TVET CERTIFICATE IV in SOFTWARE DEVELOPMENT

BASICS AND FUNDAMENTALS OF DATABASE

SFDSFD401

Develop a basic database

Competence

Credits: 9

Learning hours: 90

Sector: ICT

Sub-sector: Software Development

Module Note Issue date: June, 2020

Purpose statement

This module describes the skills, knowledge and attitudes to be acquired by the learner to Analyze database requirements design and build simple database.

At the end of this module the learner will be able to describe basics and fundamentals of database and building a simple database.

Table of Contents

Total Number of Pages: 77

LO 1.1 – Define database key terms, models, types and relationships

- Topic 1 : Database definition

A. Data

Data, in the context of databases, refers to all the single items that are stored in a database, either individually or as a set. Data in a database is primarily stored in database tables, which are organized into columns that dictate the data types stored therein.

B. Database

A database is simply a collection of related data that is organized. This may come in the form of a set of index cards (e.g. a recipe collection), an ordered list (e.g. a phone book), or a set of tables in the computer (e.g. student records at a certain school).

The database is a collection of tables, relationships and queries which enable to store data with less redundancy. Those data should be used into programs by different users.

C. Entities

An entity is an object that exists. It doesn't have to do anything; it just has to exist. In database administration, an **entity** can be a single thing, person, place, or object. Data can be stored about such entities. A design tool that allows database administrators to view the relationships between several entities is called the **entity relationship diagram (ERD)**.

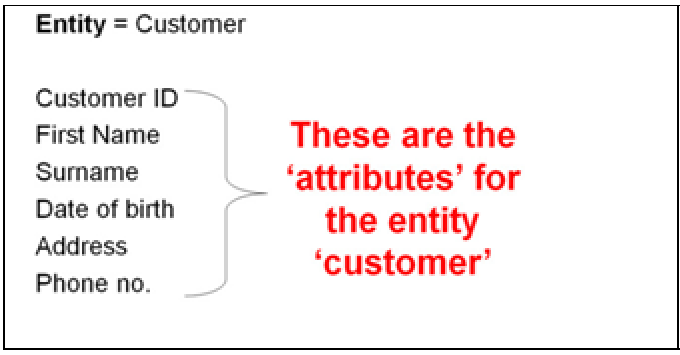
In database administration, only those things about which data will be captured or stored is considered an entity. If you aren't going to capture data about something, there's no point in creating an entity in a database.

If you're creating a database of your employees, examples of entities you may have include employees and health plan enrollment.

D. Attributes

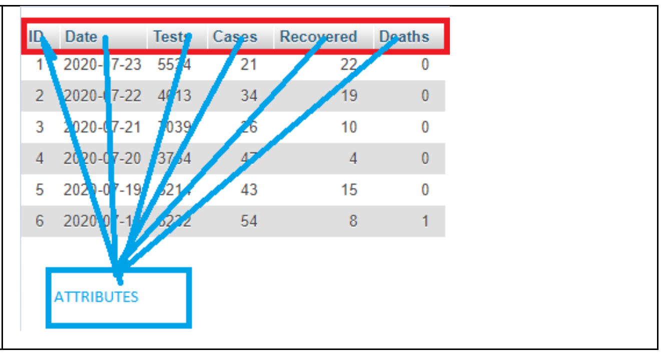
An **attribute** defines the information about the entity that needs to be stored. If the entity is an employee, attributes could include name, employee ID, health plan enrollment, and work location. An entity will have zero or more attributes, and each of those attributes apply only to that entity. For example, the employee ID of 123456 belongs to that employee entity alone.

Attributes also have further refinements, such as domain and key. The **domain** of an entity describes the possible values of attributes. In the entity, each attribute will have only one value, which could be blank or it could be a number, text, a date, or a time.


Entity = Customer

Customer ID
First Name
Surname
Date of birth
Address
Phone no.

These are the 'attributes' for the entity 'customer'


ID	Date	Tests	Cases	Recovered	Deaths
1	2020-07-23	5534	21	22	0
2	2020-07-22	4613	34	19	0
3	2020-07-21	7039	26	10	0
4	2020-07-20	3764	47	4	0
5	2020-07-19	3214	43	15	0
6	2020-07-18	3232	54	8	1

E. Records

Data is stored in records. A record is composed of fields and contains all the data about one particular person, company, or item in a database. In this database, a record contains the data for one day of testing COVID_19 in Rwanda.

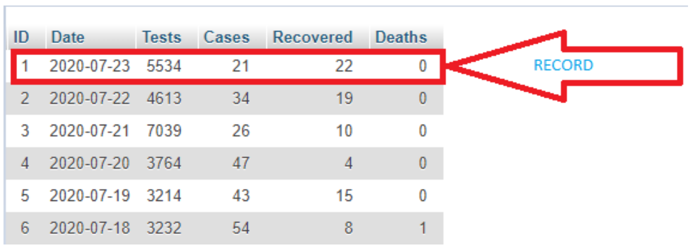
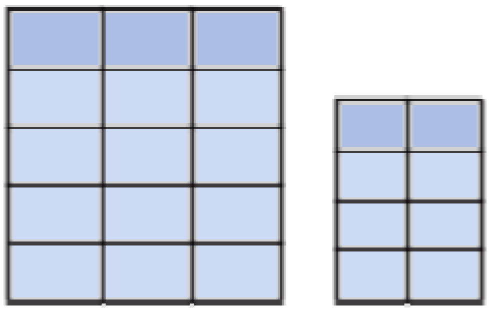

ID	Date	Tests	Cases	Recovered	Deaths
1	2020-07-23	5534	21	22	0
2	2020-07-22	4613	34	19	0
3	2020-07-21	7039	26	10	0
4	2020-07-20	3764	47	4	0
5	2020-07-19	3214	43	15	0
6	2020-07-18	3232	54	8	1

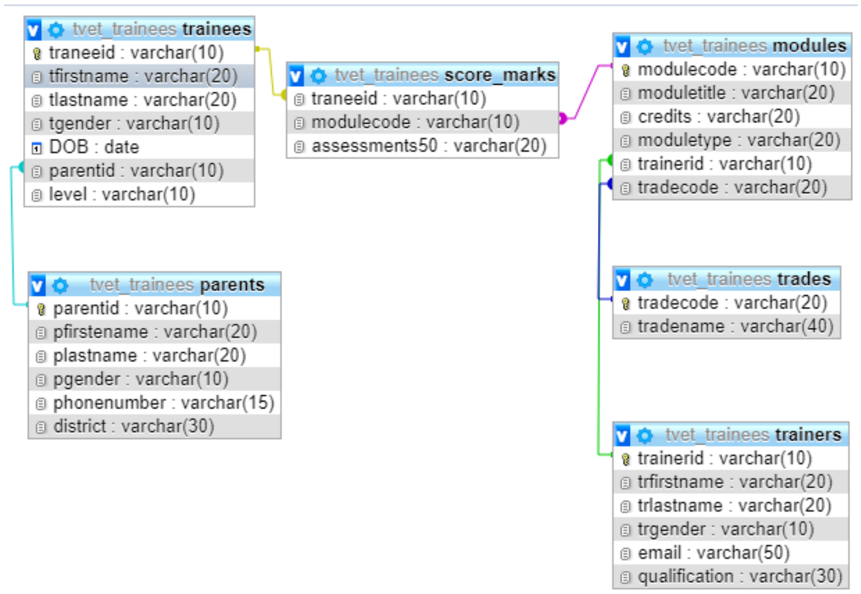
Figure: Records appear as rows in a database table.

- *Topic 2: Types of database models*

A. Relational database

Relational Model: This is the model that we will use and the most used one. Data are stored into tables of two dimensions. Manipulations are made using mathematical theory of relationship. ***DBMS or this model are usually called RDBMS, R for Relational***




tvet_trainees trainees
traneeid : varchar(10)
tfirstname : varchar(20)
tlastname : varchar(20)
tgender : varchar(10)
DOB : date
parentid : varchar(10)
level : varchar(10)

tvet_trainees score_marks
traneeid : varchar(10)
modulecode : varchar(10)
assessments50 : varchar(20)

tvet_trainees modules
modulecode : varchar(10)
moduletitle : varchar(20)
credits : varchar(20)
moduletype : varchar(20)
trainerid : varchar(10)
tradecode : varchar(20)

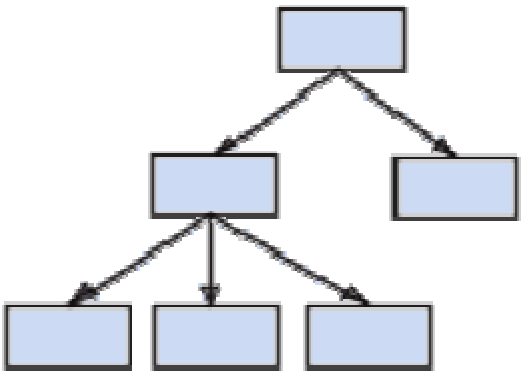
tvet_trainees parents
parentid : varchar(10)
pfirstename : varchar(20)
plastname : varchar(20)
pgender : varchar(10)
phonenumber : varchar(15)
district : varchar(30)

tvet_trainees trades
tradecode : varchar(20)
tradename : varchar(40)

tvet_trainees trainers
trainerid : varchar(10)
trfirstname : varchar(20)
trlastname : varchar(20)
trgender : varchar(10)
email : varchar(50)
qualification : varchar(30)

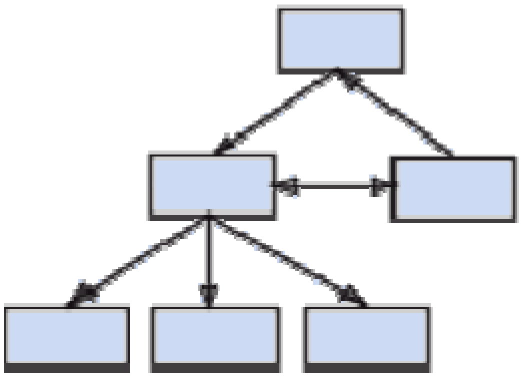
B. Hierarchical database

Data are arranged hierarchically in a descending arborescence. This model uses pointers between different records. It was the first model of DBMS.



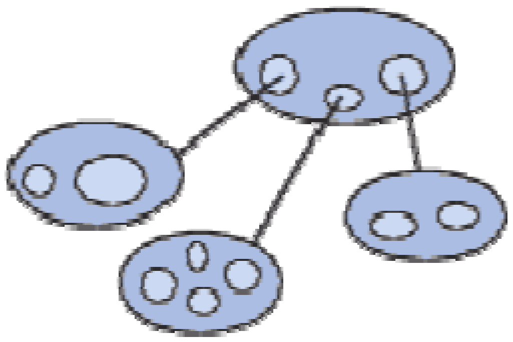
C. Network database

Network Model: It also used pointers between different records as the hierarchical one, but the structure of the arborescence is not only a descending one.



D. Object Oriented model

Object Model: This model is the recent one. Data are stored into classes. A class has fields and methods. The DBMS of this model are often called ODBMS, O for Object. *But it is important to note that, even if ODBMS are the recent ones, they have not yet shown their proof. From the last two passed decades and up to now, RDBMS remain the most popular and used ones. Some authors are even saying* ***Relational Object DBMS****:* ***It is remaining difficult to forget the old but still powerful relational model***



- Topic 3:Database relationships

More details LO.2.1: B

Using foreign keys, or other candidate keys, you can implement three types of relationships between tables:

A: One-to-one: This type of relationship allows only one record on each side of the relationship. The primary key relates to only one record – or none – in another table. For example, in a marriage, each spouse has only one other spouse. This kind of relationship can be implemented in a single table and therefore does not use a foreign key.

B: One-to-many/many-to-one: A one-to-many relationship allows a single record in one table to be related to multiple records in another table. Consider a business with a database that has Customers and Orders tables.

A single customer can purchase multiple orders, but a single order could not be linked to multiple customers. Therefore, the Orders table would contain a foreign key that matched the primary key of the Customers table, while the Customers table would have no foreign key pointing to the Orders table.

C: Many-to-many: This is a complex relationship in which many records in a table can link to many records in another table. For example, our business probably needs not only Customers and Orders tables, but likely also needs a Products table.

Again, the relationship between the Customers and Orders table is one-to-many, but consider the relationship between the Orders and Products table. An order can contain multiple products, and a product could be linked to multiple orders: several customers might submit an order that contains some of the same products. This kind of relationship requires a minimum three tables.

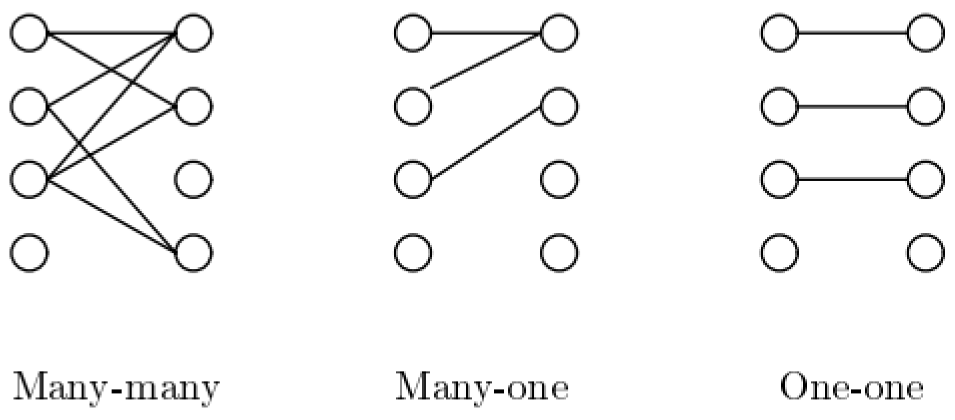


Figure: Type of relationships

LO 1.2 – Review organizational and task requirements to identify user requirement

- Topic 1:Techniques and Methods to collect data

A. Interview

It is probably the most used and efficient one. The designer holds a conversation with the employees of the organization who can provide the right information. He asks questions to the employees who answer them.He has to prepare these questions before he meets the employees. Those questions should be clear, objective and simple. The questions are asked one by one and he notes progressively the responses.

B. Documentation

Data can be found in documents. The documentation represents the biggest source of information in a given organization. The designer has to request all the documents that are related to the processes to computerize. Hard and soft documents as well as web pages should be used.
Here we mean: Existing manual reports, books and e-books, Articles and journals, websites etc.The designer has to present the reports of the software almost like the manual documents used before the computerization. This will help the users to quickly tame the new system.

C. Questionnaire

This technique is almost like interview, but the only difference is that the designer doesn't have to meet employees. He prepares his questionnaire before and submits it to the employees who fill it. He collects it later after being filled. The importance of this technique is that you can get many data in a short time.

D. Observation

Some data can just be collected by making carefully an observation on the processes to comprise. The designer has to visit therefore the places where processes are made.

- Topic 2:Types of requirements

Functional and non-functional requirements

Statements of services the system should provide how the system should react to particular inputs and how the system should behave in particular situations.

