012712

Figura 6.19: Template di caricamento degli equipment

È importante notare che gli identificatori degli equipment in questo template sono gli ID di progetto, i quali, in fase di caricamento nel DB, saranno inseriti nel campo *equip_xls_code*,

con il quale sarà possibile ricostruire i riferimenti (logiche descritte nelle sezioni riguardanti gli script di caricamento).

Terminato l'inserimento degli equipment, è stato compilato il template delle sedi tecniche; nel caso di questo progetto non erano presenti assembly, e di conseguenza il template è rimasto vuoto. Come accennato in precedenza, in parallelo alla compilazione di queste schede, è stato portato avanti l'inserimento delle informazioni di validazione nel rispettivo template (Figure 6.20 e 6.21).

	C	D	E	F	G	H	I	J	K	L	M
2	Location	Location desc	Location Plant	Catalog Profile	Catalog Profile desc	Company	Company desc	Control key	Control key desc	Cost Center	Cost Center desc
3	SY4L	SY4L structure	SY20	dummy	dummy	SY01	società SY01	ESTE	ESTE Ctrl Key	PL100	PL100
4	SY4M	SY4M structure	SY20							PL201	PL201
5										TF501	TF501
6										TF502	TF502
7										TF503	TF503
8										TF551	TF551
9										TF552	TF552
10										TF553	TF553
11										TF554	TF554
12										TF854	TF854
13										TF855	TF855
14											
15											
16											
17											

Figura 6.20: Template di caricamento di alcune informazioni di validazione

	A	B	C	D	E	F	G
1	Class	Charachteristic	Charachteristic description	Data Type	Number of chars	Decimal places	UoM
2	PM_M_P	PM_MAX_PRESS	Presione massima	NUM	15	3	bar
3	PM_M_P	PM_HEAD	Prevalenza	NUM	15	3	m
4	PM_M_P	PM_HEIGHT	Altezza	NUM	15	2	m
5	PM_M_P	PM_MAXIMUM_RATED_CAPACITY	Portata massima	NUM	15	3	m^3/h
6	PM_M_P	PM_MINIMUM_RATED_CAPACITY	Portata massima	NUM	15	3	m^3/h
7	PM_M_P	PM_RATED_POWER	Potenza nominale	NUM	15	3	kVA
8	PM_M_P	PM_MAX_RATED_CAPACITY_LITH	Portata massima	NUM	15	3	l/h
9	PM_M_P	PM_MIN_RATED_CAPACITY_LITH	Portata massima	NUM	15	3	l/h
10	PM_M_S	PM_VOLUME	Volume	NUM	15	2	m^3
11	PM_M_S	PM_MATERIALE	Materiale	CHAR	15	0	UL
12	PM_M_S	PM_DIAM	Diametro	NUM	15	2	m
13	PM_M_S	PM_HEIGHT	Altezza	NUM	15	2	m
14	PM_M_C	PM_VOLUME	Volume	NUM	15	2	m^3
15	PM_M_C	PM_HEIGHT	Altezza	NUM	15	2	m
16	PM_M_C	PM_MATERIALE	Materiale	CHAR	15	0	UL
17	PM_M_F	PM_VOLUME	Volume	NUM	15	2	m^3
18	PM_M_F	PM_HEIGHT	Altezza	NUM	15	2	m
19	PM_M_F	PM_MATERIALE	Materiale	CHAR	15	0	UL
20	PM_M_T	PM_VOLUME	Volume	NUM	15	2	m^3
21	PM_M_T	PM_MATERIALE	Materiale	CHAR	15	0	UL
22	PM_M_PK	PM_MAXIMUM_RATED_CAPACITY	Portata massima	NUM	15	3	m^3/h

Figura 6.21: Template di caricamento delle informazioni di validazione relative alle caratteristiche per classe di oggetti

Terminata la fase di inserimento delle informazioni di asset register, si sono presentati dei problemi quando si è giunti a dover caricare la parte di maintenance register. Lo standard JobPlan, infatti, sotto questo aspetto, si distacca notevolmente dalla rappresentazione presente nel DB; quello che si ha è una rappresentazione semplificata a due livelli: il livello *JobCard* ed il livello *operation*. Essenzialmente, un jobcard viene definito come un insieme di operazioni che possono essere associate ad un certo oggetto. Di conseguenza, la differenza fra i due standard è rappresentata dal fatto che, nel primo, abbiamo JobCard ed Operazioni, mentre nel secondo abbiamo Piano, Posizioni/Cicli ed operazioni.

