ITC-6000

Assignment 2

July 20, 2020

**[Sunil Raj Thota]**

# Question 1

**List at least four advantages of indexing. Mention some indexing tips that would be of particular help to a database designer. Also, discuss indexing practices that can adversely affect the performance of a database. In your discussion, include information on how a designer should decide on how many indexes to provide per table.**

**Advantages:**

* Helps to retrieve data from the DBMS system easily and provides uniqueness in searching for any query
* Indexing is used just to speed up to find relevant information
* Whenever the user adds new data in a row, it needs to verify that the same value is not entered. It helps to reduce the verification time or displays repeated value in the table if occurred any
* Useful for sorting the data. This will help us to group the records and aggregate values

**Indexing Tips:**

* Database designer must first check whether their database is large or small
* If the database is small then indexing will be used less. And if the database is large then indexing becomes very useful at this point
* Since the database becomes less ordered and many rows are added and removed, due to this the number of records increases
* So surely at this point indexing helps the database designer to speed up the database reads

**Affects:**

* Don’t over index transactional tables with heavy I/ O activity it may affect the functioning of the database system
* Creating duplicate indexes make the DBMS system complex
* It requires additional space to store indexes in a storage disk
* So, these all could affect the overall performance as the indexes will consume a lot of space on the storage disk

**Designer Decisions:**

* Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed
* An index is a data structure which is used to quickly locate and access the data in a database table
* Unlimited Indexes can be created per table. But a Total number of columns per table that can be indexed is 32 for B-Tree index and 30 for Bit Map index
* We can create many indexes for a table as long as the combination of columns differs for each index

# Question 2

**List at least four basic rules for identifying primary keys in a relational database. Mention how you would go about identifying the foreign keys in your design. Briefly discuss foreign key ownership and how to indicate a foreign key in your relational database model.**

**Basic Rules:**

* The primary key field cannot be null
* The value must be unique for each instance of an entity
* The values must not change or become null during the life of each entity instance
* Two rows cannot have the same primary key value
* Every row should have a primary key value
* If any foreign key is referred to as the primary key, then the value in the primary key column can never be modified or updated

**Foreign Key Identification:**

* Foreign keys are useful for ensuring data integrity. If you define them, the database itself will make sure that an invalid foreign key value will not be allowed to be inserted into the table
* A foreign key is an attribute that completes a relationship by identifying the parent entity. Foreign keys provide a method for maintaining integrity in the data (called referential integrity) and for navigating between different instances of an entity. Every relationship in the model must be supported by a foreign key
* Foreign keys are formed in dependent and subtype entities by migrating the entire primary key from the parent or generic entity. If the primary key is composite, it may not be split
* Both the tables must have the same structure and data type, the values present in each row of the referencing table should match with the value of the corresponding primary key columns of a row in a referenced table

**Foreign Key Ownership:**

* Foreign key attributes are not considered to be owned by the entities to which they migrate, because they are reflections of attributes in the parent entities
* Thus, each attribute in an entity is either owned by that entity or belongs to a foreign key in that entity
* If the primary key of a child entity contains all the attributes in a foreign key, the child entity is said to be "identifier dependent" on the parent entity, and the relationship is called an "identifying relationship”
* If any attributes in a foreign key do not belong to the child's primary key, the child is not identifier dependent on the parent, and the relationship is called "non identifying"
* It must be a non-null value and unique for an entity. The primary key must also be the candidate key. The value must not be changed or altered over the database for lifetime

**Relational Database Example:**

# 

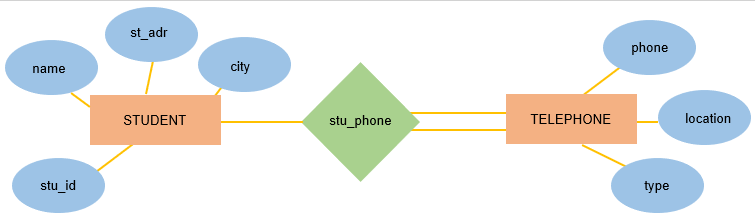
# Question 3

**Describe at least two examples of common errors in entity relationship modeling. If possible, provide a graphical illustration of the problems and solutions. List some questions the designer should consider before designing the models so that these errors can be avoided.**

**Common Errors:**

* Designers often forget to indicate the cardinality i.e. 1:1, 1:M: M:1, M: N
* Sometimes designers forget to mention the primary key and foreign key if the diagram contains various attributes
* Using improper field names
* This creates unnecessary confusion for peers with whom designers work

Now let us consider this E-R Diagram, where a student uses the telephone in his university.