A. Functional requirements

Describe functionality or system services. They depend on the type of software, expected users and the type of system where the software is used Functional user requirements may be high-level statements of what the system should do BUT functional system requirements should describe the system services in detail.

Example

The user shall be able to search either all of the initial set of databases or select a subset from it. The system shall provide appropriate viewers for the user to read documents in the document store. Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

B. Non-functional requirements

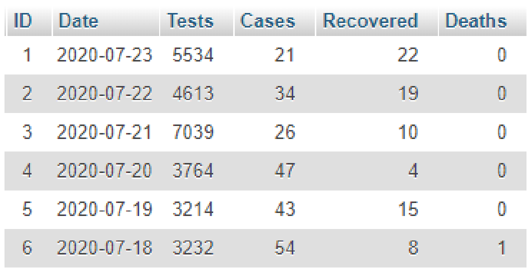
Define system properties and constraints. e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
Process requirements may also be specified mandating a particular CASE system, programming language or development method
Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

LO 1.3 – Determine the information that the database is required to hold

- Topic 1:Determination of database elements

A. Tables

A database table is composed of records and fields that hold data. Tables are also called datasheets. Each table in a database holds data about a different, but related, subject.



ID	Date	Tests	Cases	Recovered	Deaths
1	2020-07-23	5534	21	22	0
2	2020-07-22	4613	34	19	0
3	2020-07-21	7039	26	10	0
4	2020-07-20	3764	47	4	0
5	2020-07-19	3214	43	15	0
6	2020-07-18	3232	54	8	1

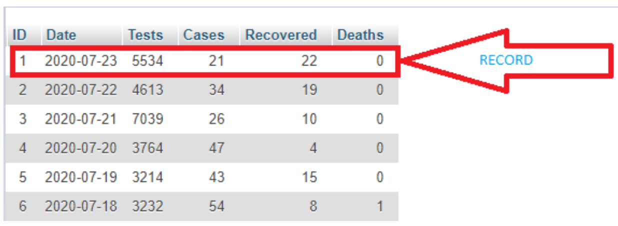
Table: COVID_RECORDS

B. Properties

Properties (= attributes and relationships)

C. Records

Data is stored in records. A record is composed of fields and contains all the data about one particular person, company, or item in a database. In this database, a record contains the data for one customer support incident report. Records appear as rows in the database table. A record for Log ID 1201242 is highlighted in Figure.



ID	Date	Tests	Cases	Recovered	Deaths
1	2020-07-23	5534	21	22	0
2	2020-07-22	4613	34	19	0
3	2020-07-21	7039	26	10	0
4	2020-07-20	3764	47	4	0
5	2020-07-19	3214	43	15	0
6	2020-07-18	3232	54	8	1

D. Fields

A field is part of a record and contains a single piece of data for the subject of the record. A field or article or attribute is a propriety which characterizes an object.

OBJECT
Field 1
..
..
Field n

STUDENTS
StudentID
Surname
FirstName
Sex

Note: When you are naming your fields you should:

- ✓ Give short and simple names
- ✓ Give explicit names

- ✓ Avoidspaces and accents
- ✓ Avoid dash (-), if yes use underscore (_)

  - Topic 2:Determination of data types

A. Character

Character String Data Types

This section of the article will talk about the character data types. These data types allow characters of fixed and variable length. Refer to the below table.

Data Type	Description / Maximum Size		Storage
	Description	**Maximum Size**	
text	Allows a variable length character string	2GB of text data	4 bytes + number of chars
varchar(max)	Allows a variable length character string	2E + 31 characters	2 bytes + number of chars
varchar	Allows a variable length character string	8,000 characters	2 bytes + number of chars
char	Allows a fixed length character string	8,000 characters	Defined width

Unicode Character Strings Data Types

Data Type	Description / Maximum Size		Storage
	Description	**Maximum Size**	
ntext	Allows a variable length Unicode string	2GB of text data	4 bytes + number of chars
nvarchar(max)	Allows a variable length Unicode string	2E + 31 characters	2 bytes + number of chars
nvarchar	Allows a variable length Unicode string	4,000 characters	2 bytes + number of chars
nchar	Allows a fixed length Unicode string	4,000 characters	Defined width * 2

Binary Data Types

This section of the article will talk about binary data types. These data types allow binary values of fixed and variable length. Refer to the below table.

Data Type	Description / Maximum Size	
	Description	**Maximum Size**
image	Allows a variable length binary data	2,147,483,647 bytes
varbinary(max)	Allows a variable length binary data	2E + 31 bytes

varbinary**[(length)]**	Allows a variable length binary data	8,000 bytes
binary**[(length)]**	Allows a fixed length binary data	8,000 bytes

B. Number

SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression has a related data type in SQL. You can use these data types while creating your tables. You can choose a data type for a table column based on your requirement.

Number types:

Numeric Data Types

This section of the article will talk about the numeric data types. These data types allow both signed and unsigned integers. I have divided the numeric data types into the following two sections:

- Exact Numeric Data Types
- Approximate Numeric Data Types

Exact Numeric Data Types

Data Type	Description / Range			Storage
	Description	**FROM**	**TO**	
bit	Integers which can either be 0, 1, or NULL.			–
tinyint	Allows whole numbers	0	255	1 byte
smallint	Allows whole numbers	-32,768	32,767	2 bytes
int	Allows whole numbers	-2,147,483,648	2,147,483,647	4 bytes
bigint	Allows whole numbers	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	8 bytes
numeric(p,s)	Allows a numeric value. Where '**p**' is **precision value** and '**s**' is **scale value**	-10^38 +1	10^38 -1	5-17 bytes
decimal(p,s)	Allows a decimal value. Where '**p**' is **precision value** and '**s**' is **scale value**	-10^38 +1	10^38 -1	5-17 bytes
smallmoney	Allows data as currency	-214,748.3648	+214,748.3647	4 bytes
money	Allows data as currency	-922,337,203,685,477.5808	922,337,203,685,477.5807	8 bytes

Now, let us look into Approximate Numeric Data Types.

Approximate Numeric Data Types

Data Type	**Description / Range**			**Storage**
	Description	**FROM**	**TO**	
float(n)	Allows Floating precision number data	-1.79E + 308	1.79E + 308	4 or 8 bytes
real	Allows Floating precision number data	-3.40E + 38	3.40E + 38	4 bytes

C. Date

Date & Time Data Types

This section of the article will talk about the date and time data types. These data types allow different formats of date and time. Refer to the below table.

Data Type	**Description / Range**			**Storage**
	Description	**FROM**	**TO**	
date	Stores date in the format of Year, Month & Days.	January 1, 0001	December 31, 9999	3 bytes
time	Stores time in the format of Hours, Minutes & Seconds.			3-5 bytes
datetime	Stores both date and time(with an accuracy of 3.33 milliseconds)	January 1, 1753	December 31, 9999	8 bytes
datetime2	Stores both date and time(with an accuracy of 100 nanoseconds)	January 1, 0001	December 31, 9999	6-8 bytes
smalldatetime	Stores both date and time(with an accuracy of 1 minute)	January 1, 1900	June 6, 2079	4 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset			8-10 bytes
timestamp	Stores a unique number which gets updated every time a row gets created or modified.			

LO 2.1 – Design an entity relationship diagram (ERD)

- Topic 1: Database relationship description

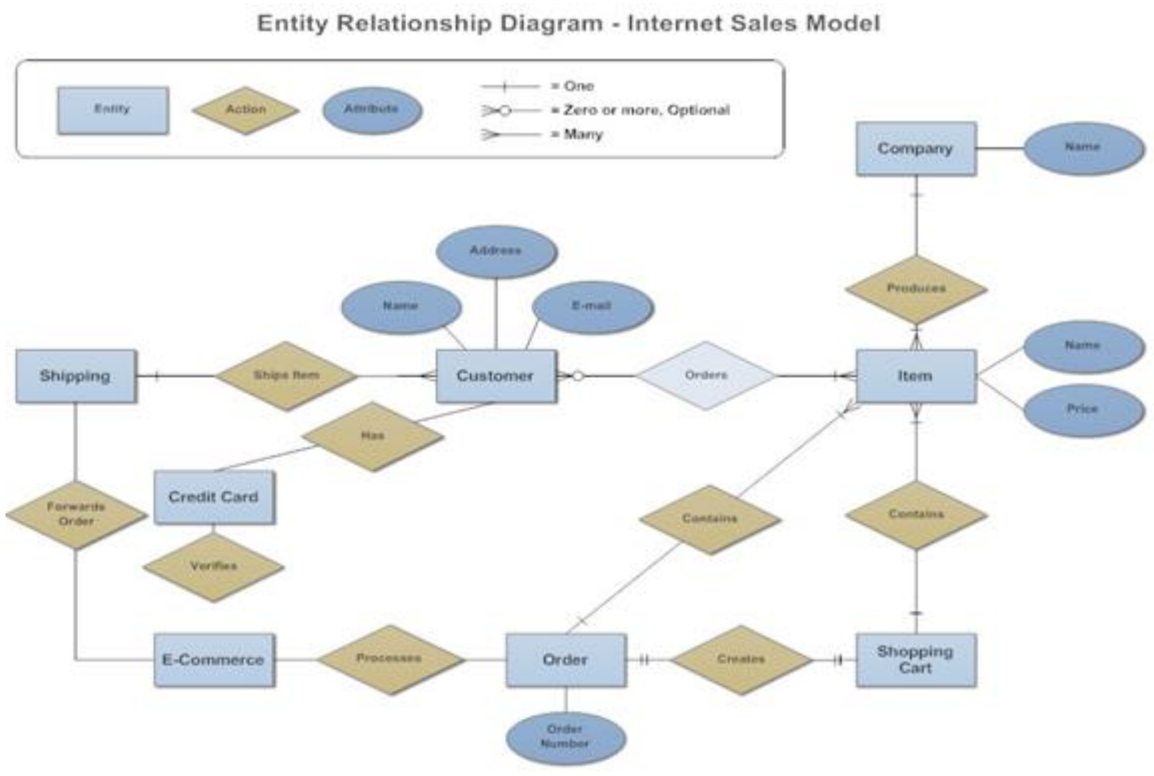
A. ER Diagrams

What is an Entity Relationship Diagram (ERD)?

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

ER diagrams are used to sketch out the design of a database.



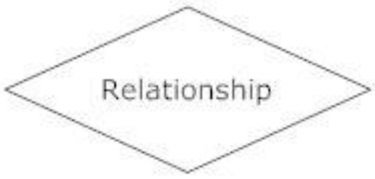
Common Entity Relationship Diagram Symbols

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

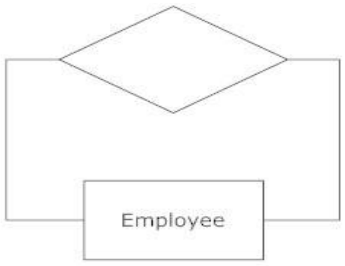
- **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information. A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Entity
Entity

- **Actions**, which are represented by diamond shapes, show how two entities share information in the database.

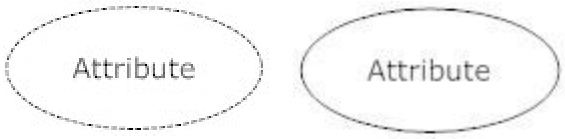


In some cases, entities can be self-linked. For example, employees can supervise other employees.

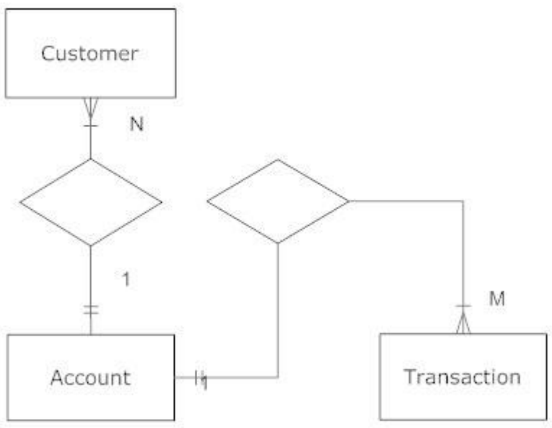


- **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

  A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values. A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.

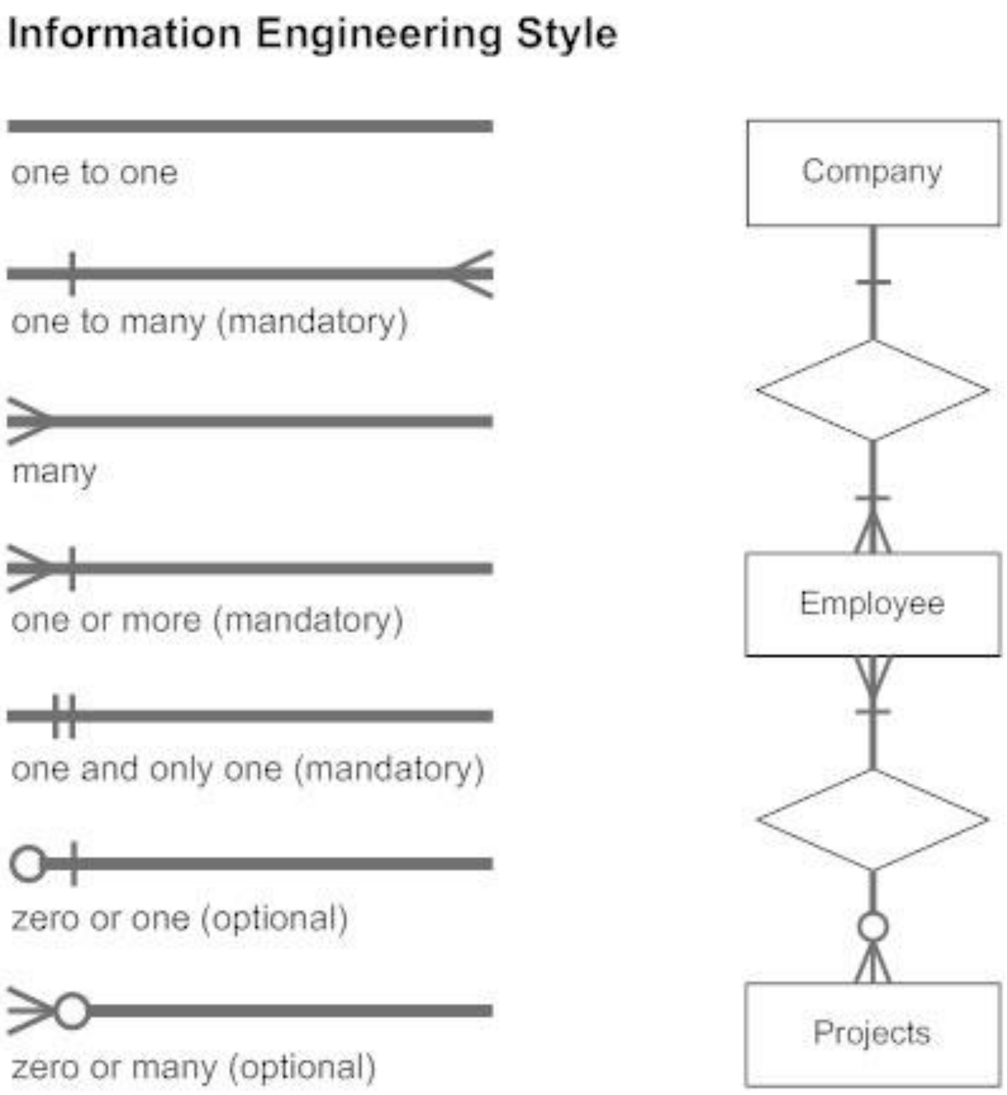


- **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.
- **Cardinality** specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.