Al fine di risolvere tale discrepanza, quindi, è stato necessario riempire uno dei tre livelli dettati dal DB con delle informazioni fittizie; la soluzione più facile è stata quella di inserire un unico piano di manutenzione in grado di raggruppare tutti i job card, i quali, a loro volta, sono stati rappresentati come cicli di manutenzione; per quanto riguarda le posizioni,

esse sono state generate seguendo i cicli di manutenzione. In Figura 6.22 si è cercato di rappresentare graficamente la conversione effettuata; sfruttando le informazioni prese dallo standard JobPlan si sono generati record nel template di caricamento in modo che il significato semantico legato ai dati fosse lo stesso. Riassumendo, quindi, dopo aver introdotto il piano di manutenzione fittizio, la mappatura è stata la seguente:

- Job Card - Maintenance Item;
- Job Card - Task List;
- Operation - Operation.

In un certo senso, è come se si fosse diviso l'oggetto Job Card in due elementi (posizioni e cicli) legati da un rapporto uno ad uno.

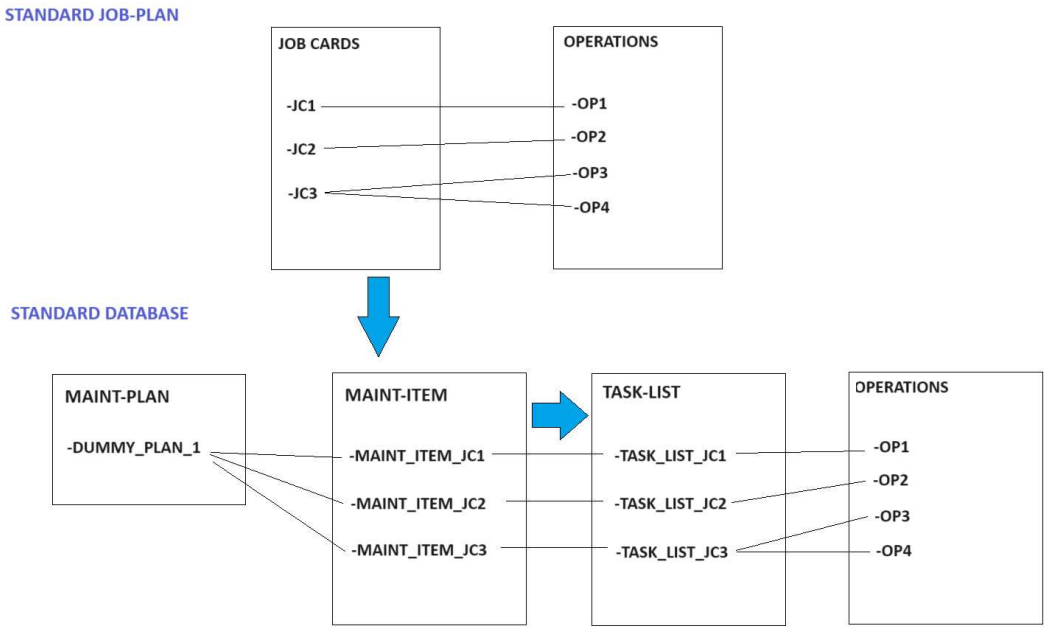


Figura 6.22: Illustrazione della conversione fra standard JobPlan e standard del database

Terminata questa fase di compilazione dei vari template, si è potuto procedere con l'esecuzione del codice delle varie componenti di caricamento.

Tramite questa mappatura, è stato possibile mappare a piano il progetto rappresentato con lo standard JobPlan sulle tabelle del nostro database; il fatto stesso che si è riusciti ad utilizzare la rappresentazione del database al fine di contenere dati caratterizzati da una forma sostanzialmente differente è un indicatore della qualità e della malleabilità della rappresentazione realizzata.

