Basics of database systems

Project – Database design

Lappeenranta-Lahti University of Technology LUT
Software Engineering

Basics of database systems
Spring 2022

Santeri Hynninen 565519

TABLE OF CONTENTS

1 DEFINITION

Small manufacturing company needs ERP database, which stores sales and information about inventory, products, and stakeholders. Company doesn't have a huge number of SKUs but most of them are manufactured in house and made from not sale ready components. So there need to be way to track which products need which materials and how much. This will help with manufacturing and procurement processes.

Every level of employee will have to use the database for different purposes. Production employees must find what and in what quantity they need to make wanted products, Salespersons must manage customer, sales, and inventory information. Also, there must be way find suppliers information to make orders. Administration might want to see how salespersons are doing and how much they are selling. Also, they might want to see how long the employee have been in the company. All except manufacturer workers will have access to edit and update data.

Because in this case our resources are really limited, we have chosen 7 different actions which has to be implemented.

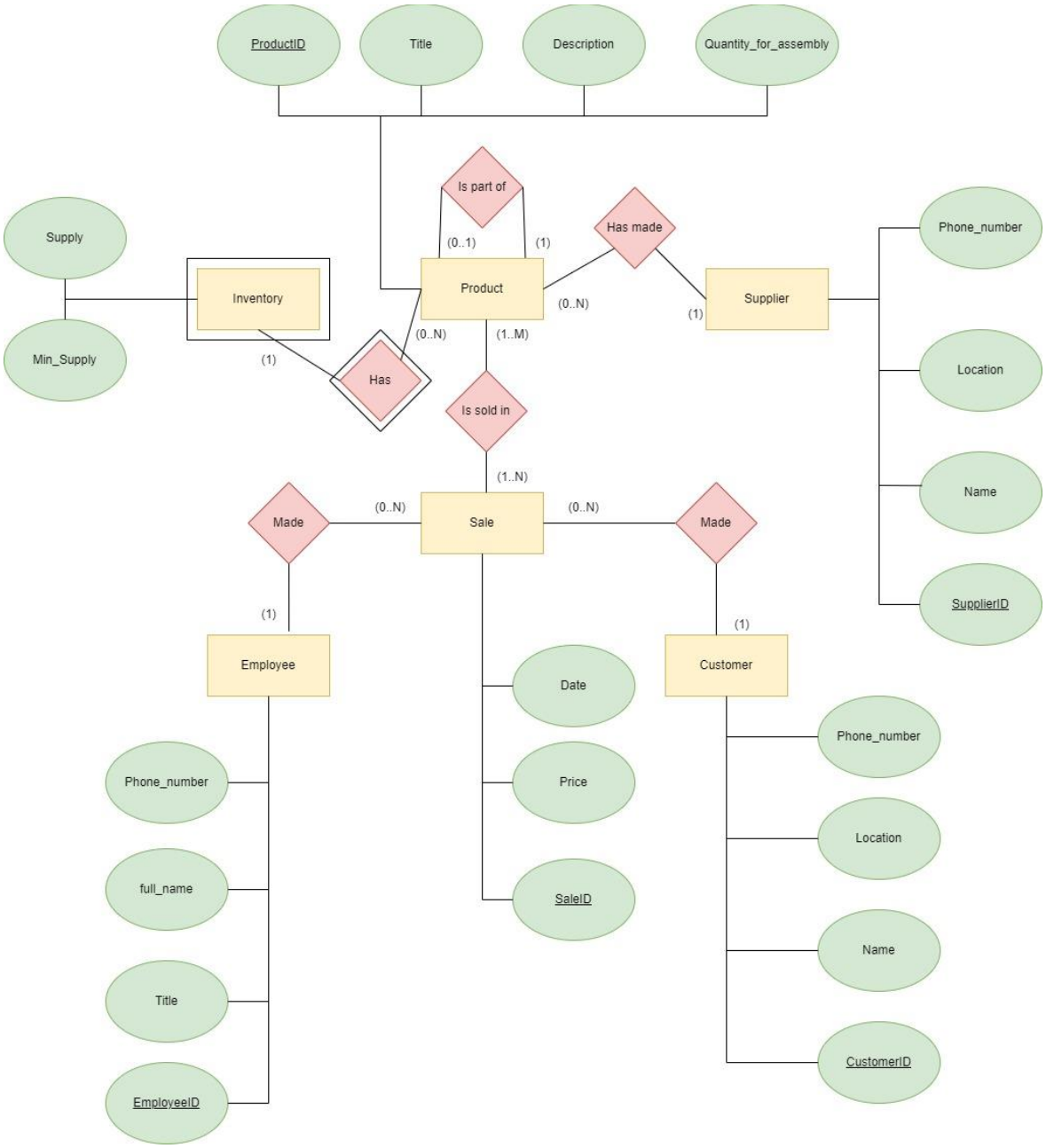
Queries to implement:

- Find every component of a product
- Find supplier for every product
- Make customer profiles about how much they have bought in dollars and in what quantities
- Input sale information (and update inventory, when product is sold)
- Update inventory separately
- Show inventory status
- List all sales and all items sold in that sale

2 MODELING

2.1 Concept model

Database project started by identifying what different entities database could have, and how they would interact with each other. Also, entities might have something data what they want to save so they might need attributes. For mapping these we used entity relation table, where everything is mapped out.



In ER-model we can see all entities as yellow rectangles, attributes as a green ellipses and relationships as a red rectangle. Connected to either Sale or Product entities. Employee will

make Sale and he will tell the software which products are sold and to whom. Therefore, Employee will make a sale. One employee can make multiple sales, but sale can have only one salesperson who made it. Same goes to the customer who was also making the sale, only one customer can be signed to one sale. But customer might do more purchases in the future so there is possibility to have multiple sales under one customer.

Products in other hand are made by one Supplier who probably does other products as well. That is why there can be only one supplier against multiple possible products. Company did not want to have half-finished products as separate entity if someone wants to buy them. That is why Product can be part of the product, for example in example data special steels are part of the drill bits which company manufactures. In this case because company has only few materials and many products every product has the part product saved in it. So, one end product can have one either one or zero parts. Inventory is weak entity because it does not have id for SKUs, it only references to the product id which will be unique. Again, there is only one inventory and inside it there is multiple products. Last relationship is between products and sales. One sale might have multiple different products sold in a package and product will be in multiple sales. That is why we get m:n relationship between them. Each entity has a unique value, expect Inventory which is weak entity, with the unique value you can uniquely identify the entity and refer to it if needed.