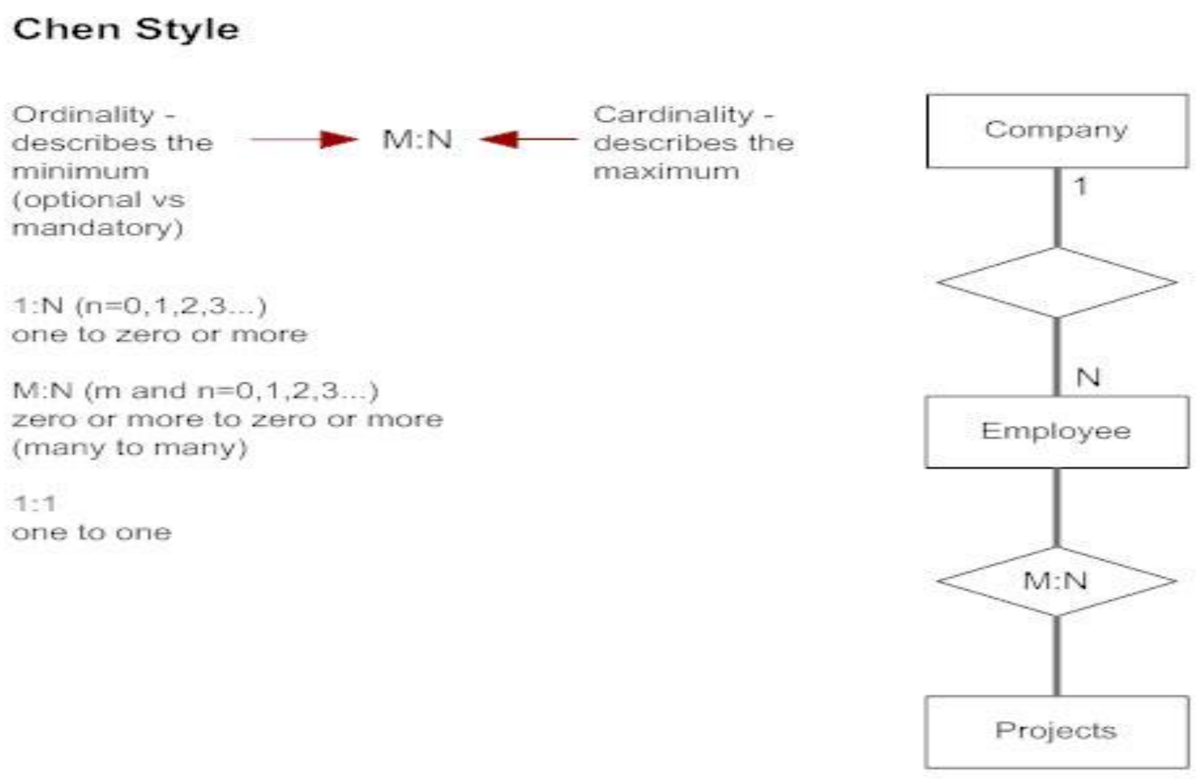


- There are many notation styles that express cardinality.

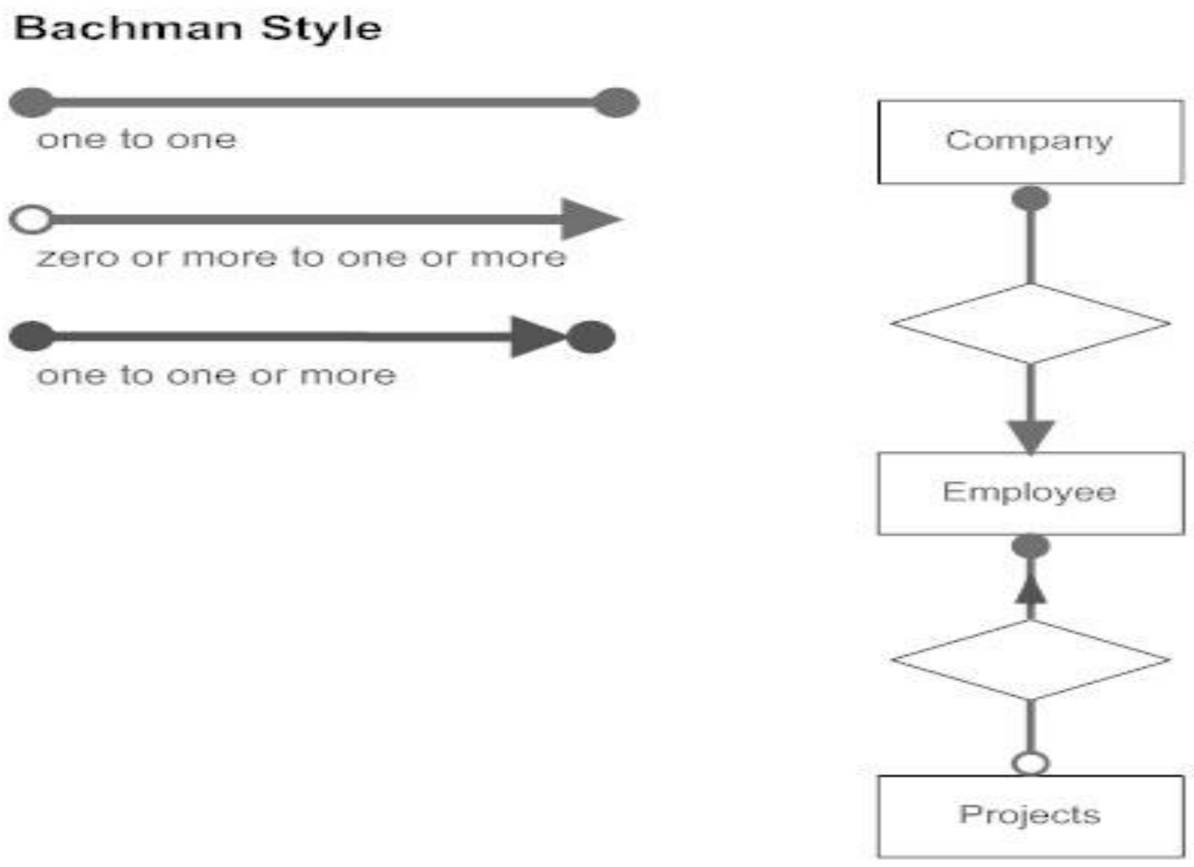
  Information Engineering Style



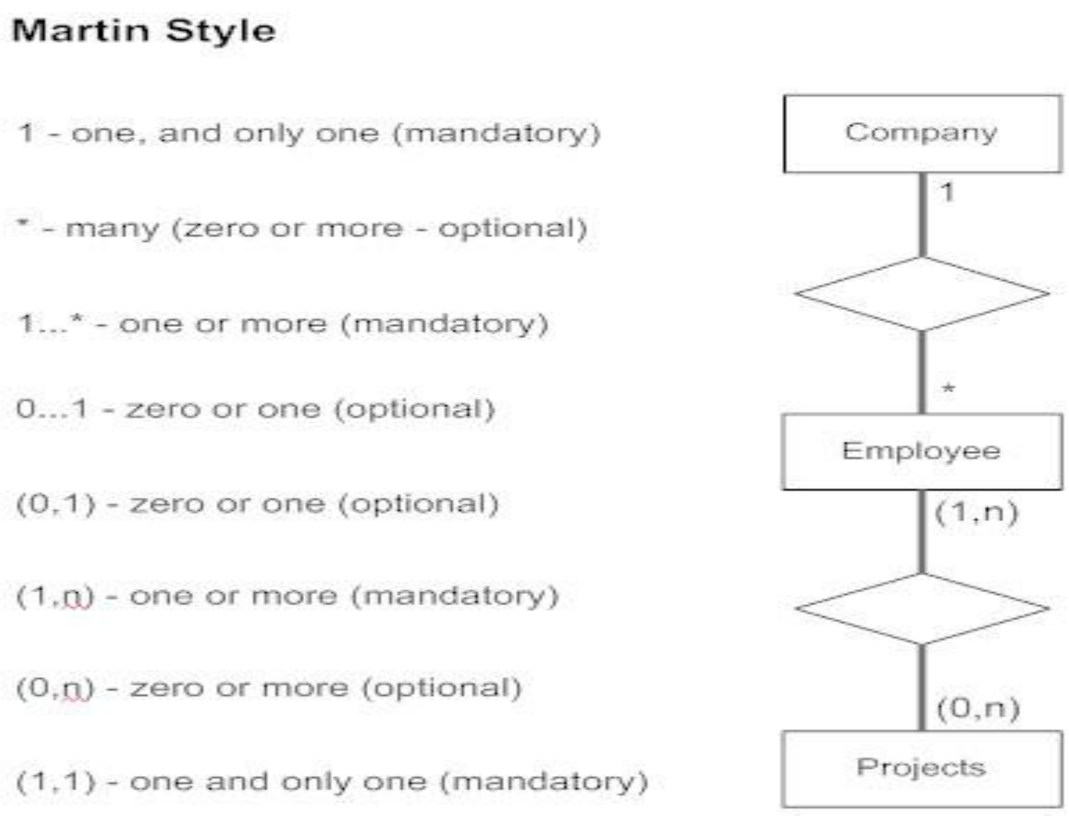
Chen Style


Chen Style
Ordinality - describes the minimum (optional vs mandatory)
M:N
Cardinality - describes the maximum
1:N (n=0,1,2,3...) one to zero or more
M:N (m and n=0,1,2,3...) zero or more to zero or more (many to many)
1:1 one to one
Company
1
N
Employee
M:N
Projects

Bachman Style



Martin Style


Martin Style

1 - one, and only one (mandatory)

* - many (zero or more - optional)

1...* - one or more (mandatory)

0...1 - zero or one (optional)

(0,1) - zero or one (optional)

(1,n) - one or more (mandatory)

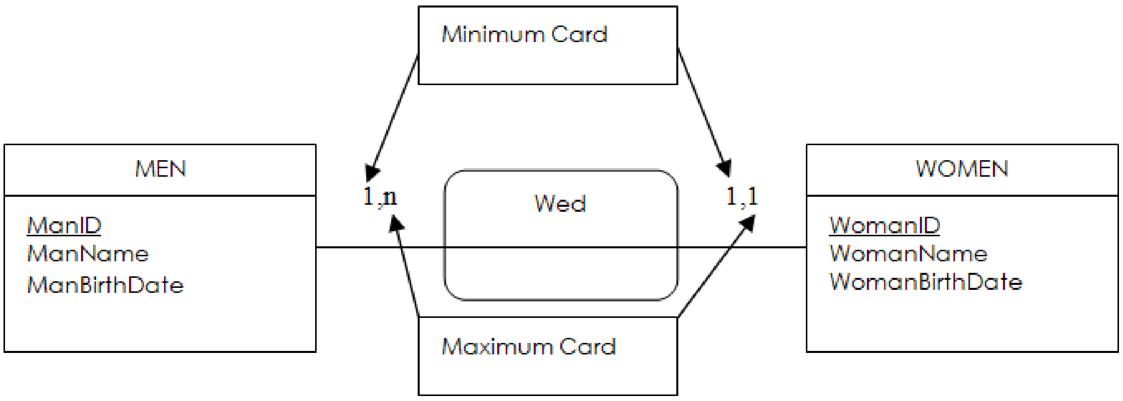
(0,n) - zero or more (optional)

(1,1) - one and only one (mandatory)

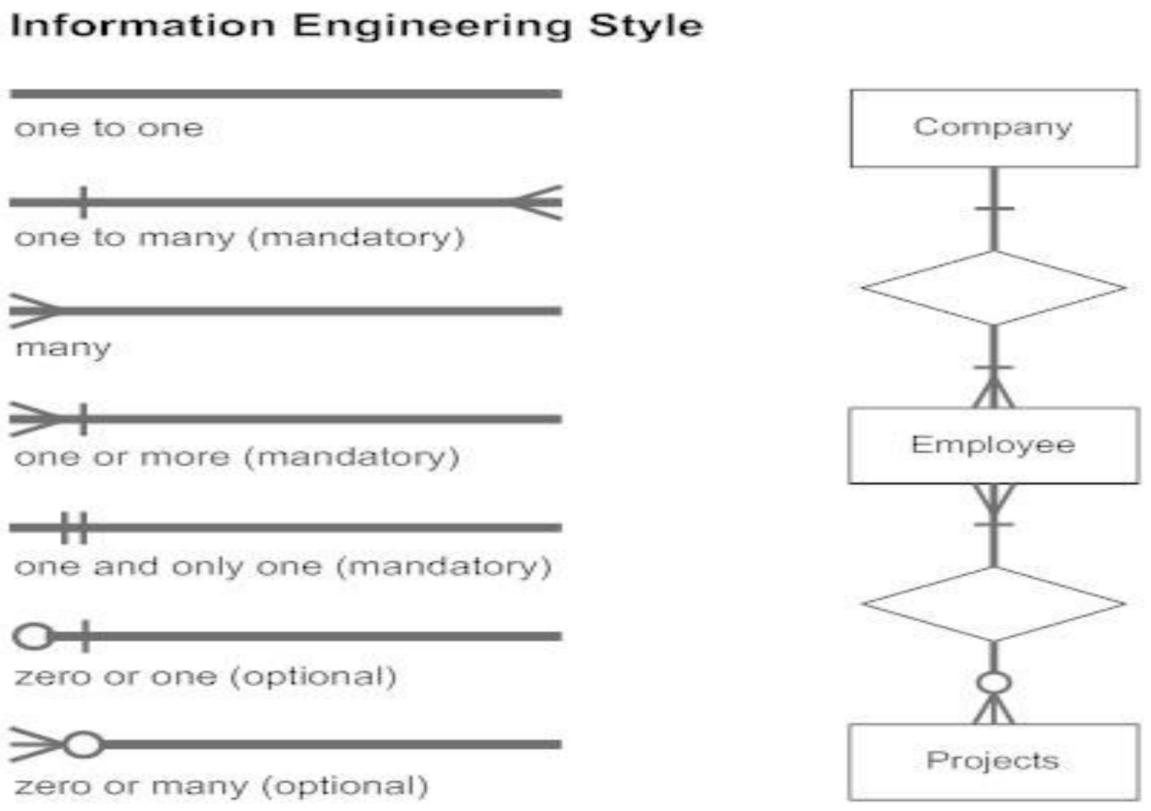
Cardinalities :

- The minimum cardinality which is the minimum number of times that an occurrence participates in a relationship
- The maximum cardinality which is the maximum number of times that an occurrence participates in a relationship.