Conclusioni

Durante l'intera durata del progetto delineato in questa tesi il focus primario è stato indirizzato verso la creazione di una struttura informativa aziendale in grado di ospitare dettagli relativi alla gestione dei piani di manutenzione. Tale infrastruttura è stata concepita come una solida base di supporto per l'intera gestione dei dati aziendali. Attualmente, l'azienda non adotta uno standard definito per la forma dei dati, determinando una varietà di rappresentazioni diverse. Queste rappresentazioni cambiano in base al cliente e alle specifiche esigenze richieste, generando una gestione frammentata che ha contribuito a creare uno storico di dati aziendali privo di una struttura fissa, apparendo come un "blob" di informazioni eterogenee.

Considerando che gran parte dei clienti dell'azienda fa uso di SAP PM come CMMS per la gestione dei piani di manutenzione, si è deliberato di sfruttare la rappresentazione interna dei dati in SAP per generare la struttura finale del database. Di conseguenza, è stata eseguita una dettagliata mappatura della struttura interna delle tabelle che governano il funzionamento di SAP PM. Questa mappatura ha rappresentato il fondamento per la fase di progettazione della base di dati aziendale.

Il sistema descritto in questa tesi mira a fornire un'organizzazione ben definita delle informazioni progettuali. Per garantire la completezza del sistema, si è ritenuto opportuno incorporare anche moduli esterni per consentire lo scambio efficiente di dati. In particolare, è stata inclusa un'interfaccia CRUD per l'azione mirata su record specifici e una componente di data exchange finalizzata al caricamento e alla validazione massiva di informazioni.

Basando i progetti futuri su questa solida infrastruttura, si prevedono innumerevoli vantaggi. Tra quelli di maggior rilievo vi sono i seguenti:

- *Organizzazione e strutturazione*: un'infrastruttura informativa, come quella delineata in questa tesi, assume un ruolo centrale nella promozione dell'organizzazione e della strutturazione delle informazioni aziendali. Attraverso una progettazione oculata, i dati vengono organizzati in modo efficiente, consentendo una categorizzazione, una classificazione e una correlazione avanzate. Tale strutturazione facilita notevolmente l'accesso e la comprensione dei dati, creando un ambiente informativo dove la ricerca, l'analisi e l'utilizzo delle informazioni risultano più agevoli e intuitivi.

- *Automatizzazione dei processi*: l'introduzione di un sistema di questo tipo non si limita a fornire un'organizzazione migliore dei dati, ma porta anche a un livello superiore di automatizzazione dei processi aziendali. La chiave di questa automatizzazione risiede nella riduzione della dipendenza dagli interventi umani, mitigando il rischio di errori derivanti da dimenticanze, incomprensioni o distrazioni. I processi automatizzati

seguono con precisione le regole e le istruzioni predefinite, integrando controlli di validazione che minimizzano gli errori umani. Questa automazione non solo migliora l'efficienza operativa, ma contribuisce anche a garantire una maggiore coerenza e accuratezza nei processi aziendali.

- *Integrazione dei dati*: l'infrastruttura informativa proposta facilita in modo significativo l'integrazione dei dati provenienti da diverse fonti. Questo aspetto assume un'importanza cruciale nel contesto aziendale esaminato, in cui le informazioni possono aderire a standard rappresentativi differenti. Una progettazione attenta dell'infrastruttura consente l'unificazione e l'integrazione armoniosa dei dati provenienti da varie fonti, creando un ambiente informativo dove l'uniformità è più completa e accurata. La capacità di integrare dati da fonti eterogenee consente all'azienda di avere una visione più esaustiva e affidabile delle informazioni, contribuendo alla presa di decisioni informate e tempestive.

La conclusione del progetto delineato in questa tesi è coronata da un successo evidente, con la quasi totalità degli obiettivi prefissati raggiunti con soddisfazione. L'approccio dettagliato alla mappatura di SAP PM ha permesso di ottenere uno schema di rappresentazione delle informazioni che non solo si presenta completo, ma rivela anche una notevole flessibilità. La decisione strategica di basare la progettazione sulla parte core dello schema interno a SAP ha garantito un'eccellente sovrapponibilità dello schema finale alla maggior parte dei progetti inclusi nello storico aziendale.