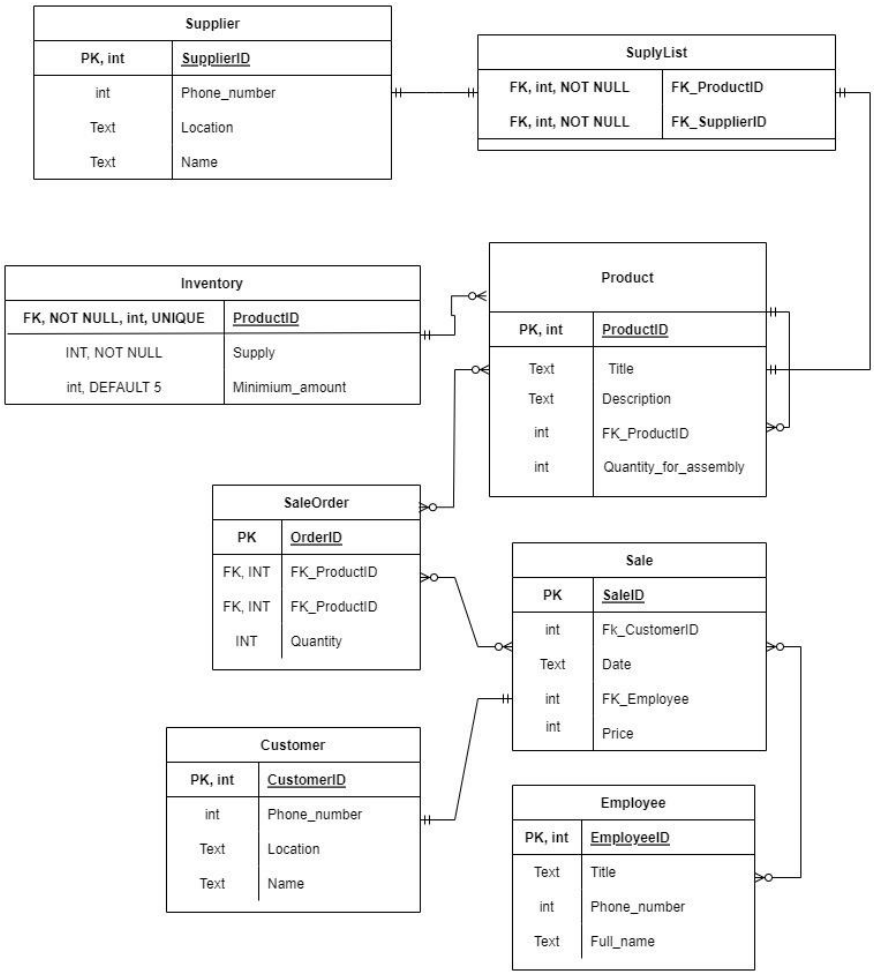
2.2 Relational model

Starting the transforming, first step is to transform every strong entity to relation. And after those, weak entities are also transformed but they will need a unique key. This is resolved by giving inventory the productid of all products as a foreign key but further specified that key is unique. This way it will have unique key to follow. All strong entities are converted without problem.

Second step is transforming attributes. In the ER there were no derived, multivalued or composite attributes so transforming them can be skipped. Things like full name can be seen as a composite but there is no need to use names separately so making them both independent value is redundant.

Third step relationships are transformed. Every one to one or one to many relationships needs least one foreign key, in case of one to many it will be placed on the side of “many” After that relationship between sales and products were transformed. Because it is many to

many relationships both need a foreign key. This is done by making table called ordering list. There is every sale event and every product sold in that event. These actions can be identified by orderid which is primary key of the table and tells one product to sale relation. Also, between supplier and products same kind of table is made. This is not mandatory because it is one to one relationship, but it might help in the future.



3 DATABASE IMPLEMENTATION

In general

In process I had to think some attributes again and think if they are really needed. I first thought that things like cost and weight would be beneficial to product but later in implementation I found them unnecessary to the project and updated the models. Also, I regretted to implement SupplyList table, I had thought that it did have purpose in the project but found it was completely unnecessary. But removing it later would take time. Also, in middle of implementing I completely changed how products work and their relationship. First, I had idea to have one master product to be sold but found out it would be more practical to have few raw materials or parts which would be used in many finished products. Also, I first thought it would be sufficient to have one product type in one sale

action. But in practice it would make only extra duplicates in the system and inefficient to use. That why I designed later Saleorder table which would group all products under one saleid. This was technically little bit more challenging but surely better choice. After I had implemented everything, I also updated all models to correspond the actual design.

Implementation

For using the database python program was developed. Program has simple interface with menu. All queries can be used by giving numbers to the software. For creating tables I crated one .sql file which can be separately or via python. I also used the bokeh library to visualize the customer habits in sales. If queries need multiple values they are asked separately, if user must give id number to program, program prints list where user can read which id matches to wanted entity.

Note during testing:

In testing phase I found that error handling could need more work. Software has some exceptions cases against software problems like query not working or empty, but there is a lot of space for human error. For example, if you would give wrong ID to the program, it would probably crash. This could be dealt with but would need more limited resources so it will be neglected now. For the same reason fixing the inventory status has been neglected. If you update or sell mor than there is in the inventory you will have negative error. This is not catastrophic but in dream case we would implement trigger for it to notify staff to either make an order or cancel the sale action.

4 DISCUSSION

4.1 Indices

I did not utilize any indices in the project. This is largely since the material was not available when I made the project. In fact, I had programmed and reported everything before I heard about the new topic. Also, I did not see a need for them when data amount was not that big and in real world there would not be real problem considering database was made for small company. But if the data amount would increase, I see that Indices would be one of the easiest ways to increase capacity of the system. Most useful they would be with customer or sale tables because those have most data and are regularly searched. For customers are not going to grow rapidly, but you might to want to send some customers advertisement

messages. With indices you can make the query a lot faster. But again, it is small scale database and indices are nor essential.

4.2 Reflection and conclusion

I found the project interesting but little bit too large and bit undefined. Because rules were so open it was hard to find what is enough and what is not. But I think I have found answer to most of the requirements and used enough time to this project. I think it was one of the most useful software projects to this point and I think I can utilize some of the knowledge gathered in the future.