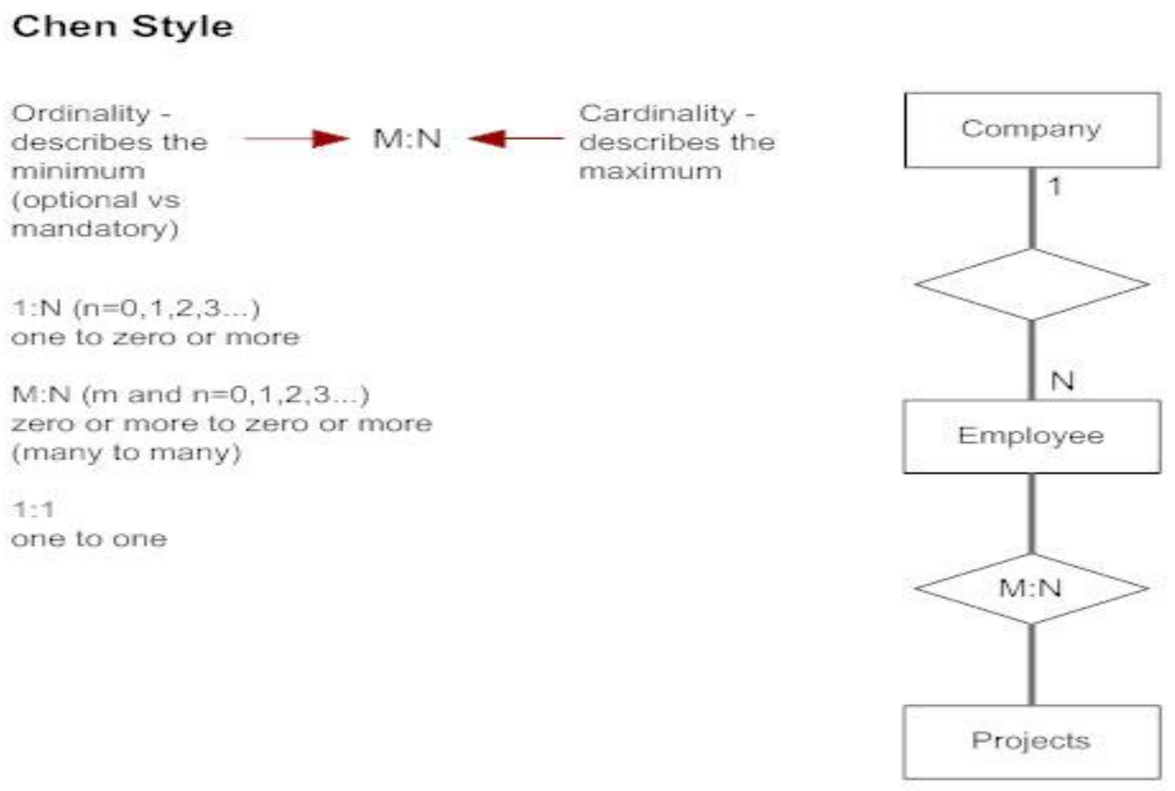
Example:



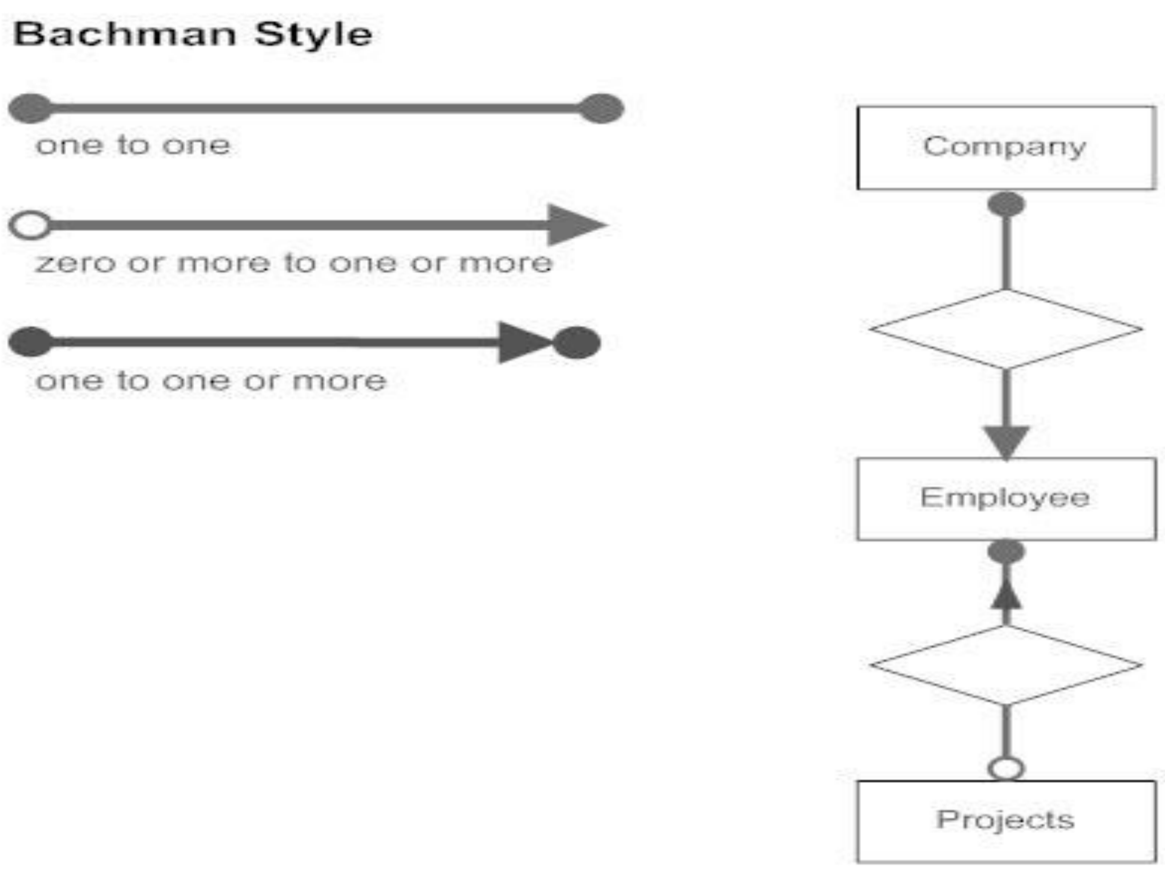
- There are many notation styles that express cardinality.


Information Engineering Style
one to one
one to many (mandatory)
many
one or more (mandatory)
one and only one (mandatory)
zero or one (optional)
zero or many (optional)
Company
Employee
Projects

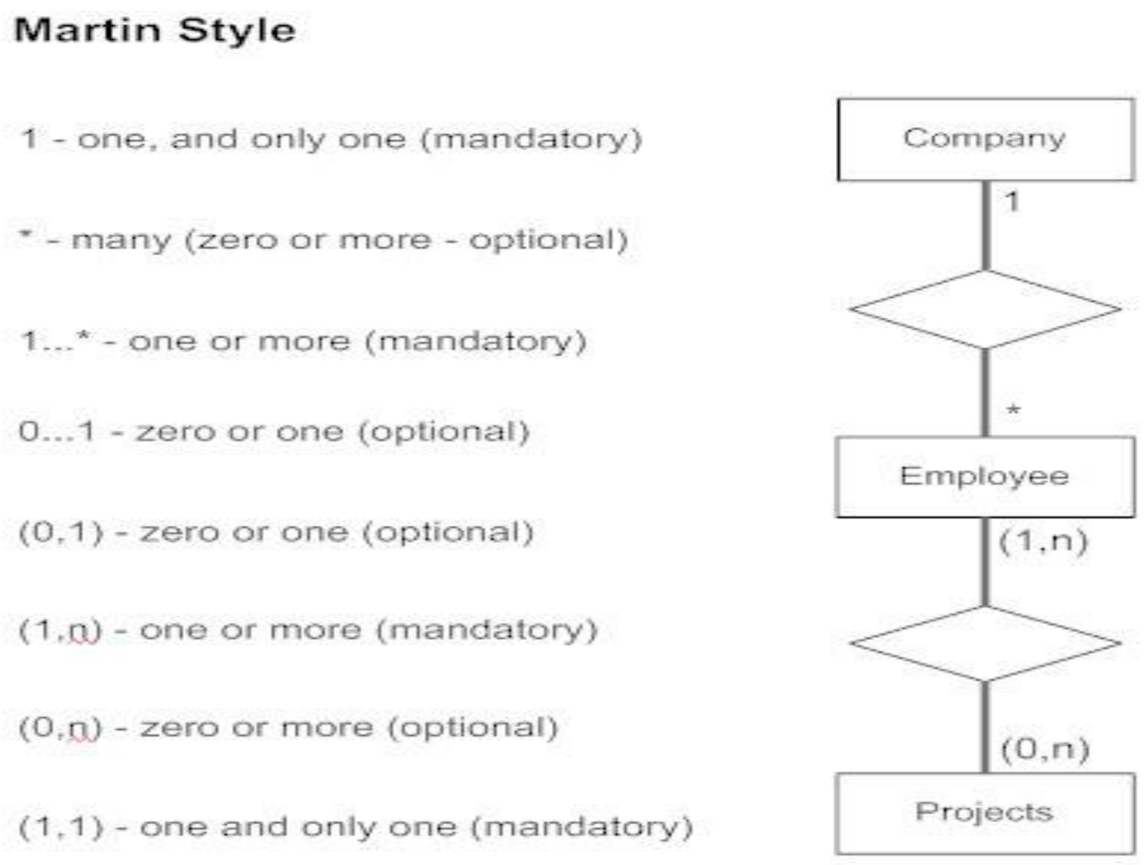
Chen Style


Chen Style
Ordinality - describes the minimum (optional vs mandatory)
M:N
Cardinality - describes the maximum
1:N (n=0,1,2,3...) one to zero or more
M:N (m and n=0,1,2,3...) zero or more to zero or more (many to many)
1:1 one to one
Company
1
N
Employee
M:N
Projects

Bachman Style



Martin Style


Martin Style
1 - one, and only one (mandatory)
* - many (zero or more - optional)
1...* - one or more (mandatory)
0...1 - zero or one (optional)
(0,1) - zero or one (optional)
(1,n) - one or more (mandatory)
(0,n) - zero or more (optional)
(1,1) - one and only one (mandatory)
Company
1
*
Employee
(1,n)
(0,n)
Projects

B. Types of Relationships:

B.1:one to one

A **row** in **table** A can have only one matching row in table B, and vice versa.

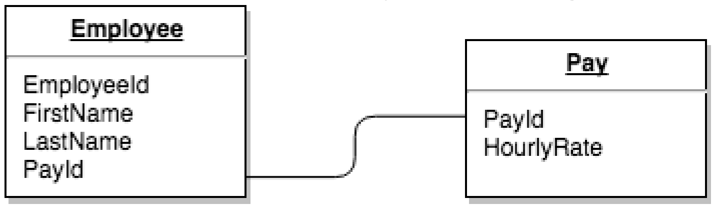


Figure: *Example of a one-to-one relationship*

This is not a common relationship type, as the data stored in table B could just have easily been stored in table A. However, there are some valid reasons for using this relationship type. A one-to-one relationship can be used for security purposes, to divide a large table, and various other specific purposes.

In the above example, we could just as easily have put anHourlyRate field straight into the Employee table and not bothered with the Pay table. However, hourly rate could be sensitive data that only certain database users should see. So, by putting the hourly rate into a separate table, we can provide extra security around the Pay table so that only certain users can access the data in that table.

B.2:One-to-Many (or Many-to-One)

This is the most common relationship type. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.

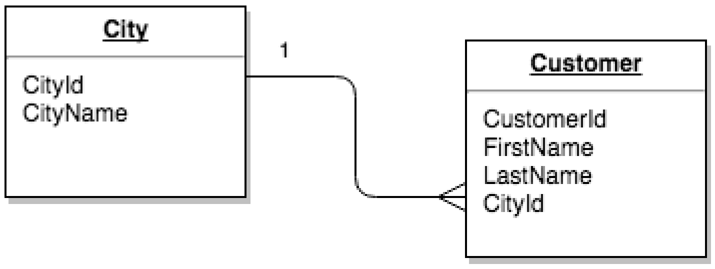


Figure:*Example of one-to-many relationship*

One-to-Many relationships can also be viewed as Many-to-One relationships, depending on which way you look at it.

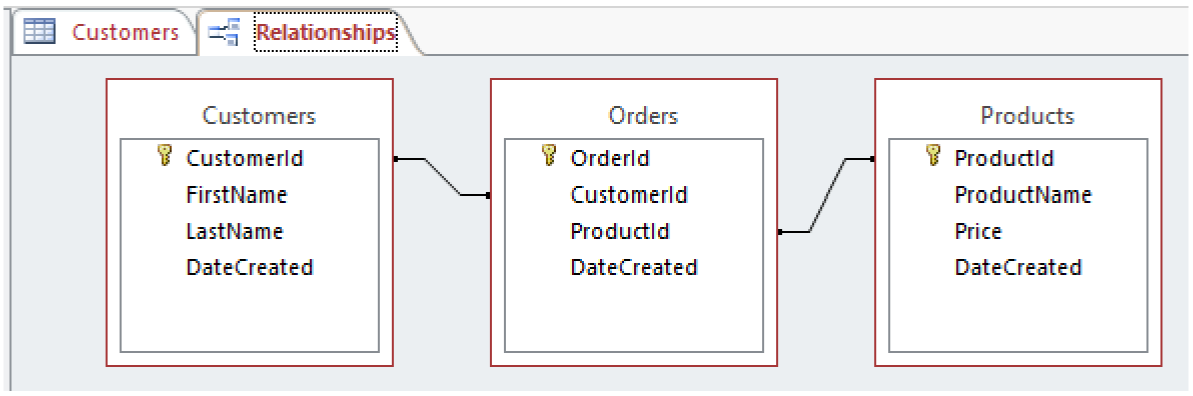
In the above example, the Customer table is the “many” and the City table is the “one”. Each customer can only be assigned one city. One city can be assigned to many customers.

B.3: Many -to- many

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa.

A many-to-many relationship could be thought of as two one-to-many relationships, linked by an intermediary table.

The intermediary table is typically referred to as a “junction table” (also as a “cross-reference table”). This table is used to link the other two tables together. It does this by having two fields that reference the primary key of each of the other two tables.
The following is an example of a many-to-many relationship:



This is the Relationships tab that is displayed when you create a relationship Microsoft Access. In this case, a many-to-many relationship has just been created. The Orders table is a junction table that cross-references the Customers table with the Products table.

