Basics of database systems

**Project – Database design**

Lappeenranta-Lahti University of Technology LUT

Software Engineering

Basics of database systems

Spring 2023

Table of contents

[Table of contents 1](#_Toc92697309)

[1 Definition 2](#_Toc92697310)

[2 modeling 4](#_Toc92697311)

[2.1 Concept model 4](#_Toc92697312)

[2.2 Relational model 5](#_Toc92697313)

[3 database implementation 6](#_Toc92697314)

[4 discussion 7](#_Toc92697315)

# Definition

Describe the need of your work, for what the database is developed. Describe the problem area in such detail that it can be modelled into a database and it can be used to critically compare the concept analysis during evaluation. Finally, describe the five queries/views you are going to create.

https://cloud.smartdraw.com/editor.aspx?credID=-45799653&depoId=43082779&flags=128

**Order database**

**The order database is a database developed to track and handle orders which can have one or multiple products in them. The shipping and supplying is outsourced to external companies to reduce the costs and needed manpower. The database is heavily focused around different IDs (shipperID, orderID etc.) that ensure a reliable and efficient way of handling and tracking orders. The required information related to each table is kept to a minimum, to increase efficiency and make the database clearer.**

**The database should be able to quickly display if a product is out of stock, so suppliers and the company can be noted about it. Also it should be possible to view the amount of non-filled orders (orders where the isFilled is set to false) to make sure that the system doesn’t get overwhelmed by orders.**

The following database queries are to be implemented:

1. List current orders where isFilled = false (not yet given to shipper)
2. List the InStock amount of each product
3. Display the products related to a specific category
4. Display the price of a specified product
5. Display the products that are in a specified order

# modeling

## Concept model

Database design begins with concept analysis. The use of concept analysis leads to database design decisions that are independent of the data and implementation.

At this point of the project, the aim is to describe the conceptual model of the database that has been developed. Use the ER-modeling notation.

Represent at least: Entities (concepts), relationships (the connection between concepts) and the cardinalities of the relationships (one-to-one 1:1, one-to-many 1:N, many-to-many N:M), and properties (attributes). Describe the ER model so that you point out the most important parts that may be altered or go through modifications during the transformation to a releational model.

Figure 1 shows the ER model of the database. In total there are 7 entities, each with varying amounts of attributes. In total there are 5 relationships. There are two M:N relationships between products and categories (a product can have multiple categories and one category can have multiple products) and suppliers and products (a single product can have multiple suppliers, and a single supplier can supply multiple different products).

The customer entity is separate from everything else but is linked with the **CustomerID** primary key. The Address data is located in Order details instead of customers, since the address of the order and the customer aren’t necessarily the same. Customer can also have multiple orders, making the OrderID in CustOrders a (possibly) multivalued attribute. Customers also have a composite attribute CustName, which consists of CustFirst and CustLast.

The model is subject to change when transforming to a relational model (most notably the implementation of customers-table), but the relationships and overall structure should remain the same. There will be added linking tables between suppliers – products and products – categories, due to them having a many-to-many relationship.

[https://lucid.app/lucidchart/c9763dc2-1e83-4a92-bfcd-a2a63dfc5b62/edit?page=0\_0#](https://lucid.app/lucidchart/c9763dc2-1e83-4a92-bfcd-a2a63dfc5b62/edit?page=0_0)

Diagram

Description automatically generated

**Figure 1:** ER model

## Relational model

After the ER-modelling, transform it to a relational form using the appropriate transformation rules. Represent the result either as a relational diagram or a freeform UML-style table structure.

Each entity should have the necessary primary and foreign keys. The definition of accepted value range should be done for each attribute: length, type.

Relationships have their own integrity constraints when necessary. If a relationship has attributes, it is described as an entity.

**Example text:**

Figure 2 shows the relational model that has been created based on the ER model. Due to the N:M relationship, an interim relation was created between Staff and Task entities. The derived attribute ‘age’ was removed and the multivalued attribute 'Phone’ was separated into an additional entity. Finally, the ‘work’ relationship was created as a relation because of the related attribute.

https://app.diagrams.net/#G15jEz4GipIw5M9bH-mtUOqy9EC-l3gwWJ

Diagram

Description automatically generated

**Figure 2:** Relational modelfrom the ER model

# Database implementation

During the implementation, you have to develop the different integrity constraints as well as indices. Describe the constraints and indices you have created for your database. You can decide what kind of format you use for describing them. The example shows each relation in a list and the constraints in them, you can use the same format or use tables or whatever seems best for you.

If you have created a Python interface, describe that here as well.

**Example text**

During implementation, the following constraints are created for the relations:

* **Shipper**
  + All data must exist (NOT NULL)
* **Order**
  + OrderID cannot be null (NOT NULL)
* **Order details**
  + Foreign key (OrderID) reference to order
  + Foreign key (CustomerID) reference to customer
  + All data must exist (NOT NULL)
* **Customer**
  + Email and customerID cannot be null (NOT NULL)
  + Other fields not mandatory
* **Product**
  + Foreign key (SupplierID) references to Supplier
  + Foreign key (CategoryID) references to Category
  + All data must exist (NOT NULL)
* **Supplier**
  + Foreign key (ProductID) references to Product
  + All data must exist (NOT NULL)
* **ProductSupply**
  + Foreign key (ProductID, SupplierID) references to Product and Supplier
* **ProductCategory**
* **Category**
* **Staff**:
  + Name, date of birth and address cannot be null (NOT NULL)
  + Date of birth has to be at least 18 years ago (CHECK)
* **Wife**:
  + Foreign key reference to staff.
  + Name cannot be null (NOT NULL)
  + ON DELETE CASCADE
* **Child**:
  + Foreign key reference to staff
  + unique composite key of staffID and child name so that the same staff member doesn’t have children with the same name.
* **Perform**:
  + Foreign key reference to staff and task
  + ON UPDATE CASCADE
* **Task**:
  + Name cannot be null (NOT NULL)
* **Work**:
  + Foreign key reference to staff and company.
  + Date cannot be null and defaults to current date (NOT NULL, DEFAULT)
  + ON UPDATE CASCADE
* **Company**:
  + Name cannot be null (NOT NULL)
* **PhoneNumber**:
  + Foreign key reference to staff
  + unique phone number so that there aren’t any others with the same phone number. (UNIQUE)
  + ON UPDATE CASCADE
  + ON DELETE RESTRICT

In addition to the integrity constraints listed above, the database will also implement two indices; One based on the Task name, another based on the Work since. These indices are to allow quickly search who perform the same tasks as well as to search for employees that have started working during a specific time period.

# discussion

If you want to mention something that has not been discussed in the previous chapters, you can discuss them here.

**Example**

Nothing to discuss.