As specified above, here we can see that the designer has forgotten to mention the type of relationship between the entities.

So, the database designer needs to define the cardinality i.e. 1:1, 1:M: M:1, M: N. Secondly, using confusing names. Here we can find that the name “stu\_phone” is a bit confusing. However, the database designer could have simply defined the relationship as “uses”. However, using “stu\_phone” is not wrong but that creates unnecessary confusion for others.

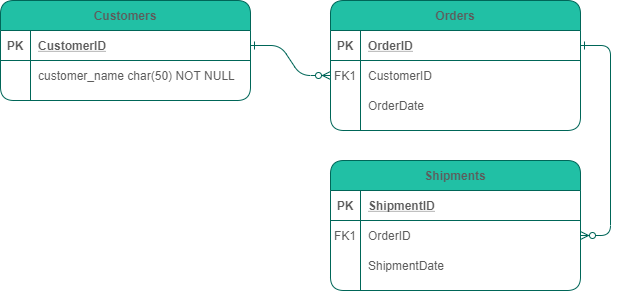
**Few Questions:**

* These errors can easily be avoided if the database designers consider certain questions before designing the database models:
* Defining the purpose of designing the database model?
* How to convert the information items into columns?
* Check whether appropriate names are used in fields?
* Whether the cardinality is defined in the diagram or not?
* Whether the primary key is underlined or not?
* How to divide the information into tables?
* Do I need to apply the normalization rules in database models?

# Question 4

**Using library resources or the Internet, locate an entity relationship diagram (ERD) utility that can be used to create graphic database designs. Provide a general description about the company that produces the tool. Mention the ER modeling techniques offered by the tool and the database products that it supports. Discuss other features that might make this an attractive product to a database designer and explain why.**

**Draw.io** is online software that allows making flowcharts, process diagrams, and different charts like flowcharts, org charts, UML, ER, and network diagrams.

****

**About the company:**

This software is provided by JGraph Ltd, a UK-based company. **Draw.io** is founded by Mr. Gaudenz Alder in 2005 with a mission statement “provide free, high quality diagramming software for everyone”. They work in collaboration with various cloud services like Google Drive, one drive, Jire drive, Confluence cloud. Their influenced customers include Apple, Canon, IBM, Phillips, CISCO, HP, Siemens, Visa, T-Systems.

**Features:**

* Easy to use and it is Drag and Drop
* Availability of huge shapes library
* Easy installation and training material provision
* Users can import and export various formats. This allows users to publish and share their work at ease
* Secure and reliable and works everywhere with any device

**Some E-R Modelling techniques used in Draw.io:**

* **Connecting shapes to show relationships:** We can then select the connector and change the start and end connector style using the drop-down lists in the Style tab in the Format panel
* **Expanding or collapsing the entity:** Some entities have many attributes. To simplify the diagram, we can expand or collapse the entity
* **Using sliding connectors:** When drawing connectors for relationships, we can drag the shapes around to make room, the connectors will slide or float into the right positions
* **Exporting the text:** We can create a text list of entities and their attributes and keys by using the text plugin provided in draw.io software and export it to a text file
* **Using template shapes:** We can use the scratchpad for storing groups of shapes we don’t have to build each up from scratch. If we are creating the second style of ER diagram, we can store a template with as many extra rows for the attributes as we need

**Other featured needs to be considered:**

* **Aggregation**: It cannot express relationships among relationships. So, aggregation has to be used by which relationship is treated as higher-level entities
* **Generalization:** It is a relationship that exists between a higher-level entity set and one or more lower-level entity sets. Using this feature, users will be able to define a higher-level entity and lower-level entity
* **Attribute inheritance**: As we have aggregation for high-level entities, there is attribute inheritance for low-level entities. Using this feature, the user will be able to indicate a low-level entity in their diagram