So in order to create a many-to-many relationship between the Customers table and the Products table, we created a new table called Orders.

In the Orders table, we have a field called CustomerId and another called ProductId. The values that these fields contain should correspond with a value in the corresponding field in the referenced table. So any given value in Orders.CustomerId should also exist in the Customer.CustomerId field. If this wasn’t the case, then we could have orders for customers that don’t actually exist. We could also have orders for products that don’t exist. Not good referential integrity.
Most database systems allow you to specify whether the database should enforce referential integrity. So, when a user (or a process) attempts to insert a foreign key value that doesn’t exist in the primary key field, an error will occur.
In our example, Orders.CustomerId field is a foreign key to the Customers.CustomerId (which is the primary key of that table). And the Orders.ProductId field is a foreign key to the Products.ProductId field (which is the primary key of that table).

A relationship is established between two database tables when one table has a foreign key that references the primary key of another table. This is the basic concept behind the term relational database.

How a Foreign Key Works to Establish a Relationship?

Let's review the basics of primary and foreign keys. A primary key uniquely identifies each record in the table. It is a type of candidate key that is usually the first column in a table and can be automatically generated by the database to ensure that it is unique.

A foreign key is another candidate key (not the primary key) used to link a record to data in another table. For example, consider these two tables that identify which teacher teaches which course.

Here, the Courses table's primary key is Course_ID. Its foreign key is Teacher_ID:

Course_ID	**Course_Name**	**Teacher_ID**
Course_001	Biology	Teacher_001
Course_002	Math	Teacher_001
Course_003	English	Teacher_003

Courses

You can see that the foreign key in Courses matches a primary key in Teachers:

Teacher_ID	**Teacher_Name**
Teacher_001	Carmen
Teacher_002	Veronica
Teacher_003	Jorge

Teachers

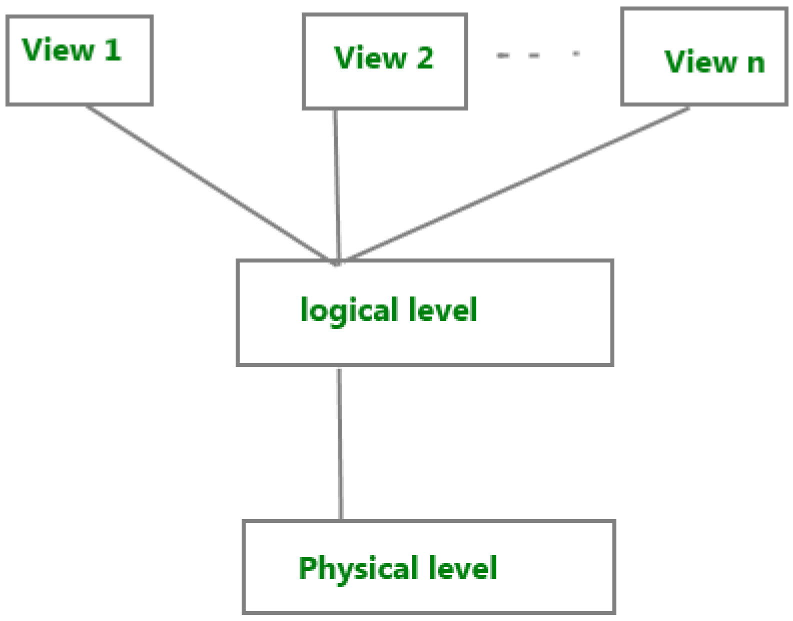
We can say that the Teacher_ID foreign key has helped to establish a *relationship*between the Courses and the Teachers tables.

C. **Entity – Attributes definition (click this link below to learn them)we have already seen their definitions at LO 1.1 – Define database key terms.**

Topic 2: Description of database design abstraction levels

Data Abstraction in DBMS

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.



We have three levels of abstraction:

1. **Physical level**: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.
2. **Logical level**: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.
3. **View level**: Highest level of data abstraction. This level describes the user interaction with database system.

Example: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

DBMS Methodology

- A structured approach that uses procedures, techniques, tools, and documentation help to support and make possible the process of design is called **Design Methodology**.
- A design methodology encapsulates various phases, each containing some stages, which guide the designer in the techniques suitable at each stage of the project. A design methodology also helpsthe designer to plan, manage, control, and evaluate database development and managing

projects. Furthermore, it is a planned approach for analyzing and modeling a group of requirements for a database in a standardized and ordered manner.

The methodology is depicted in three main phases of database design, namely: conceptual, logical, and physical design.

The primary aim of each phase is as follows:

- **Conceptual database design**: to build the conceptual representation of the database that has the identification of the important entities, relationships, and attributes.
- **Logical database design**: to convert the conceptual representation to the logical structure of the database that includes designing the relations.
- **Physical database design**: to decide how the logical structure is to be physically implemented (as base relations) in the target Database Management System (DBMS).

I. CONCEPTUAL MODEL OF DATA (CMD)

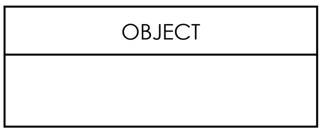
1. Concepts

1.1. Object

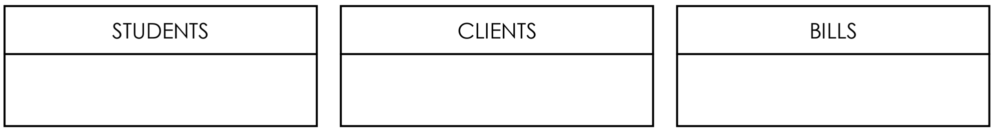
An object is an entity which exists inside the studying information system and which has an interest for the system to be designed.

An object should have fields which characterize it.

Representation



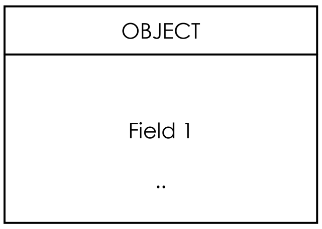
Examples



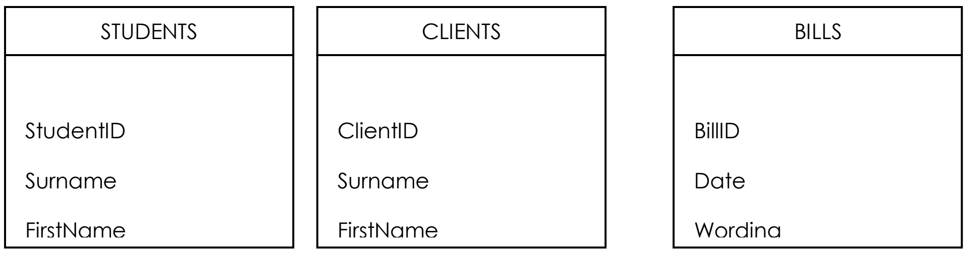
1.2. Field

A field or article or attribute is a propriety which characterizes an object.

Representation



Examples



Note: When you are naming your fields you should:

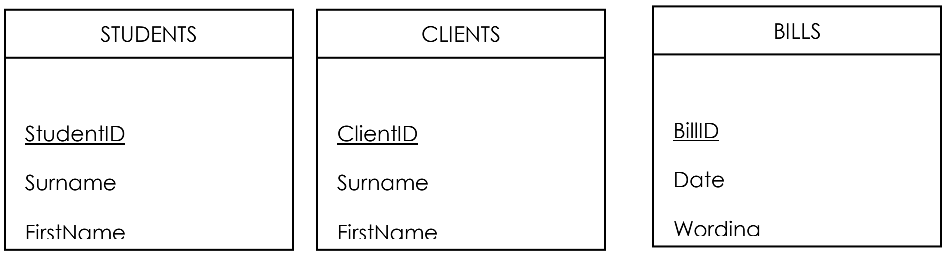
- ✓ Give short and simple names
- ✓ Give explicit names
- ✓ Avoidspaces and accents
- ✓ Avoid dash (-), if yes use underscore (_)

1.3. Identifier

An identifier is a specific field which enables to identify an object's occurrence individually.

An identifier should always be underlined.

Examples



A given student can just be identified by his student number (studentID) because we can find many students with the same name, the same birthday and even in the same promotion.

1.4. Occurrence

An occurrence is an individualized object

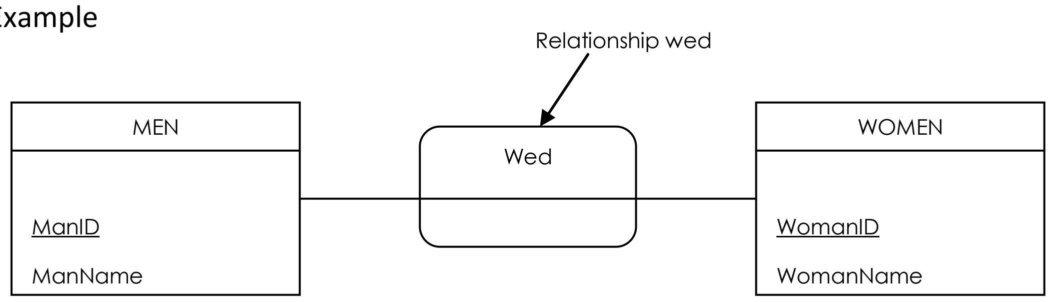
Exemple :

- KEZA Martine who has the ID number 13254 is an occurrence of the object students
- The bill number 12544562 is an occurrence of the object bills

1.5. Relationship or Association

A relationship or an association is a link which materializes a linkage between occurrences of two or more than two objects.

Example



Rememberthat :

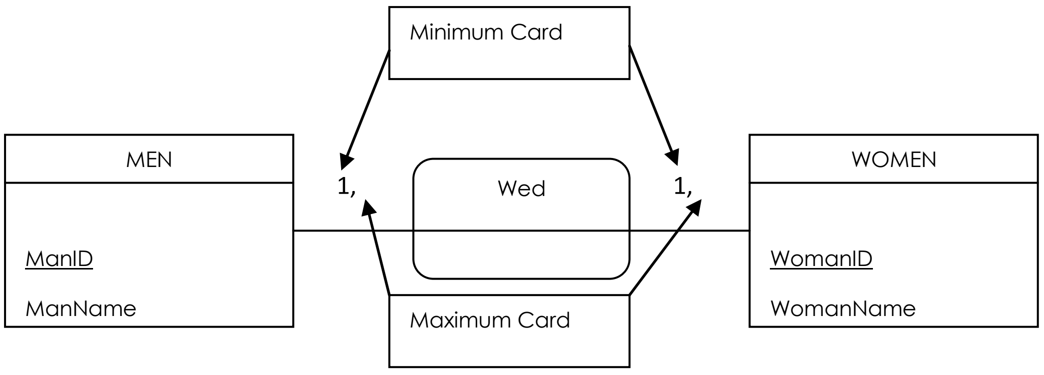
✓ We differentiate ManName and WomanName to avoid the confusion during programming

1.6. Cardinalities

We have :

- The minimum cardinality which is the minimum number of times that an occurrence participates in a relationship.
- The maximum cardinality which is the maximum number of times that an occurrence participates in a relationship.

Example



2. Management Rule

A management rule is a functional constraint in the studying Information System. The overall functions, processes and procedures of a given enterprise are driven by management rules. The most important management rules are stated by the decisional system.

Therefore, during a computerization process the software engineers should be driven in their assessment by management rules. They do not need to create any management rule. They just use existing rules.

2.1. Examples of management rules

- A student cannot take more than 18 credits during one semester at AUCA
- Student who paid his school fees in one time must have a reduction of 3% on his payment.

Management rules can even be simple things: « Students are not allowed to use this WC »

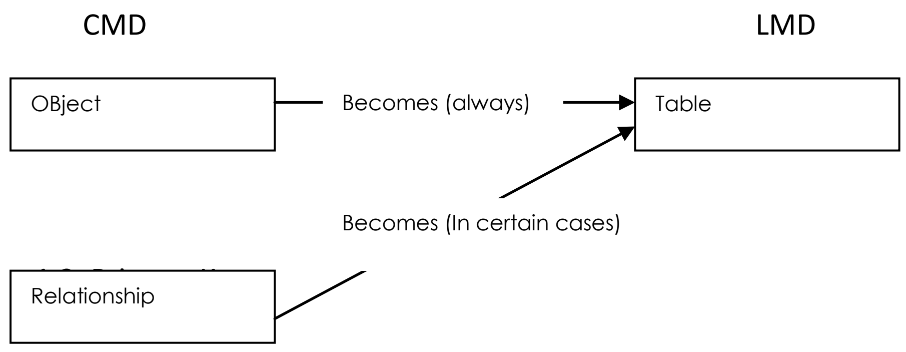
II. LOGICAL MODEL OF DATA (LMD)

The LMD is the database architecture. It is obtained using the CMD. It is the LMD which is implemented to have the database.

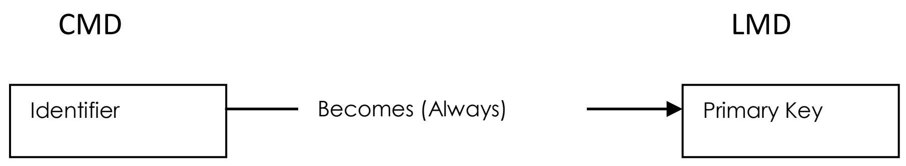
1. Concepts

1.1. Table

In the LMD, the object of the CMD always becomes a table and the relationship can also become a table in certain cases.

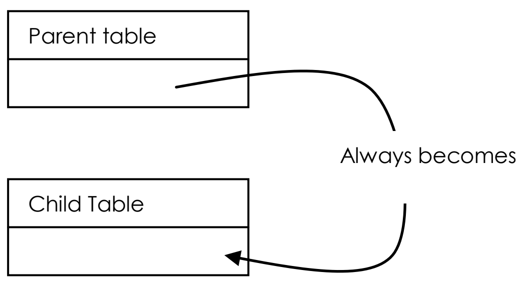


In the LMD, the identifier of the CMD always becomes a primary key



1.2. Secondary or Foreign Key

The foreign key is the parent table primary key inherited by the child table.

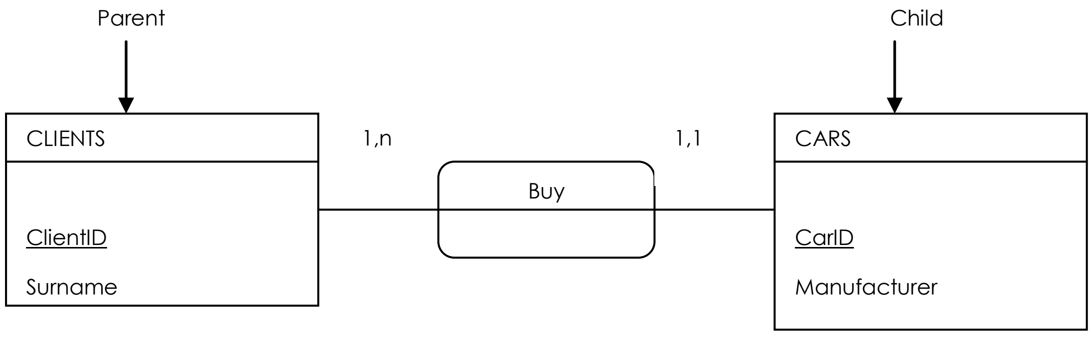


2. Translation of CMD to LMD

2.1. Case of Parent-Child Relationship

In this case, the child table inherits the primary key of the parent table to become its foreign key.

