EJERCICIO N° 6

Lea e interprete el siguiente texto y en clase se realizara un cuestionario

Database design

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database management system manages the data accordingly.

Database design involves classifying data and identifying interrelationships. This theoretical representation of the data is called an ontology. The ontology is the theory behind the database's design.

Determining data to be stored

In a majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore, the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system.

This process is one which is generally considered part of requirements analysis, and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge. This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.

Determining data relationships

Once a database designer is aware of the data which is to be stored within the database, they must then determine where dependency is within the data. Sometimes when data is changed you can be changing other data that is not visible. For example, in a list of names and addresses, assuming a situation where multiple people can have the same address, but one person cannot have more than one address, the address is dependent upon the name. When provided a name and the list the address can be uniquely determined; however, the inverse does not hold - when given an address and the list, a name cannot be uniquely determined because multiple people can reside at an address. Because an address is determined by a name, an address is considered dependent on a name.

(NOTE: A common misconception is that the relational model is so called because of the stating of relationships between data elements therein. This is not true. The relational model is so named because it is based upon the mathematical structures known as relations.)

Logically structuring data

Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to arrange the data into a logical structure which can then be mapped into the storage objects supported by the database management system. In the case of relational databases the storage objects are tables which store data in rows and columns. In an Object database the storage objects correspond directly to the objects used by the Object-oriented programming language used to write the applications that will manage and access the data. The relationships may be defined as attributes of the object classes involved or as methods that operate on the object classes.

The way this mapping is generally performed is such that each set of related data which depends upon a single object, whether real or abstract, is placed in a table. Relationships between these dependent objects is then stored as links between the various objects.

Each table may represent an implementation of either a logical object or a relationship joining one or more instances of one or more logical objects. Relationships between tables may then be stored as links connecting child tables with parents. Since complex logical relationships are themselves tables they will probably have links to more than one parent.

Database designs also include ER (entity-relationship model) diagrams. An ER diagram is a diagram that helps to design databases in an efficient way.

Attributes in ER diagrams are usually modeled as an oval with the name of the attribute, linked to the entity or relationship that contains the attribute.

ER models are commonly used in information system design; for example, they are used to describe information requirements and / or the types of information to be stored in the database during the conceptual structure design phase.

A design process suggestion for Microsoft Access

1. **Determine the purpose of the database** - This helps prepare for the remaining steps.
2. **Find and organize the information required** - Gather all of the types of information to record in the database, such as product name and order number.
3. **Divide the information into tables** - Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
4. **Turn information items into columns** - Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.
5. **Specify primary keys** - Choose each table's primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be Product ID or Order ID.
6. **Set up the table relationships** - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.
7. **Refine the design** - Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.
8. **Apply the normalization rules** - Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables, as needed.

Normalization

In the field of relational database design, *normalization* is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics—insertion, update, and deletion anomalies that could lead to loss of data integrity.

A standard piece of database design guidance is that the designer should create a fully normalized design; selective denormalization can subsequently be performed, but only for performance reasons. The trade-off is storage space vs performance. The more normalized the design is, the less data redundancy there is (and therefore, it takes up less space to store), however, common data retrieval patterns may now need complex joins, merges, and sorts to occur - which takes up more data read, and compute cycles. Some modeling disciplines, such as the dimensional modeling approach to data warehouse design, explicitly recommend non-normalized designs, i.e. designs that in large part do not adhere to 3NF. Normalization consists of normal forms that are 1NF,2NF,3NF,BOYCE-CODD NF (3.5NF),4NF and 5NF

Document databases take a different approach. A document that is stored in such a database, typically would contain more than one normalized data unit and often the relationships between the units as well. If all the data units and the relationships in question are often retrieved together, then this approach optimizes the number of retrieves. It also simplifies how data gets replicated, because now there is a clearly identifiable unit of data whose consistency is self-contained. Another consideration is that reading and writing a single document in such databases will require a single transaction - which can be an important consideration in a Microservices architecture. In such situations, often, portions of the document are retrieved from other services via an API and stored locally for efficiency reasons. If the data units were to be split out across the services, then a read (or write) to support a service consumer might require more than one service calls, and this could result in management of multiple transactions, which may not be preferred.

Physical design

The physical design of the database specifies the physical configuration of the database on the storage media. This includes detailed specification of data elements, data types, indexing options and other parameters residing in the DBMS data dictionary. It is the detailed design of a system that includes modules & the database's hardware & software specifications of the system. Some aspects that are addressed at the physical layer:

- Security - end-user, as well as administrative security.
- Replication - what pieces of data get copied over into another database, and how often. Are there multiple-masters, or a single one?
- High-availability - whether the configuration is active-passive, or active-active, the topology, coordination scheme, reliability targets, etc all have to be defined.
- Partitioning - if the database is distributed, then for a single entity, how is the data distributed amongst all the partitions of the database, and how is partition failure taken into account.
- Backup and restore schemes.

At the application level, other aspects of the physical design can include the need to define stored procedures, or materialized query views, OLAP cubes, etc.