IE 442 Enterprise Information Systems Modeling
SQL Project Report

Prepared by: Sude Filiz
Student ID: 2021402273

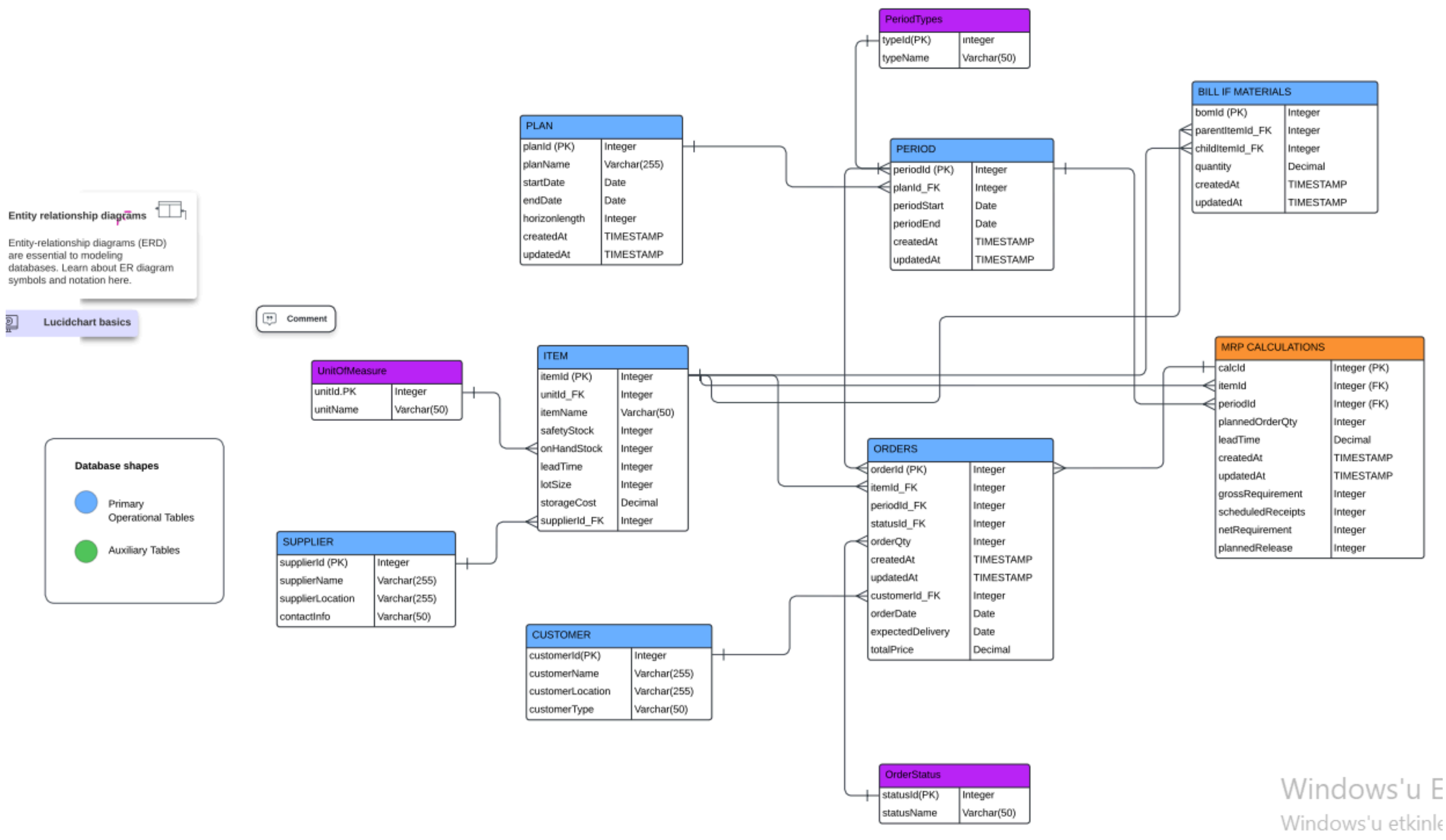
INTRODUCTION

In this project, a database system was designed and implemented. The project involved creating SQL schemas for data organization, modeling the database structure with an Entity-Relationship (ER) diagram, and integrating MRP computations into the database. This was accomplished by using SQL to carry out all required computations within the database. Additionally, a Python-based application was created to interact with the database and execute SQL queries. The project also included a test function that inserts sample data, executes MRP calculations, and validates the expected outcomes.
A Streamlit-based user interface was created in order to enhance accessibility and user experience. This UI allows users to input necessary data, execute MRP calculations, and view the results dynamically. A short instructional video showing the developed system's functionality was the final deliverable.
This project ensured data accuracy, automation, and user-friendliness in enterprise information systems by providing an effective and systematic approach to MRP.

ER diagram

An entity-relationship diagram (ERD) is a graphical representation that illustrates the relationships between different entities within a database system. (IBM, n.d.)
An entity is a matter that exists independently and can be uniquely identified. It can be a physical object (e.g., a car, a house), an event (e.g., a sale, a service), or a concept (e.g., a customer transaction). Entities belong to entity types, which categorize them. How entities relate to one another is defined by a relationship. Relationships function as verbs, connecting two or more nouns (entities). Entities and relationships can both have attributes. An Entity-Relationship Diagram (ERD) visually represents entity sets (groups of similar entities) and relationship sets (groups of similar relationships), rather than single instances. Outlining the important entities, their characteristics, and their connections within a system, helps model database structures.