Ex:Translate the following CMD to LMD

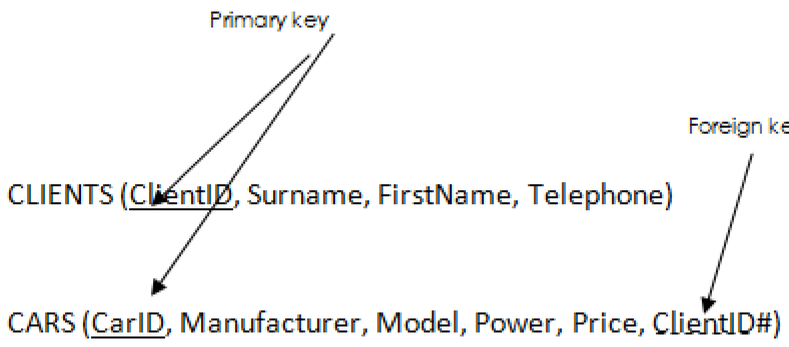


NB: We are driven by the cardinalities:

- ✓ (1, n) in parent side
- ✓ (1,1) in childside

Solution

This isthe LMD :



CLIENTS (<u>ClientID</u>, Surname, FirstName, Telephone)

CARS (<u>CarID</u>, Manufacturer, Model, Power, Price, ClientID#)

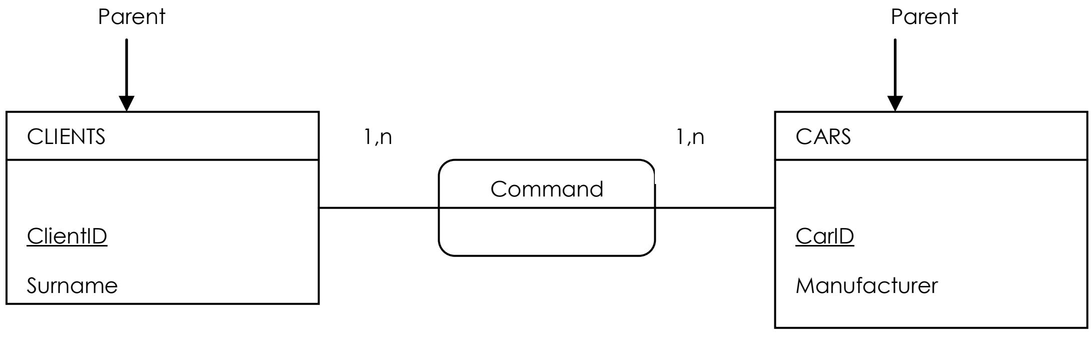
Rememberthat:

- ✓ The primary key is always underlined
- ✓ The foreign key is always underlined using dashes followed by #
- ✓ A child table can have several foreign keys

2.2. Case of Parent-Parent Relationship

In this case, the tables remain with their fields, but the relationship becomes a table which will get the primary keys of tables as its primary key: It is a concatenated primary key.

Ex:Translate the following CMD to LMD:



Remember that :

✓ The cardinality in CARS side has changed: Several Clients can indeed command the same Car, but just one of them will buy it.

Solution

This is the LMD:

Concatenated Primary key

CLIENTS (<u>ClientID</u>, Surname, FirstName, Telephone)

COMMAND (<u>ClientID, CarID</u>)

CARS (<u>CarID</u>, Manufacturer, Model, Power, Price)

EXERCISES OF APPLICATION FOR CMD AND LMD

EXAMPLE 1:

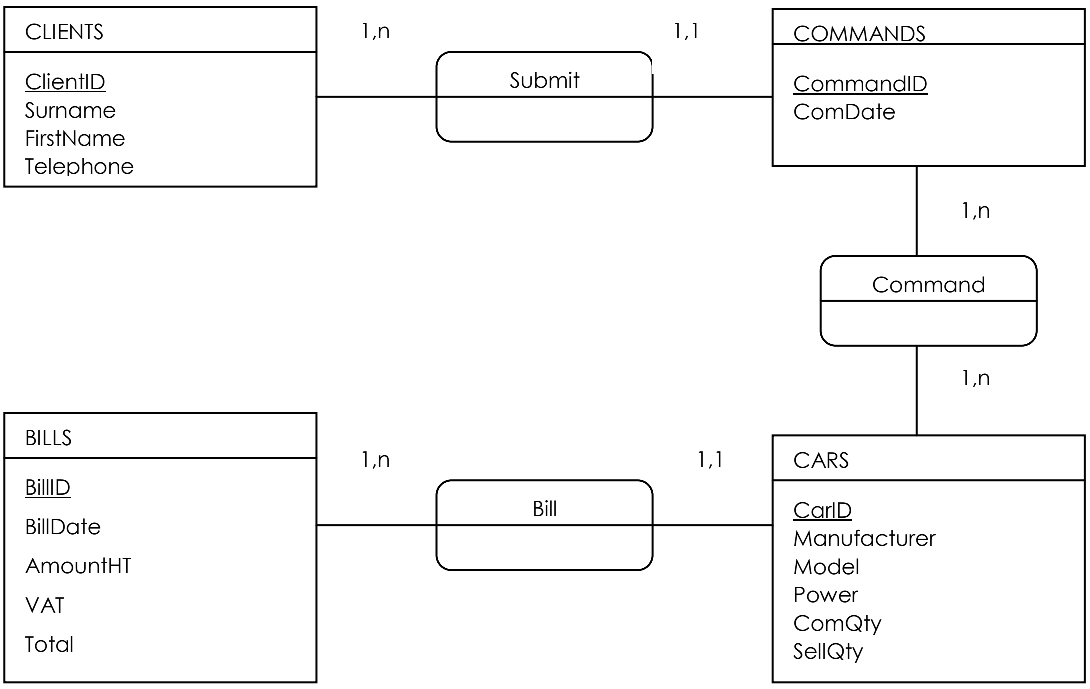
RWANDA MOTORS needs to have an IT based sales system. During the interview the following management rules were collected:

- A car can be commanded by several customers
- A car can be bought just by one client
- A bill can contain several cars

Design the CMD and the LMD.

Solution

1. CMD



2. LMD

CLIENTS (ClientID, Surname, FirstName, Telephone)

COMMANDS (CommandID, ComDate, ClientID#)

COMMAND (CommandID, CardID)

CARS (CardID, Manufacturer, Model, Power, ComQty, SellQty, Price, BillID#)

BILLS (BillID, BillDate, AmountHT, VAT, Total)

III. PHYSICAL DATABASE DESIGN

- **Physical database design** - to decide how the logical structure is to be physically implemented (as base relations) in the target Database Management System (DBMS).

We have the LMD above and MySQL as a DBMS, now let's go on to create a database.

I am to create the database RWANDAMOTOR

```
Create database RWANDAMOTOR;
```

```
CREATE TABLE CLIENTS(
ClientID varchar(10)  primary key not null,
   Surname Varchar(50),
FirstNamevarchar(30),
   Telephone varchar(20));

   CREATE TABLE COMMANDS(
CommandIDvarchar(20) primary key not null,
ComDate date,
ClientIDvarchar(10),
      FOREIGN KEY (ClientID) REFERENCES CLIENTS (ClientID)
      );
 CREATE TABLE COMMAND(
CarIDvarchar(20),
CommandIDvarchar(20),
PRIMARY KEY (CarID, CommandID)
FOREIGN KEY (CarID) REFERENCES CARS(CarID),
FOREIGN KEY (CommandID) REFERENCES COMMANDS(CommandID));

           CREATE TABLE BILLS(
BillIDvarchar (10) primary key not null,
BillDate date,
AmountHTdecimal(10,2)  ,
              VAT decimal(10,2),
              Total decimal(10,2));
```

```
        CREATE TABLE CARS(
CarIDvarchar(20) primary key not null,
          Manufacturer varchar (50),
          Model varchar(30),
          Power varchar(30),
ComQtyvarchar(20),
SellQtyvarchar (20),
          Price decimal(10,2),
BillIDvarchar(20),
          FOREIGN key (BillID) REFERENCES BILLS(BillID));
```

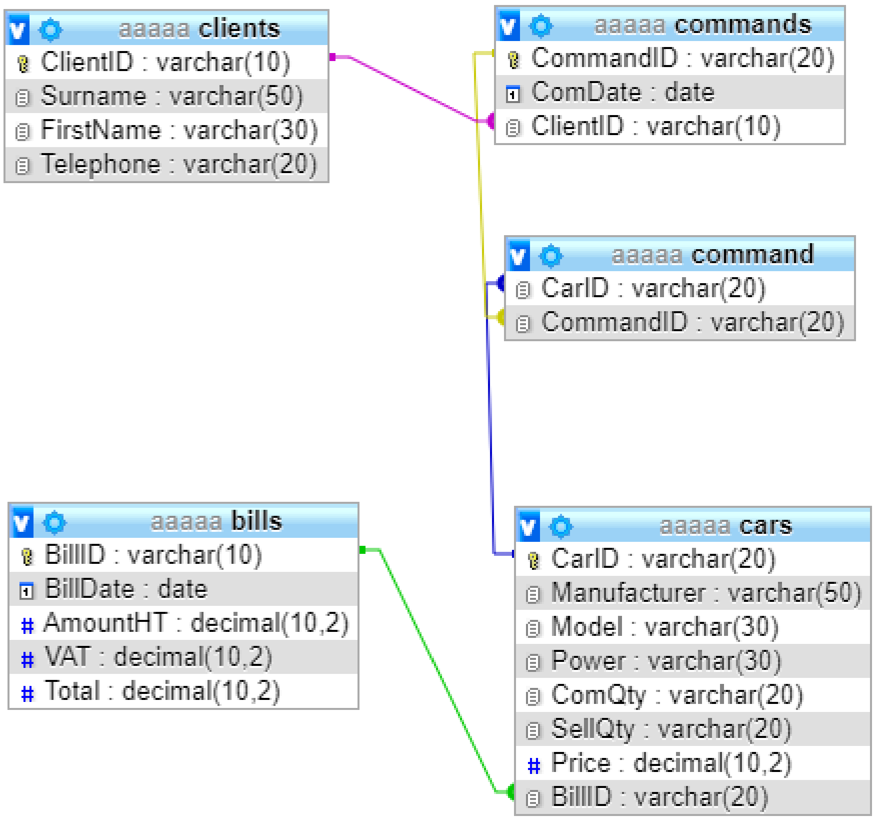


Figure: RWANDAMOTOR designer view

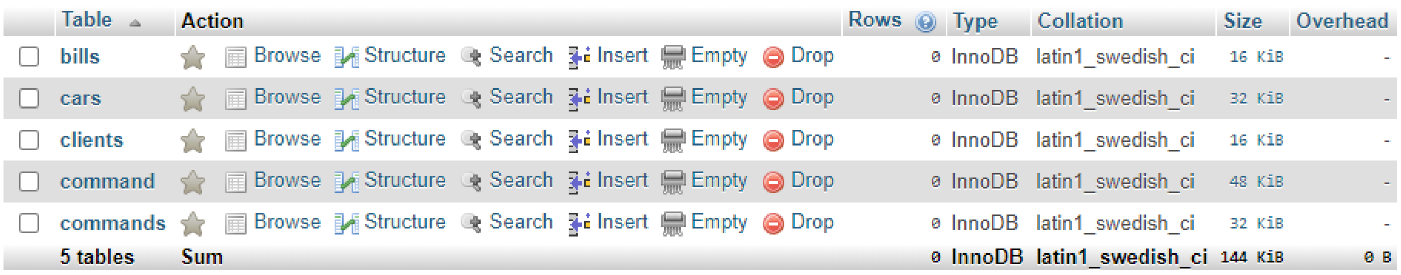


Table	Action						Rows	Type	Collation	Size	Overhead
bills	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
cars	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
clients	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
command	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
commands	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
5 tables	Sum						0	InnoDB	latin1_swedish_ci	144 KiB	0 B

Figure: Structure view

Now you can go ahead and insert some records

Let's insert records in table client

Syntax of inserting data into the database table is as follows-

```
INSERTINTOtable_name(column1,column2,column3,...)
VALUES (value1,value2,value3,...)
```
This form is recommended!

```
INSERT INTO CLIENTS(ClientID,FirstName,Surname,Telephone)
values('C0001','Alice','UWERA','0785979270');

INSERT INTO CLIENTS(ClientID,FirstName,Surname,Telephone)
values('C0002','Theogene','BAHIGIRORA','0785999271');
```

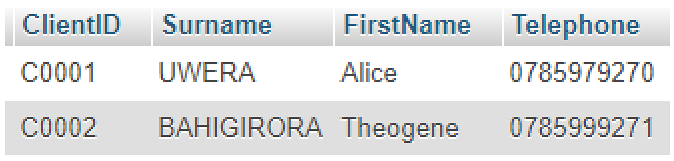
Verify if the records have been inserted

Syntax: SELECT* FROM Table_name;

Type:

```
SELECT* FROM clients;
```

Output:



ClientID	Surname	FirstName	Telephone
C0001	UWERA	Alice	0785979270
C0002	BAHIGIRORA	Theogene	0785999271

Topic 2: Description of system metadata

- ✓ **Types of metadata**

What is Metadata?

