Basics of database systems

**Project – Database design**

Lappeenranta-Lahti University of Technology LUT

Software Engineering

Basics of database systems

Spring 2022

Student name: Nghia Nguyen

Student number: 000275466

Table of contents

[Table of contents 1](#_Toc96083488)

[1 Definition 2](#_Toc96083489)

[2 modeling 3](#_Toc96083490)

[2.1 Concept model 3](#_Toc96083491)

[2.2 Relational model 6](#_Toc96083492)

[3 database implementation 8](#_Toc96083493)

[3.1 SQL code explanation 8](#_Toc96083494)

[3.1.1 Trigger statement with ‘nig’ keyword 8](#_Toc96083495)

[3.1.2 Trigger statement with ‘Turku’ keyword 9](#_Toc96083496)

[3.1.3 ‘orders\_done\_by\_worker’ VIEW 9](#_Toc96083497)

[3.1.4 ‘top\_trending\_products\_Kuopio’ VIEW 10](#_Toc96083498)

[3.1.5 Purchase order overview in Kuopio 11](#_Toc96083499)

[3.1.6 Update actual\_delivery\_date information in the database 12](#_Toc96083500)

[3.2 Python code 12](#_Toc96083501)

[4 discussion 13](#_Toc96083502)

[Figure 1. ER model 4](#_Toc96083503)

[Figure 2. Entity (Location) and its attributes 5](#_Toc96083504)

[Figure 3. Entity(Store) and its attributes 5](#_Toc96083505)

[Figure 4. Entity (Purchase\_Order) and its attributes 5](#_Toc96083506)

[Figure 5. Entity (Customer) and its attributes 5](#_Toc96083507)

[Figure 6. Entity (Products) and its attributes 5](#_Toc96083508)

[Figure 7. Entity (Worker) and its attributes 6](#_Toc96083509)

[Figure 8. Entity (Buy\_order) and its attributes 6](#_Toc96083510)

[Figure 9. Types of cardinalities of the relationships 6](#_Toc96083511)

[Figure 10. Relational model 7](#_Toc96083512)

[Figure 11. Linking/interim relation 8](#_Toc96083513)

[Figure 12. First query 9](#_Toc96083514)

[Figure 13. Second query 10](#_Toc96083515)

[Figure 14. ‘orders\_done\_by\_worker’ view 10](#_Toc96083516)

[Figure 15. Result of the Figure 14 11](#_Toc96083517)

[Figure 16. ‘top\_trending\_products\_Kuopio’ VIEW 11](#_Toc96083518)

[Figure 17. Result of the Figure 16 12](#_Toc96083519)

[Figure 18. Purchase order overview in Kuopio 12](#_Toc96083520)

[Figure 19. Result of Figure 18 12](#_Toc96083521)

[Figure 20. Update query 13](#_Toc96083522)

[Figure 21. Result of update query 13](#_Toc96083523)

# Definition

The idea behind the project is to show users logistics manager the big picture of supply chain business. Precisely, this project is a relative projection of logistics outbound department in Kuopio. Since this project is just a projection, the database is as small and lightweight as possible to meet the requirements.

The database allows outbound logistics manager to keep track of all information flow from the point when customers made online orders to the point the final packages were packed sent to customer via the third party which is Posti company. In other words, the moment customers paid for their orders from internet, the purchase orders will be made immediately and transferred directly to the store from which they bought products. When store receives purchase order, logistics department will assign workers belonging to that store to pick for those orders. Note that, one picker can pick one or many products depending on how big orders can be. When pickers finished picking, buy-order paper will be printed and transferred to Posti and they make sure these orders will be delivered to customers within seven days. Beside keeping track of material flow, user or logistics manager can even alter the database so that it fits for purpose in the future. Note that, the database is designed for data analysis with built-in views and visualization created with Python through the application.

There are totally 6 queries inside the database:

1. ‘compare\_estimate\_delivery\_date\_with\_actual\_deliver\_date’ VIEW
2. ‘orders\_done\_by\_worker’ VIEW
3. ‘top\_trending\_products\_Kuopio’ VIEW
4. Purchase order overview in Kuopio
5. Update actual\_delivery\_date information in the database
6. ‘location\_not\_allowed’ TRIGGER

# modeling

## Concept model

Diagram

Description automatically generated

Figure 1. ER model

The database design begins with business operation analysis. The target of the analysis is the whole activity domain of outbound logistics department and a subpart of the domain supply chain department. The model is represented graphically and filled with data descriptions. To be more precise, there are totally seven concepts, over thirty attributes and ten relationships between concepts. Moreover, the database has many types of cardinalities of the relationships including one-to-one 1:1, one-to-many 1:N, many-to-many M:N.

Detail information relating to entities and attributes can be found in from Figure 2 to Figure 8below.

Table

Description automatically generated

Figure 2. Entity (Location) and its attributes

Table

Description automatically generated

Figure 3. Entity(Store) and its attributes

Text

Description automatically generated

Figure 4. Entity (Purchase\_Order) and its attributes

Table

Description automatically generated

Figure 5. Entity (Customer) and its attributes

Table

Description automatically generated

Figure 6. Entity (Products) and its attributes

Table

Description automatically generated

Figure 7. Entity (Worker) and its attributes

Text

Description automatically generated with medium confidence

Figure 8. Entity (Buy\_order) and its attributes

Graphical user interface, application

Description automatically generated

Figure 9. Types of cardinalities of the relationships

All types of cardinalities of the relationships including one-to-one 1:1, one-to-many 1:N, many-to-many M:N between entities with explanation in detail can be found in Figure 9 above.

