Mater Dei College
Tubigon, Bohol
2nd Semester, A.Y. 2022-2023
CC 105 – IM 101 (INFORMATION MANAGEMENT 1)

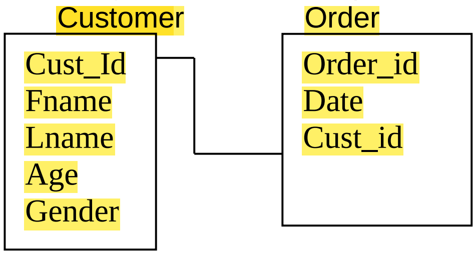
Topic 3
Database Models, SQL & Database Design Phase 1- Analysis

Database Models

There are several models of databases depending on the format in which the data is managed. The typical models are summarized below.

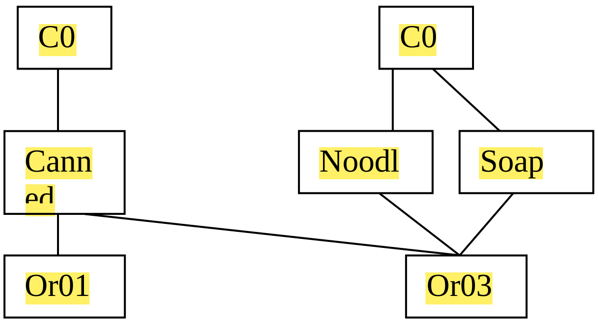
- ***Relational database***

  Data is managed in tables. Multiple tables are linked by item values to build database.



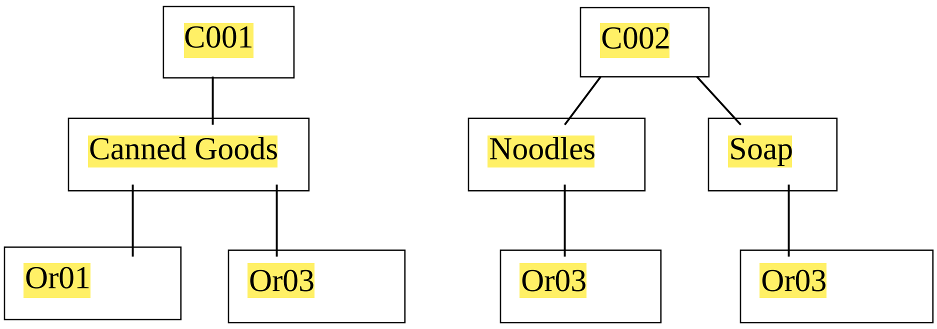
- **Network database**

  Data is managed in a meshed format. Many-to-many parent-child relationships.



- **Hierarchical database**

  Data is managed in a hierarchical structure. One-to-many parent-child relationship comprise the data



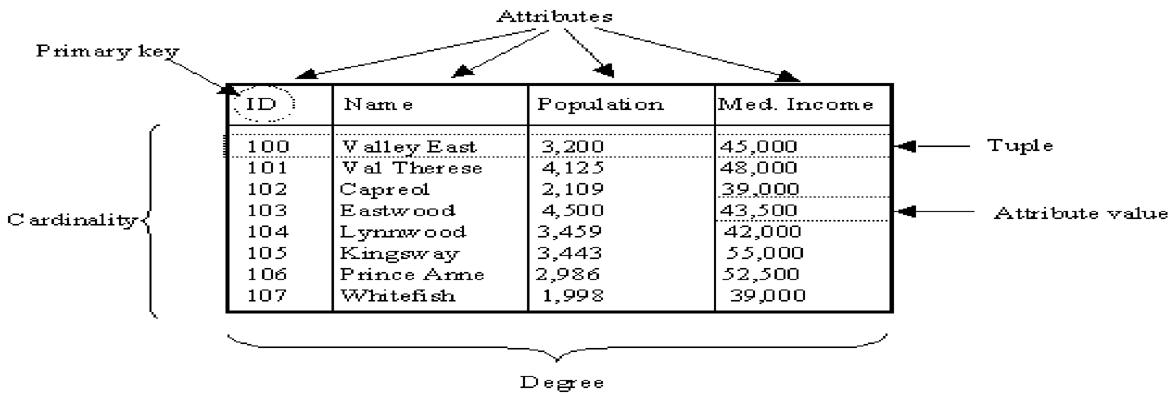
Characteristics of Relational databases

- Related data is managed in table
- Easy for user to understand
- Tables are linked by saved values
- Easy to extract the user's desired data

Table- it is a collection of data about a specific topic, such as product or supplier. Using a separate table each topic means that you store that data only once. This results in a more efficient database and fewer to data-entry error. It is a logical structures maintained by the database manager. Table are made of column/field/attribute and row/record/tuple. You can define table as part of your data definition.