Metadata is critically important for website and database management

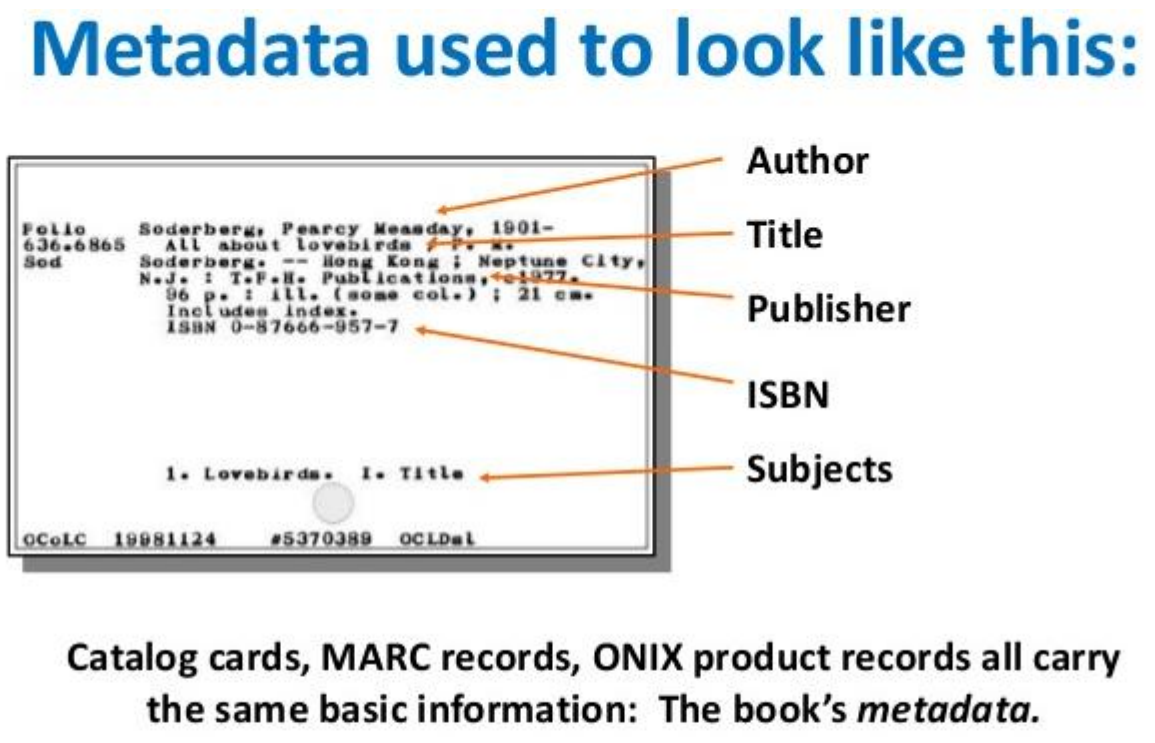
Metadata is "data [information] that provides information about other data."
Metadata is data about data. In other words, it's information that's used to describe the data that's contained in something like a web page, document, or file. Another way to think of metadata is as a short explanation or summary of what the data is.
A simple example of metadata for a document might include a collection of information like the author, file size, the date the document was created, and keywords to describe the document. Metadata for a music file might include the artist's name, the album, and the year it was released.
Metadata represents behind-the-scenes information that's used everywhere, by every industry, in multiple ways. It's ubiquitous in information systems, social media, websites, software, music services, and online retailing. Metadata can be created manually to pick and choose what's included, but it can also be generated automatically based on the data.

Meta-data - "Data about data." In a DBMS context, data stored in columns of a table have certain attributes, such as the *type*, *length*, *description* or other characteristics that allow the DBMS to process the data meaningfully, or allow the users to understand it better.

Metadata in DBMS – Overview and Types

Metadata in DBMS is the data (details/schema) of any other data. It can also be defined as data about data. The word ‘Meta’ is the prefix that is generally the technical term for self-referential. In other words, we can say that Metadata is the summarized data for the contextual data.

For example, consider the index of any book or novel; it serves as the metadata that leads us to the detailed data inside the book.


Metadata used to look like this:
Author
Title
Publisher
ISBN
Subjects
Catalog cards, MARC records, ONIX product records all carry the same basic information: The book’s *metadata.*

```
Folio      Soderberg, Pearcy Meaaday, 1901-
636.6865     All about lovebirds / P. M.
Sod        Soderberg. -- Hong Kong ; Neptune City,
           N.J. : T.F.H. Publications, c1977.
             96 p. : ill. (some col.) ; 21 cm.
             Includes index.
             ISBN 0-87666-957-7

             1. Lovebirds.  I. Title

OCoLC  19981124     #5370389  OCLDel
```

For instance, Metadata in DBMS has the following definitions:

- Metadata is the value that leads us to the actual data;
- The Metadata is an acronym for the detailed data;
- Metadata is the index of a library in the data warehouse;
- Metadata is a description or schema of the actual data

In a database, information is organized and stored in a structure. This data is arranged in tabular form (rows and columns) so that it can be accessed and reorganized quickly. After the structural arrangement of data, each part has its metadata as an index number in a book for each chapter. This index number is known as the metadata that defines the schema for other data.

✓ **Types of metadata and their uses**

There are several types of metadata according to their uses and domain.

➢ **Technical Metadata**

This type of metadata defines database system names, tables names, table size, data types, values, and attributes. Further technical metadata also includes some constraints such as foreign key, primary key, and indices.

➢ **Business Metadata**

It consists of the ownership of data, changing policies, business rules and regulations, and other business details. This type of metadata is related to a particular business.

➢ **Operational Metadata**

This type includes the data which is currently under any operation. Besides, it represents the data that is used by executive-level managers to perform any task. Also, this type of metadata can be purged, archived, or activated and can also be migrated.

➢ **Descriptive Metadata**

Descriptive metadata describes any file, folder, book, image, or video. It may include details of data such as title, author, date, size, author name, published on, and similarly others.

➢ **Metadata in DBMS**

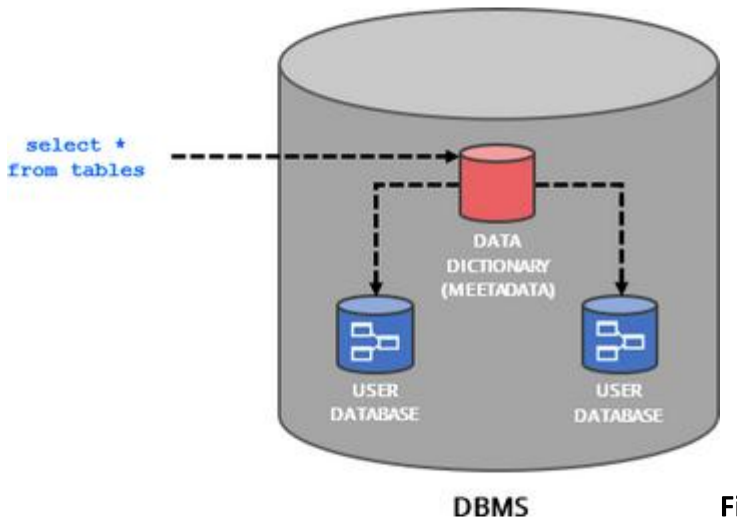


Figure: Metadata in DBMS

Metadata in DBMS refers to the information that describes the schema and other information related to the stored data in the database, including storage, programs, data elements, usage, and additional related information. Let’s check the following types of metadata in DBMS.

Schema

- Tables
- Columns
- Constraints

- Indexes
- Sequences

Physical Execution

- Backup files
- Partitions
- Restored files

Storage

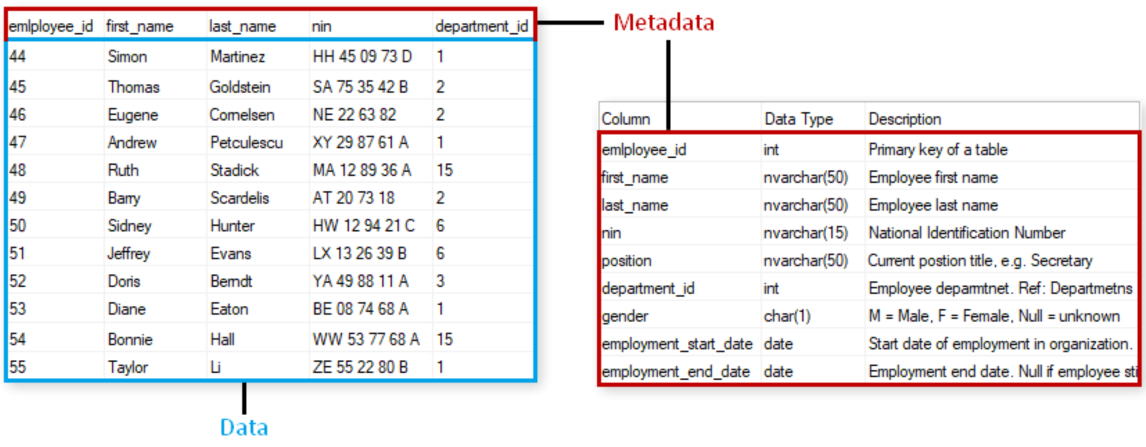
- Table size
- Data type size
- Variable size
- Program size
- Number of rows and columns

Also, Metadata has a significant role in the database. Besides, it acts as a directory in the database for accessing the different files. Further, metadata can also be useful in many ways, like database reporting, auditing, querying, and transforming. It plays a crucial role in DBMS.

Metadata is also essential for interpreting the contents in the database. For example, a set of data with their schema, such as dates and names will get you confused. It becomes hard to identify the data you are interpreting. Besides, if there's metadata for that existing data, you get an idea of the database you are interpreting with.

Basic metadata including rows and columns names that helps a user to easily identify the data being described. And if there's a list of data with no metadata, it could be hard to say what the data is about. Now suppose, there's a column with name 'Expired domains' will let you know that these domains have been expired. Additionally, if there's another column of 'Date', it will let you know the date when it got expired.

An Example of Metadata in DBMS


Metadata
Data

emlployee_id	first_name	last_name	nin	department_id
44	Simon	Martinez	HH 45 09 73 D	1
45	Thomas	Goldstein	SA 75 35 42 B	2
46	Eugene	Comelsen	NE 22 63 82	2
47	Andrew	Petculescu	XY 29 87 61 A	1
48	Ruth	Stadick	MA 12 89 36 A	15
49	Barry	Scardelis	AT 20 73 18	2
50	Sidney	Hunter	HW 12 94 21 C	6
51	Jeffrey	Evans	LX 13 26 39 B	6
52	Doris	Berndt	YA 49 88 11 A	3
53	Diane	Eaton	BE 08 74 68 A	1
54	Bonnie	Hall	WW 53 77 68 A	15
55	Taylor	Li	ZE 55 22 80 B	1

Column	Data Type	Description
emlployee_id	int	Primary key of a table
first_name	nvarchar(50)	Employee first name
last_name	nvarchar(50)	Employee last name
nin	nvarchar(15)	National Identification Number
position	nvarchar(50)	Current postion title, e.g. Secretary
department_id	int	Employee depamtnet. Ref: Departmetns
gender	char(1)	M = Male, F = Female, Null = unknown
employment_start_date	date	Start date of employment in organization.
employment_end_date	date	Employment end date. Null if employee sti

As you can see in the above image, a relational database stores the data as well as metadata in a structured way. The data stored in a structured way is called the Data Dictionary or System Catalog. The above metadata holds the following information:

- Tables (rows and columns)
- Data types
- Values
- Table relationships
- Constraints
- and others

Further, through this information, it’s easier to access or identify any database.

Final Words on Metadata

Metadata is data about the data and not the actual context of data. Hence, it can be made publically available as it only describes the raw data and doesn’t provide any access to it. Metadata gives the idea and summary details about any database which is enough to understand what the file is but does not give the complete instance of that file.

LO 2.2 – Identify the constraints

- Topic 1:Description of database constraints

Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints can either be column level or table level. Column level constraints are applied only to one column whereas table level constraints are applied to the entire table.

Constraints in General

- Think of constraints as database rules.
- All constraint definitions are stored in the data dictionary.
- Constraints prevent the deletion of a table if there are dependencies from other tables.
- Constraints enforce rules on the data whenever a row is inserted, updated, or deleted from a table.
- Constraints are important and naming them is also important.

Creating constraints

- Recall the SQL syntax for creating a table.
- In the CREATE TABLE statement shown, each column and its data type is defined.
- You use the CREATE TABLE statement to establish constraints for each column in the table.
- There are two different places in the CREATE TABLE statement that you can specify the constraint details

- At the column level next to the name and data type
- At the table level after all the column names are listed

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax

```
CREATE TABLE table_name (
   column1 datatype constraint,
   column2 datatype constraint,
   column3 datatype constraint,
   ....
);
```

```
CREATE TABLE clients
(client_number NUMBER,
first_name VARCHAR (14),
last_name VARCHAR(13));
```

- Note that the column-level simply refers to the area in the CREATE TABLE statement where the columns are defined.
- The table level refers to the last line in the statement below the list of individual column names.

The five types of CONSTRAINTS are:

NO	Constraint	Description
1.	**Primary key constraint**	constraint uniquely identifies each record in a table Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).
2.	**Foreign key constraint**	Uniquely identifies a row/record in any another database table
3.	**Unique key constraint**	Ensures that all the values in a column are different.
4.	**Not null constraint**	Ensures that a column cannot have a NULL value.
5.	**Check constraint**	The CHECK constraint ensures that all values in a column satisfy certain conditions.

The syntax to create a foreign key is as follows:

```
alter table yourSecondTableName ADD CONSTRAINT yourConstraintname FOREIGN
KEY(yourForeignKeyColumnName)
referencesyourFirstTableName (yourPrimaryKeyColumnName);
```

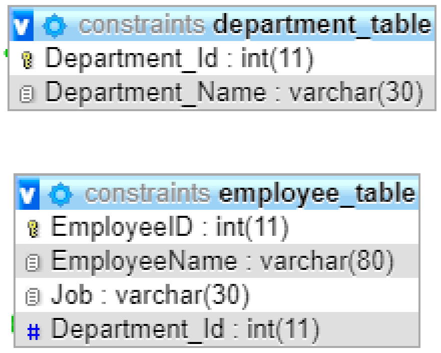
To understand the above syntax, let us create two tables. The query to create the first table is as follows −

```
create table Department_Table
  (
Department_Idint not null auto_increment primary key,
Department_Namevarchar(30)
   );
```

The query to create the second table is as follows:

```
create table Employee_Table
  (
EmployeeIDint not null auto_increment primary key,
EmployeeNamevarchar(80),
Job varchar(30),
Department_Idint not null references department (departmentID)
  );
```

The above Department_Idint, not null references department (departmentID) does not create a foreign key. Now follow the above syntax to create a foreign key.

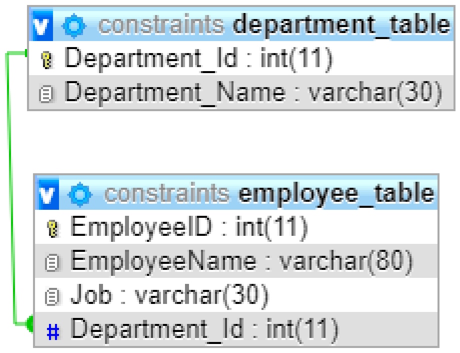


constraints department_table
Department_Id : int(11)
Department_Name : varchar(30)
constraints employee_table
EmployeeID : int(11)
EmployeeName : varchar(80)
Job : varchar(30)
Department_Id : int(11)

Figure: tables without constraint

The query is as follows:

```
alter table Employee_Table ADD CONSTRAINT fk_Department_Id FOREIGN KEY(Department_Id) references Department_Table(Department_Id);
```



constraints department_table
Department_Id : int(11)
Department_Name : varchar(30)
constraints employee_table
EmployeeID : int(11)
EmployeeName : varchar(80)
Job : varchar(30)
Department_Id : int(11)

Figure: Tables with constraints

- Topic 1: Description of database constraints rule

- Constraints that refer to more than one column (a composite key) must be defined at the table level
- The NOT NULL constraint can be specified only at the column level, not the table level
- UNIQUE, PRIMARY KEY, FOREIGN KEY, and CHECK constraints can be defined at either the column or table level
- If the word CONSTRAINT is used in a CREATE TABLE statement, you must give the constraint a name.