## Relational model

Diagram

Description automatically generated

Figure 10. Relational model

The relational model was transmitted from ER model can be found in Figure 10 above. Due to the fact that, the database has many relationships many-to-many M:N, therefore, the relational model will be more complicated than the ER model. Let’s break it into small pieces to check whether this relational model satisfy all rules of transforming entities or not.

Rule 1: Each strong entity is made into a relation. Key attributes are used to create the primary key. => CHECKED

Rule 2: Each weak entity is made into a relation similarly to Rule 1. The primary key is formed using the key attributes of the weak and strong entity. => CHECKED

Rule 3: Each derived attribute is discarded. => CHECKED

Rule 4: Each composite attribute is converted into simple attributes. Composite attribute is discarded. => CHECKED

Rule 5: Each multivalued attribute is transformed into a connecting relation. The primary key for the connecting relation will be the multivalued attribute as well as the key attribute the parent relation. => CHECKED

Rule 6: In each one-to-one relationships, one of the relations will have a foreign key. => CHECKED

Rule 7: In each one-to-many relationships, the foreign key is place on the side of ‘many’. => CHECKED

Rule 8: In each many-to-many relationships, a linking/interim relation is formed that stores foreign keys to both relations as well as attributes related to the relationship. => CHECKED

In particular, the linking relations between many-to-many relationships can be found in Figure 11 below.

Table

Description automatically generated

Figure 11. Linking/interim relation

Rule 9: Each n-degree relationships are transformed using Rule 8. => CHECKED

# database implementation

During the transformation of concept model to relational model, it was found out that if one many-to-many relationship has its own table, the relation model will be a mess. Therefore, I decided to merge possible linking relations based on the ER model so that the logistics manager can have a clear view of the big picture of this business. As a result, “WorkersDoJob” table was created for “Buy\_order” table, “Worker” table and “Products” table.

## SQL code explanation

### ‘compare\_estimate\_delivery\_date\_with\_actual\_delivery\_date’ VIEW

Text

Description automatically generated

Figure 12. ‘compare\_estimate\_delivery\_date\_with\_actual\_deliver\_date’ VIEW

The idea behind this query is to provide user with a virtual table based on specific need. In this case, the query will give the user the overview of how long each worker picks for one order comparing with estimate\_delivery\_date. For example, in accordance with Figure 13, the worker, whose ID is 44006346, did his job quite well because he finished the orders quite soon. As a result, customers receive their orders sooner than they expect. Note that, in the case, we assume that all order has same size same weight no matter different products.

Graphical user interface, table

Description automatically generated with medium confidence

Figure 13.Result of Figure 12

### ‘orders\_done\_by\_worker’ VIEW

Text

Description automatically generated

Figure 14. ‘orders\_done\_by\_worker’ view

The idea behind this query is to provide user with a virtual table based on specific need. In this case, this query help user identify which orders at which store have been picked by workers and detail information of those workers including ID, first\_name and last name. The virtual table can be found in Figure 15 below.

Table

Description automatically generated

Figure 15. Result of the Figure 14

### ‘top\_trending\_products\_Kuopio’ VIEW

A screenshot of a computer

Description automatically generated with medium confidence

Figure 16. ‘top\_trending\_products\_Kuopio’ VIEW

The idea behind this query is to provide user with a virtual table based on specific need. In this case, this query help user identify how many and what kind of products were bought from customers in Kuopio store. In addition, the virtual table also automatically sorts the quantity in decreasing so that user can know which product is the most popular one. As a result, the user can estimate the time to replenish it so that the case of out of stock will never happen. The virtual table can be found in Figure 17 below.

Graphical user interface, text, application, email

Description automatically generated

Figure 17. Result of the Figure 16

### Purchase order overview in Kuopio

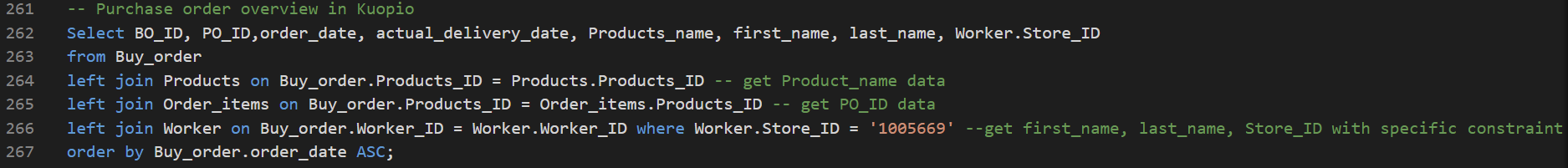


Figure 18. Purchase order overview in Kuopio

The idea behind this query is to provide user with the detail information relating which purchase orders have been transferred to Kuopio store, products inside those orders and which employers picked them.

Graphical user interface

Description automatically generated with medium confidence

Figure 19. Result of Figure 18

### Update actual\_delivery\_date information in the database

Text

Description automatically generated

Figure 20. Update query

The idea behind this query is to update information from the database. Precisely, in this case, user wants to update the actual\_delivery\_date with constraints which are BO\_ID (BO00965) and Worker\_ID (44006231).

Table

Description automatically generated with medium confidence

Figure 21. Result of update query

According to Figure 19, the actual\_delivery\_date of BO\_ID (BO00965) on the left side is from original data and the actual\_delivery\_date of BO\_ID (BO00965) on the right side is the data after implementing the update query.

### Trigger statement with ‘Turku’ keyword

A screenshot of a computer

Description automatically generated with medium confidence

Figure 22. ‘location\_not\_allowed’ TRIGGER

The idea behind this query is that the trigger will be fired automatically whenever an event such as insert occurs not relating to the cities mentioned inside the query in column city at Location table because we are focusing on analyzing data in these cities.

## Python code

I wrote a simple Python program that showed the example queries, asked the user to update data, wrote one new row to the database and demonstrated reading by showing the changes.

# discussion