Informal Terms	Formal Terms in RDBMS
Table	Relation
Column	Attribute/Domain
Row	Tuple
Values in a column	Domain
Table definition	Schema of relation
Populated table	Extension

Figure 13: Characteristics of a relational table



ID	Name	Population	Med. Income
100	Valley East	3,200	45,000
101	Val Therese	4,125	48,000
102	Capreol	2,109	39,000
103	Eastwood	4,500	43,500
104	Lynnwood	3,459	42,000
105	Kingsway	3,443	55,000
106	Prince Anne	2,986	52,500
107	Whitefish	1,998	39,000

A.K. Yeung 1998-10-10u51-13

Relation is a table with columns and rows.
Attribute is a named column of a relation
Domain is the set of allowable values for one or more attributes
Tuple is a row of a relation.
Degree of a relation is the number of attributes it contains
Cardinality of a relation is the number of tuples it contains.

Structured Query Language (SQL) – is the standard language designed to access relational databases. This language is based on the groundbreaking work of DR. E.F. Codd, with the first implementation of SQL being developed by IBM in the mid-1970s. IBM was conducting a research project known as System R, and SQL was born from that project. Later in 1979 a company then known as Relational Software Inc. (known today as Oracle Corporation) released the first commercial version of SQL.

SQL is now fully standardized and recognized by the American National Institute (ANSI). You can use SQL to access an Oracle, SQL Server, DB2, or MySQL DBMes. It uses a simple syntax that is easy to learn and use. There are five types of SQL statements:

1. Query Statement – allows you to retrieve rows stored in database tables. You can write query using the SQL SELECT statement.
2. Data Definition Language (DDL) statement – allows you to define the data structure, such as tables that make up a database. There are five basic types of DDL statements:
   a. CREATE – allow you to create a database structure. CREATE database DB_name is used to create a database, Create Table tb_name(……..) is used to create a table, and Create User is used to create a database user.
   b. ALTER – allow you to modify a database structure. ALTER table is used to modify a table.
   c. DROP – allow you to remove a database structure. Drop database is used to delete a database; drop table is used to delete a table structure and its data.
   d. RENAME – allows you to change the name of the table.
   e. TRUNCATE – allows you to delete the entire contents of the table.

2. Data Manipulating Language (DML) Statements – allow you to modify the contents of the tables. There are three DML statements:
   - INSERT – allows you to add tuple to a table
   - UPDATE – allows you to change a tuple
   - DELETE – allows you to remove tuples.

4. Transaction Control (TC) Statements – allow you to permanently record the changes made to the rows stored in a table or undo those changes. There are three TC statements:
   - COMMIT – allows you to permanently record the changes made to the rows
   - ROLLBACK – allows you to undo changes made to rows
   - SAVEPOINT – allows you to set a “savepoint” to which you can rollback changes made to rows

5. Data Control Language (DCL) Statements – allow you to change the permission on database structures. There are two types of DCL statements:
   - GRANT – allows you to give other users to access to your database structures.
   - REVOKE – allows you to prevent another user from accessing to your database structures.

Database Design: Overview

Databases exist because of the need to change data into information. Data are the raw and unprocessed facts. Information is obtained by processing the data into something useful. For example, millions of names and telephone numbers in a phone book are data. Information is the telephone number of the fire department when your house is burning down.

A database is a large repository of facts, designed in such a way that processing the facts into information is easy. If the phone book was structured in a less convenient way, such as with names and numbers placed in chronological order according to when the numbers were issued, converting the data into information would be much more difficult. Not knowing when the fire department was issued their latest number, you could search for days, and by the time you find the number your house would be a charred pile of ash. So, it's a good thing the phone book was designed as it was.

A database is much more flexible; a similar set of data to what's in a phone book could be ordered by MariaDB according to name, telephone number, address as well as chronologically. But databases are of course more complex, containing many different kinds of information. People, job titles and a company's products can all mingle to provide complex information. But this complexity makes the design of databases more complex as well. Poor design could make for slow queries, or it could even make certain kinds of information impossible to reach. This section of the Knowledge Base features articles about good database design, specifically:

- The database lifecycle
- Entity-relationship modeling
- Common mistakes in database design
- Real-world example: creating a publishing tracking system
- Concurrency control with transactions

Database Lifecycle

Like everything else, databases have a finite lifespan. They are born in a flush of optimism and make their way through life achieving fame, fortune, and peaceful anonymity, or notoriety as the case may be, before fading out once more. Even the most successful database at some time is replaced by another, more flexible and up-to-date structure, and so begins life anew. Although exact definitions differ, there are generally six stages of the database lifecycle.

Analysis

The analysis phase is where the stakeholders are interviewed and any existing system is examined to identify problems, possibilities and constraints. The objectives and scope of the new system are determined.

Design

The design phase is where a conceptual design is created from the previously determined requirements, and a logical and physical design are created that will ready the database for implementation.

Implementation

The implementation phase is where the database management system (DBMS) is installed, the databases are created, and the data are loaded or imported.

Testing

The testing phase is where the database is tested and fine-tuned, usually in conjunction with the associated applications.

Operation

The operation phase is where the database is working normally, producing information for its users.

Maintenance

The maintenance phase is where changes are made to the database in response to new requirements or changed operating conditions (such as heavier load).