L'operazione di taglio e semplificazione della struttura SAP originaria, unitamente all'inserimento di tabelle di validazione nei punti critici, ha contribuito a conferire allo schema una straordinaria adattabilità agli standard interni all'azienda. Questa scelta mirata ha dimostrato di essere cruciale nel garantire che il sistema informativo possa soddisfare le diverse esigenze e gli standard specifici adottati dall'azienda nel corso dei vari progetti di manutenzione industriale. Durante le fasi di caricamento dei diversi progetti, queste proprietà si sono rivelate ampiamente, facilitando un processo fluido e privo di particolari difficoltà.

La mappatura dettagliata ha consentito di acquisire una visione approfondita delle dinamiche interne di SAP PM, mentre l'applicazione di tagli mirati e semplificazioni ha conferito al sistema risultante una straordinaria versatilità. Questo approccio ha non solo semplificato la fase di implementazione, ma ha anche garantito una migliore coerenza con l'esperienza aziendale pregressa. La sovrapponibilità tra lo schema progettuale e gli standard interni dell'azienda ha facilitato un processo di integrazione senza intoppi, riducendo al minimo le complessità e aumentando l'efficienza operativa.

La realizzazione delle componenti di interfaccia CRUD e data exchange rappresenta un ulteriore successo nel quadro complessivo del progetto. Entrambe le componenti hanno superato con successo gli obiettivi prefissati, contribuendo in modo significativo all'efficacia complessiva del sistema informativo.

Analizzando l'interfaccia CRUD emerge con chiarezza la sua struttura intuitiva e potente, che facilita l'interazione con tutte le informazioni presenti nella base di dati. Come ampiamente discusso nei capitoli dedicati di questa tesi, questa componente ha implementato un raggruppamento logico di tutti i dati pertinenti al maintenance e all'asset register. Questa strutturazione agevola le operazioni fondamentali di creazione, eliminazione, modifica e lettura per ciascuna struttura dati. Il supporto cruciale offerto dal software SQL Maestro PHP Generator ha svolto un ruolo fondamentale nello sviluppo di questa interfaccia, permettendo un'armoniosa integrazione con le intricazioni interne del database.

Per quanto riguarda la componente di data exchange, i risultati sono altrettanto positivi. L'obiettivo primario di eseguire operazioni di caricamento massivo è stato raggiunto attraverso l'esecuzione dei tre moduli dedicati all'upload dei dati di validazione, di asset e di

manutenzione. Questa operazione di caricamento massivo è stata condotta senza intoppi, confermando il successo della componente. La recente aggiunta della funzionalità di logging promette di ampliare ulteriormente l'accessibilità e la fruibilità del software. In prospettiva futura, questa componente consentirà anche agli utenti meno esperti di operare con sicurezza. Nel caso di eventuali errori, l'output formulato in un linguaggio comprensibile semplificherà notevolmente la risoluzione delle problematiche, permettendo di individuare con precisione le aree del template di caricamento che richiedono intervento per superare gli errori e portare a termine la fase di loading senza ostacoli.

L'efficacia di queste componenti non solo si attesta nel raggiungimento degli obiettivi specifici, ma si estende alla creazione di un ambiente informativo in cui l'utente, sia egli esperto o meno, può interagire in modo fluido e senza difficoltà con i dati aziendali. L'implementazione di queste soluzioni rappresenta un passo significativo verso l'efficienza operativa e la facilità d'uso del sistema informativo aziendale, promuovendo un ambiente di lavoro dinamico ed orientato all'innovazione

Attualmente, nonostante i progressi significativi compiuti nel corso del progetto, risultano ancora presenti alcune aree di miglioramento , principalmente a causa delle limitazioni temporali. Uno dei principali focus di attenzione è la componente di data exchange, la quale potrebbe trarre beneficio da ulteriori miglioramenti e controlli di validazione. Ad esempio, l'implementazione di metodi specifici per eseguire controlli mirati sui campi testuali non vincolati a tabelle di validazione potrebbe rappresentare un passo avanti. Un esempio concreto di ciò riguarda i controlli sulla lunghezza massima delle stringhe, che, se non gestiti correttamente durante la fase di caricamento, potrebbero generare errori non previsti.