Because of its collaborative features and intuitive interface, Lucidchart was utilized to create ERDs in this project.



(Figure 1: Entity-Relationship Diagram (ERD) of the MRP System)

The key entities in the diagram are:

- **PLAN**: Stores production plans, including details such as name, start and end dates, and horizon length.
- **ORDERS**: Represents customer or production orders, linked to items, periods, and statuses.
- **ITEM**: Contains information about raw materials or products, including stock levels and unit measurements.
- **BILL OF MATERIALS (BOM)**: Defines parent-child relationships between items, specifying quantities needed for production.
- **MRP CALCULATIONS**: Stores computed Material Requirements Planning data, including planned orders and lead times.
- **PERIOD**: Defines planning periods linked to a specific production plan.
- **UnitOfMeasure, OrderStatus, and PeriodTypes**: Auxiliary tables that support classification and status tracking.

Code Part

After designing the Entity-Relationship Diagram (ERD) in Lucidchart, the following step was to implement the database schema using SQLite.
The code begins by connecting to an SQLite database file called "mrp_database_final.db". A cursor object is created to execute SQL commands.
The database schema consists of multiple interrelated tables that support Material Requirements Planning (MRP). The PLAN table stores production plans, while the PERIOD table defines time periods associated with them. The PeriodTypes table is used for period classification. Inventory management is handled through the ITEM table, which links to UnitOfMeasure for measurement units and SUPPLIER for sourcing information. The CUSTOMER and ORDERS tables manage order processing, associating items, customers, order statuses (OrderStatus), and periods. The BILL_OF_MATERIALS (BOM) table establishes hierarchical relationships between items, defining component requirements. The MRP_CALCULATIONS table stores computed values essential for planning, such as gross requirements, scheduled receipts, and net requirements.

Foreign Keys enforce relationships between tables, while Primary Keys uniquely identify each record to preserve data integrity. In order to ensure organized and trustworthy data administration, key relationships include linking PERIOD to PLAN, ITEM to SUPPLIER, and ORDERS to both ITEM and CUSTOMER.

Test Function

1. Creating a sample:
After constructing the database schema, sample data was inserted from a CSV file (Updated_MRP_Data.csv) to create relevant tables. Python (Pandas and SQLite3) was used for the implementation to ensure efficient data loading while maintaining referential integrity.
A connection was established with the SQLite database (mrp_database_final.db)
This procedure maintained relational consistency while ensuring that the MRP database was populated with meaningful data.

2. MRP Calculations:
After populating the database with sample data, MRP calculations were performed using SQL queries within the database. To calculate crucial MRP parameters based on inventory levels, safety stock, and order quantities, an INSERT INTO... SELECT query was used.
As required by the project guidelines, this SQL-based method guarantees that all MRP calculations are carried out inside the database itself. This improves production planning's efficiency and data consistency.

3. Exporting MRP Calculation Results:
After executing the MRP calculations, the calculated data was extracted from the MRP_CALCULATIONS table and exported for further analysis.

Streamlit Dashboard for MRP System

A Streamlit-based dashboard was created for managing Material Requirements Planning (MRP) operations in order to improve accessibility and user interaction. The dashboard provides an intuitive interface for users to input data, execute MRP calculations, and review results dynamically.
The Streamlit-based MRP System Dashboard provides a simple and interactive interface for managing material requirements planning. Users are welcomed and given instructions on how to utilize the system in the Home section. By entering information like the item name, safety stock, on-hand stock, lead time, lot size, storage cost, and supplier ID, users can add new inventory items in the Add Data area. This data is stored in the ITEM table while maintaining referential integrity with the SUPPLIER table. The Run MRP Calculation section initiates the MRP process by deleting earlier findings and figuring out planned order quantities, gross requirements, net requirements, and scheduled releases based on inventory levels and client orders. QL queries are used to carry out these calculations, which are then saved in the MRP_CALCULATIONS table. Finally, the View Results section displays the computed MRP data in a tabular format, allows users to examine planning outcomes, and download the results as a CSV file for additional analysis.

Conclusion

In this project, an MRP system was designed and implemented using SQL and Python. Initially, an ER diagram was created to model the database structure. Then, the database schema was developed using SQLite, defining tables and relationships. Data samples were imported from a CSV file, and MRP computations were carried out in the database via SQL queries. Finally, a Streamlit-based dashboard was developed to allow users to input data, run MRP calculations, and view results dynamically. The integration of a relational database schema, SQL-based MRP computations, and a Streamlit user interface ensured automation, data consistency, and usability. The system provides an efficient approach to inventory and production planning, offering a scalable framework for enterprise applications.

The integration of a relational database schema, SQL-based MRP computations, and a Streamlit user interface guaranteed automation, data consistency, and usability. The system offers an efficient approach to inventory and production planning, providing a scalable framework for enterprise applications.

References

IBM. (n.d.). *Entity-relationship diagram*. IBM Think. Retrieved from https://www.ibm.com/think/topics/entity-relationship-diagram

Wikipedia contributors. (n.d.). *Entity–relationship model*. Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model