Database development is not independent of systems development, often being one component of the greater systems development process. The stages of systems development basically mirror the stages of a database lifecycle but are a superset. Whereas database design deals with designing the system to store the data, systems design is also concerned with the processes that will impact on the data.

Database Design Phase 1: Analysis

Your existing system can no longer cope. It's time to move on. Perhaps the existing paper system is generating too many errors, or the old Perl script based on flat files can no longer handle the load. Or perhaps an existing news database is struggling under its own popularity and needs an upgrade. This is the stage where the existing system is reviewed.

Depending on the size of the project, the designer may be an individual, responsible for the database implementation and coding, or may be a whole team of analysts. For now, the term *designer* will represent all these possibilities.

The following are the steps in the Analysis Phase.

1. Analyze the organization
2. Define any problems, possibilities or constraints
3. Define the objectives
4. Agree on the scope

When reviewing a system, the designer needs to look at the bigger picture - not just the hardware or existing table structures, but the whole situation of the organization calling for the redesign. For example, a large bank with centralized management would have a different structure and a different way of operating from a decentralized media organization, where anyone can post news onto a website. This may seem trivial, but understanding the organization you're building the database for is vital. to designing a good database for it. The same demands in the bank and media organizations should lead to different designs because the organizations are different. In other words, a solution that was constructed for the bank cannot be unthinkingly implemented for the media organization, even when the situation seems similar. A culture of central control at the bank may mean that news posted on the bank website has to be moderated and authorized by central management, or may require the designer to keep detailed audit trails of who modified what and when. On the flip-side, the media organization may be more laissez-faire and will be happy with news being modified by any authorized editor.

Understanding an organization's culture helps the designers ask the right questions. The bank may not ask for an audit trail, it may simply expect it; and when the time comes to roll out the implementation, the audit trail would need to be patched on, requiring more time and resources.

Once you understand the organization structure, you can question the users of any existing system as to what their problems and needs are, as well as what constraints will exist then. You need to question different role players, as each can add new understanding as to what the database may need. For example, the media organization's marketing department may need detailed statistics about the times of day certain articles are read. You may also be alerted to possible future requirements. Perhaps the editorial department is planning to expand the website, which will give them the staff to cross-link web articles. Keeping this future requirement in mind could make it easier to add the cross-linking feature when the time comes.

Constraints can include hardware ("We have to use our existing database server") or people ("We only have one data capturer on shift at any one time"). Constraints also refer to the limitations on values. For example, a student's grade in a university database may not be able to go beyond 100 percent, or the three categories of seats in a theatre database are small, medium and large.

It is rarely sufficient to rely on one level of management, or an individual, to supply objectives and current problems, except in the smallest of organizations. Top management may be paying for the database design, but lower levels will need to use it, and their input is probably even more important for a successful design.

Of course, although anything is possible given infinite time and money, this is (usually) never forthcoming. Determining scope, and formalizing it, is an important part of the project. If the budget is for one month's work but the ideal solution requires three, the designer must make clear these constraints and agree with the project owners on which facets are not going to be implemented.

Database Design Example Phase 1: Analysis

Real-world example: creating a publishing tracking system

Now let's walk through the database design process with a step-by-step example. The Poet's Circle is a publisher that publishes poetry and poetry anthologies. It is keen to develop a new system that tracks poets, poems, anthologies and sales. The following sections show the steps taken from the initial analysis to the final, working database.

Poet's circle database phase 1: analysis

The following information is gleaned from speaking to the various stakeholders at Poet's Circle. They want to develop a database system to track the poets they have recorded, the poems they write, the publications they appear in, as well as the sales to customers that these publications make.

The designer asks various questions to get more detailed information, such as "What is a poet, as far as the system goes? Does Poet's Circle keep track of poets even if they haven't written or published poems? Are publications recorded even before there are any associated poems? Does a publication consist of one poem, or many? Are potential customer's details recorded?" The following summarizes the responses in our example:

- Poet's Circle is a publisher that bases its choices of publications on an active poetry community on its website. If enough of the community wants a poem published, Poet's Circle will do so.
- A poet can be anybody who wants to be a poet, not necessarily someone who has a poem captured in the system or someone who has even written a poem.
- Poems can be submitted through a web interface, by email or on paper.
- All captured poems are written by an associated poet, whose details are already in the system. There can be no poems submitted and stored without a full set of details of the poet.
- A publication can be a single poem, a poetry anthology, or a work of literary criticism.
- Customers can sign up through a web interface and may order publications at that point in time, or express interest in receiving updates for possible later purchases.
- Sales of publications are made to customers whose details are stored in the system. There are no anonymous sales.
- A single sale can be for one publication, but many publications can also be purchased at the same time. If more than one customer is involved in this sale, Poet's Circle treats it as more than one sale. Each customer has their own sale.
- Not all publications make sales — some may be special editions, and others simply never sell any copies.

SOURCE: https://mariadb.com/kb/en/database-design-overview/
https://mariadb.com/kb/en/database-lifecycle/
https://mariadb.com/kb/en/database-design-phase-1-analysis/