Inoltre, sempre per quanto riguarda la componente di data exchange, è previsto, nel futuro immediato, lo sviluppo di un modulo software dedicato all'output dei dati progettuali, presenti nella base di dati, verso formati specifici. Un obiettivo chiave di questo sviluppo è la creazione di un modulo capace di generare dati progettuali validati e strutturati, trasferendoli direttamente in caricatori VBA per SAP. Questa funzionalità risulterebbe estremamente utile, poiché, attualmente, l'upload di dati in SAP PM è possibile solo attraverso l'utilizzo di script Visual Basic in Excel. L'introduzione di questo modulo consentirebbe un flusso diretto delle informazioni prodotte dall'azienda ai clienti attraverso caricatori Excel.

Per quanto concerne l'interfaccia CRUD, un'ulteriore area di miglioramento potrebbe concentrarsi sull'espansione delle sue funzionalità. Attualmente, la componente permette unicamente di agire in modo mirato sui record, senza la possibilità di operare in modo massivo sui dati dopo il caricamento. Come discusso nelle sezioni introduttive della tesi, ciò rappresenta una delle forze principali di Excel, che consente di manipolare facilmente grandi quantità di dati con pochi passaggi. Ad esempio, l'operazione di modifica dei dati presenti in una colonna in Excel può essere eseguita inserendo il valore desiderato e copiandolo lungo tutta la colonna con pochi click. In un database tradizionale, questa operazione richiederebbe la modifica puntuale della colonna in ogni record della tabella. Pertanto, l'espansione delle funzionalità CRUD potrebbe includere la capacità di operare su larga scala sui dati, migliorando, così, l'efficienza operativa e la flessibilità dell'interfaccia.

In prospettiva futura, emergono opportunità significative che potrebbero notevolmente arricchire l'interfaccia di interazione con il database. L'inclusione di moduli grafici, ispirati alle funzionalità presenti in Excel, rappresenterebbe un passo in avanti considerevole. Questi moduli grafici potrebbero consentire un'interazione più intuitiva e visuale con i dati, superando l'approccio tradizionale basato esclusivamente su query SQL. L'obiettivo sarebbe quello di offrire all'utente la possibilità di manipolare e analizzare i dati attraverso un'interfaccia grafica user-friendly, eliminando la necessità di competenze SQL avanzate.

L'integrazione di funzionalità grafiche avanzate nell'interfaccia consentirebbe all'azienda di eseguire operazioni di creazione dati direttamente nel database, sfruttando appieno i

vantaggi di una gestione strutturata e di processi di validazione automatizzati. Questo approccio non solo semplificherebbe il processo di inserimento dei dati, ma aprirebbe anche nuove opportunità di analisi e visualizzazione degli stessi attraverso strumenti visivi intuitivi.

Una delle prospettive di sviluppo più rilevanti nel prossimo futuro riguarda l'automazione dei processi di creazione dati. Attualmente, l'azienda sta esplorando la possibilità di utilizzare tecniche derivate dal machine learning e dalla data analysis per automatizzare l'associazione tra operazioni di manutenzione e attrezzature. Questa soluzione potrebbe basarsi sulla categorizzazione delle tipologie di operazioni di manutenzione e delle attrezzature.

Per quanto concerne gli asset, le informazioni relative all'object type e alla classe già consentono una categorizzazione preliminare delle attrezzature. Tuttavia, l'intenzione è quella di implementare approcci più avanzati, sfruttando algoritmi di machine learning che possano apprendere dai dati storici, e associare automaticamente le operazioni di manutenzione alle attrezzature pertinenti. Questo livello di automazione aumenterebbe non solo l'efficienza operativa, ma anche la precisione e l'adattabilità del sistema informativo, garantendo una gestione più intelligente e proattiva delle attività di manutenzione.

La medesima metodologia dovrebbe essere adottata anche nel futuro sviluppo di una metodologia di raggruppamento delle operazioni di manutenzione. L'obiettivo sarà quello di assegnare ogni asset e ogni operazione a una specifica categoria, facendo leva sulle potenzialità delle tecniche di apprendimento automatico. L'implementazione di componenti software avanzate potrebbe conseguentemente consentire la proposta automatica di operazioni di manutenzione da eseguire su un oggetto specifico fornito in input.

Questa prospettiva rappresenterebbe una fonte di valore significativa per l'azienda, permettendo la generazione automatica di parte dei piani di manutenzione. In molti casi, è riscontrabile che determinate tipologie di oggetti siano associate sistematicamente alle stesse operazioni di manutenzione. Una volta generato il piano di manutenzione basato sulle associazioni apprese tra tipo di oggetto e operazione, sarà, comunque, necessaria una revisione umana mirata. Questa revisione permetterà di correggere e modificare eventuali errori, garantendo un livello aggiuntivo di precisione e adattabilità alle specifiche esigenze aziendali.

In conclusione, il sistema illustrato in questa tesi non si pone come un punto di arrivo, ma costituisce, piuttosto, una solida base di partenza per l'inizio di un percorso volto al miglioramento dell'infrastruttura informativa interna all'azienda. Le soluzioni sviluppate, in sinergia con potenziali sviluppi futuri collegati, sono in grado sin da subito di innalzare significativamente le metodologie con cui i dati sono gestiti all'interno di Pansoinco. L'adozione di tecniche avanzate di apprendimento automatico e l'implementazione di sistemi di raccomandazione per le operazioni di manutenzione rappresentano passi fondamentali verso l'efficienza operativa e l'ottimizzazione continua dei processi aziendali. La revisione e l'evoluzione costante di queste soluzioni garantiranno un contributo continuo al successo e alla competitività dell'azienda nel panorama sempre mutevole dell'industria della manutenzione industriale.

Bibliografia

BENGTSSON, S. (2020), «The possibilities of improving maintenance through CMMS data analysis», .

DAILY, J. (2017), «Predictive maintenance: How big data analysis can improve maintenance», .

HUO, Z. (2005), «CMMS Based Reliability Centered Maintenance», .

KISTER, T. C. (2006), «Maintenance Planning and Scheduling», .

LIEBSTÜCKEL, K. (2020), «Configuring Plant Maintenance in SAP S/4HANA», .

POÓR, P. (2015), «CMMS as an effective solution for company maintenance costs reduction», .

RAOUF (1999), «Planning and Control of Maintenance Systems: Modelling and Analysis», .

RICKY SMITH, R. K. M. (2007), «Rules of Thumb for Maintenance and Reliability Engineers», .

STENERIK, B. (2013), «Maintenance Knowledge Management with Fusion of CMMS and CM», .

VELMURUGAN, R. S. (2021), «Asset Maintenance Management in Industry», .

Siti Web consultati

- SE80 - Documentazione SAP PM – `https://www.se80.co.uk/`
- Documentazione DBeaver – `https://dbeaver.com/docs/dbeaver/`
- Documentazione Plpgsql – `https://www.postgresql.org/docs/current/plpgsql.html`
- Documentazione Postgres – `https://www.postgresql.org/docs/`

- Sql Maestro PHP Generator – `https://www.sqlmaestro.com/products/mysql/phpgenerator/`
- Documentazione Pandas – `https://pandas.pydata.org/docs/`
- Comandi Linux – `https://docs.rockylinux.org/books/admin_guide/03-commands/`

Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutte le persone che hanno contribuito al completamento di questa tesi di laurea. Senza il loro supporto, questo traguardo non sarebbe stato possibile.

Innanzitutto, vorrei ringraziare il mio relatore, il Prof. Domenico Ursino, e i miei Correlatori, l'Ing. Antonio Spadaccini e l'Ing. Botond Zalai-Ruzsics, per la guida ed il costante sostegno. I vostri consigli preziosi e la pazienza mostrata nel corso di questo percorso sono stati fondamentali per la realizzazione di questo lavoro.

Un sentito ringraziamento va anche alla mia famiglia, ai miei genitori Giordano Bedetta e Michela Petrini, per il loro amore, il loro incoraggiamento e il sostegno incondizionato che mi hanno offerto nel corso degli anni. Senza il loro sostegno emotivo e morale, non sarei mai potuto giungere a questo punto.

Desidero, inoltre, ringraziare i miei amici e compagni di corso per le lunghe ore di studio condivise, i momenti di distensione e il supporto reciproco che ci ha aiutato a superare le sfide di questo percorso.

Infine, voglio esprimere la mia gratitudine a tutte le persone che hanno partecipato alle varie fasi legate allo sviluppo e alla progettazione del sistema esposto nella mia tesi. Il vostro contributo è stato essenziale per il mio lavoro.

Grazie a tutti coloro che hanno reso possibile la realizzazione di questa tesi. Il vostro supporto e la vostra presenza nella mia vita sono un regalo inestimabile.