A constraint which refers to two columns cannot be defined as part of a column definition. Which of the two column definitions would we put it alongside? It couldn't be both!
A table level constraint must be user-named, because table level constraints must include the keyword CONSTRAINT.

NOT NULL constraints cannot be defined at the table level because the ANSI/ISO SQL standard forbids it

LO 2.3 – Develop data dictionary

- Topic 1:Description of DBMS

Database management system is a collection of interrelated data and set of programs to access those data. Collection of data is called database. Database contains information related to an enterprise. Primary goal of database management system is to provide a way to store and retrieve database information that is convenient and efficient.

Top Database Management Systems

1. Oracle
2. Microsoft SQL Server
3. MySQL

Popular DBMS Software

Here, is the list of some popular DBMS system:

- MySQL
- Microsoft Access
- Oracle
- PostgreSQL

- dBASE
- FoxPro
- SQLite
- IBM DB2
- LibreOffice Base
- MariaDB
- Microsoft SQL Server etc.

A. DBMS key terms

Basic Terminologies/terms used in DBMS

Here, we are going to learn about the some of the **basic terms / terminologies which are used in the database management system (DBMS)**.

1**) What is Data?**

The raw information (facts and figures) input by the user is called Data. Raw means data isn't been processed yet. It has no significance beyond its existence and simply exists. Data means which has no meaning of itself.

For e.g. – **{1}** is a raw data.

2) What is Information?

Processed Data is termed as Information, i.e., data been contextualized, calculated, categorized and condensed. Information means meaningful data by relational connection.

For e.g. – **{Roll No: 1}** is an information.

What is the difference between Data and Information?

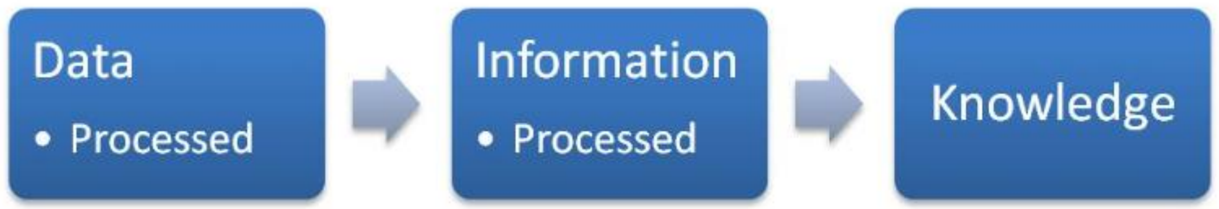
Data	**Information**
Raw Data	Processed Data
It doesn't help in decision making	It helps in decision making
It doesn't required any prerequisite	It require prerequisite as Data
Unorganized	Organized

3) What is Knowledge?

A processed data, i.e. Information when coming into practical use, an understanding of the information through senses, is termed as knowledge. It refers to the practical use of information and involves personal experience. Knowledge is a deterministic process. Learning numbers and alphabets, memorizing key points in a chapter, etc. are considered to be as knowledgeable.

For example

Primary school children memorize knowledge of the time table. They can tell you that **"3 x 3 = 9"** because they have memorized it. But when asked what is **"1580 x 337"**, they cannot respond correctly because that entry is not in their time table. To correctly answer such a question requires reasoning and logical aptitude that is only included in the next level understanding.



- **Database**

A database is a named collection of tables. A database can also contain views, indexes, sequences, data types, operators, and functions. Other relational database products use the term catalog.

- **Command**

A command is a string that you send to the server in hopes of having the server do something useful. Some people use the word statement to mean command. The two words are very similar in meaning and, in practice, are interchangeable.

- **Query**

A query is a type of command that retrieves data from the server.

- **Table (relation, file, class)**

A table is a collection of rows. A table usually has a name, although some tables are temporary and exist only to carry out a command. All the rows in a table have the same shape (in other words, every row in a table contains the same set of columns). In other database systems, you may see the terms relation, file, or even class. These are all equivalent to a table.

- **Column (field, attribute)**

A column is the smallest unit of storage in a relational database. A column represents one piece of information about an object. Every column has a name and a data type. Columns are grouped into rows, and rows are grouped into tables. In Figure 1.1, the shaded area depicts a single column.

The terms **field** and **attribute** have similar meanings.

- **Row (record, tuple)**

A row is a collection of column values. Every row in a table has the same shape (in other words, every row is composed of the same set of columns). If you are trying to model a real-world application, a row represents a real-world object. For example, if you are running an auto dealership, you might have a vehicles table. Each row in the vehicles table represents a car (or truck, or motorcycle, and so on). The kinds of information that you store are the same for all vehicles (that is, every car has a color, a vehicle ID, an engine, and so on). In Figure 1.2, the shaded area depicts a row.

You may also see the terms record or tuple. These are equivalent to a row.

- **View**

A view is an alternative way to present a table (or tables). You might think of a view as a "virtual" table. A view is (usually) defined in terms of one or more tables. When you create a view, you are not storing more data; you are instead creating a different way of looking at existing data. A view is a useful way to give a name to a complex query that you may have to use repeatedly.

- **Client/server**

In a client/server product, there are at least two programs involved. One is a client and the other is a server. These programs may exist on the same host or on different hosts that are connected by some sort of network.

- **Client**

A client is an application that makes requests. Before a client application can talk to a server, it must connect to a postmaster (see postmaster) and establish its identity. Client applications provide a user interface and can be written in many languages.

- **Server**

A server is a program that services commands coming from client applications.

- **Postmaster**

Postmaster listens for connection requests coming from a client application. When a connection request arrives, the postmaster creates a new server process in the host operating system.

- **Transaction**

A transaction is a collection of database operations that are treated as a unit. MySQL guarantees that all the operations within a transaction complete or that none of them complete. This is an important property. It ensures that if something goes wrong in the middle of a transaction, changes made before the point of failure will not be reflected in the database. A transaction usually starts with a BEGIN command and ends with a COMMIT or ROLLBACK (see the next entries).

- **Commit**

A commit marks the successful end of a transaction. When you perform a commit, you are telling MySQL that you have completed a unit of operation and that all the changes that you made to the database should become permanent.

- **Rollback**

A rollback marks the unsuccessful end of a transaction. When you roll back a transaction, you are telling MySQL to discard any changes that you have made to the database (since the beginning of the transaction).

- **Index**

An index is a data structure that a database uses to reduce the amount of time it takes to perform certain operations. An index can also be used to ensure that duplicate values don't appear where they aren't wanted.

- **Result set**

When you issue a query to a database, you get back a result set. The result set contains all the rows that satisfy your query. A result set may be empty.

B. Goals of DBMS

Goals of Database Management System are:

(1) To remove problem of data redundancy and inconsistency- Different programmers create files and application programs over a long period and these files are likely to have different format and programs are written in different languages. Same information may be duplicated in various files.

(2) Easy access of data- If a bank officer needs to find out names of all customer who live within a particular postal- code area, the officer then ask the data processing department to generate such a list, because the designers of original database system did not anticipate this request, there is no application program on hand to meet it, there is however an application program to generate the list of all customer.
(3) To solve data isolation problem- As data are scattered in various files, and files may be in various formats, writing new application programs to retrieve the appropriate data is difficult.

(4) To solve atomicity problem- A computer system may fail. In many applications it is crucial that, if a failure occurs the data be restored to the consistent state that existed prior to the failure. E.g. let us having a program to transfer $ 50 from account A to account B. If a system failure occur during the execution of program, it is possible that the $ 50 was removed from account A bit was not clearly it is essential to data consistency that either both the credit and debit occur or that neither occur. That is fund transfer must be atomic- it must happen in it's entirely of not at all, it is difficult to ensure atomicity in a conventional file - processing system.

(5) To solve problem of concurrent- access anomalies- For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.

(6) To solve security problem- Not every user of database system should be able to access all the data. For example in a banking system, the payroll personnel is allowed to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts.

(7) To solve integrity problems- The data values stored in database must satisfy certain consistency constraints.

C. Objectives of DBMS

The objectives of DBMS can be narrated as follows:

1. Eliminate redundant data.
2. Make access to the data easy for the user.
3. Provide for mass storage of relevant data.
4. Protect the data from physical harm and un-authorised systems.
5. Allow for growth in the data base system.
6. Make the latest modifications to the data base available immediately.
7. Allow for multiple users to be active at one time.
8. Provide prompt response to user requests for data.

Let's explain few of them:

- **Mass Storage**

DBMS can store a lot of data in it. So for all the big firms, DBMS is really ideal technology to use. It can store thousands of records in it and one can fetch all that data whenever it is needed.

- **Removes Duplicity**

If you have lots of data then data duplicity will occur for sure at any instance. DBMS guarantee it that there will be no data duplicity among all the records. While storing new records, DBMS makes sure that same data was not inserted before.

- **Multiple Users Access**

No one handles the whole database alone. There are lots of users who are able to access database. So this situation may happen that two or more users are accessing database. They can change whatever they want, at that time DBMS makes it sure that they can work concurrently.

- **Data Protection**

Information such as bank details, employee's salary details and sale purchase details should always be kept secured. Also all the companies need their data secured from unauthorized use. DBMS gives a master level security to their data. No one can alter or modify the information without the privilege of using that data.

- **Data Backup and recovery**

Sometimes database failure occurs so there is no option like one can say that all the data has been lost. There should be a backup of database so that on database failure it can be recovered. DBMS has the ability to backup and recover all the data in database

- **Everyone can work on DBMS**

There is no need to be a master of programming language if you want to work on DBMS. Any accountant who is having less technical knowledge can work on DBMS. All the definitions and descriptions are given in it so that even a non-technical background w=person can work on it.

- **Integrity**

Integrity means your data is authentic and consistent. DBMS has various validity checks that make your data completely accurate and consistence.

- **Platform Independent**

One can run DBMS at any platform. No particular platform is required to work on database management system.

D. Advantages of DBMS

As said above, the database is created, managed and administered using a DatabaseManagement System (DBMS). The role of a DBMS is to:

- Create database;
- Create tables;
- Create supporting structures (e.g indexes);
- Read database data;
- Modify (insert, update, or delete) database data;
- Maintain database structures;
- Enforce rules;
- Control concurrency;
- Provide security;
- Perform backup and recovery
- Integrity
- Data Independence
- Shared Data
- Conflict Resolution
- Reduction of Redundancies

Let's explain few of them:

1. Integrity:
Centralised control can also ensure that adequate checks are incorporated in the DBMS to provide data integrity. Data integrity means that the data contained in the data base is both accurate and consistent. Therefore, Data values being entered for storage could be checked to ensure that they fall within a specified range and are of the correct format.

For example, the value for the age of an employee may be in the range of 16 and 75.

Another integrity check that should be incorporated in the data base is to ensure that if there is a reference to certain object, that object must exist. In the case of an automatic teller machine, for example, a user is not allowed to transfer funds from a non-existent savings account to a checking account.

2. Security:
Data is of vital importance to an organisation and may be confidential. Such confidential data must not be accessed by un-authorised persons. The data base administrator (DBA) who has the ultimate responsibility for the data in the DBMS can ensure that proper access procedures are followed, including proper authentication schemes for access to the DBMS and additional checks before permitting access to sensitive data.

3. Data Independence:

Data independence is usually considered from two points of view; physical data independence and logical data independence. Physical data independence allows changes in the physical storage devices or organisation of the files to be made without requiring changes in the conceptual view or any of the external views and hence in the application programs using the database.

Thus, the files may migrate from one type of physical media to another or the file structure may change without any need for changes in the application programs. Logical data independence implies that application programs need not be changed if fields are added to an existing record; nor do they have to be changed if fields not used by application programs are deleted.

Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schemas. Data independence is advantageous in the data base environment since it allows for changes at one level of the data base without affecting other levels. These changes are absorbed by the mappings between the levels.

4. Shared Data:
A data base allows the sharing of data under its control by any number of application programs or users. In the example discussed earlier, the applications for the public relations and payroll departments could share the data contained for the record type employee.

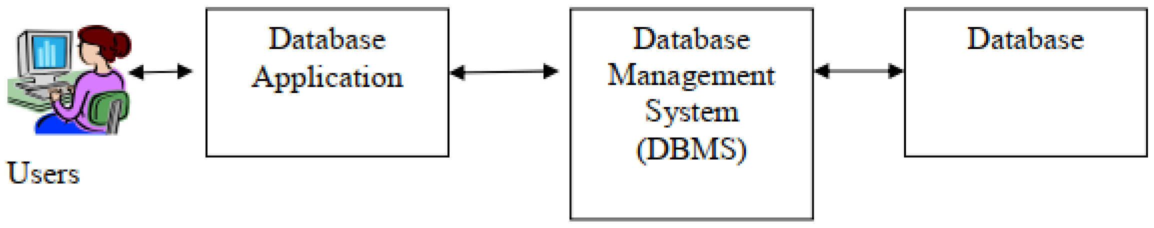
5. Conflict Resolution:
Since the data base is under the control of the data base administrator (DBA), he should resolve the conflicting requirements of various users and applications. In essence, the DBA chooses the best file structure and access method to get optimal performance for the critical applications, while permitting less critical applications to continue to use the database, albeit with the relative response.

6. Reduction of Redundancies:
Centralised control of data by the DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required. It also eliminates the extra processing necessary to trace the required data in a large mass of data.

Another advantage of avoiding duplication is the elimination of the inconsistencies that tend to be present in redundant data files. Any redundancies that exist in the DBMS are controlled and the system ensures that these multiple copies are consistent.

E. Components of DBMS

As shown in the figure below, a database system consists of four components: users, the database application, the database management system and the database.



• **Database:** The database is a collection of tables, relationships and queries which enable to store data with less redundancy. Those data should be used into programs by different users.
• **Database Management System:** The DBMS is a computer program used to create process and administer the database. The DBMS receives requests (queries) encoded in SQL and translates

those queries into actions on the database.
• **Database Application:** The database application or most often the **Application**is a set of one or more computer programs that serves as an intermediary between the user and the database. The application read or modifies database data by sending SQL statements to the DBMS.The application also present data to users in the format of forms and reports. Application can be acquired from software vendors, and they are also frequently written (developed) in-house. The knowledge you will gain from this document you to write your own database applications.
• **Users:** The users employ a database application to keep track of things. They use forms to read, enter and request data. They also produce reports (the real results of a software) to be printed.

F. Application of DBMS type

F.1: MySQL

MySQL: MySQL is open-source DBMS software which can be considered as an alternative to Microsoft SQL Server. MySQL is used primarily for e-commerce websites or social networking sites. Popular sites and companies using MySQL are Adobe, Google, Facebook, Alcatel, Lucent and Zappos.

F.2: Oracle

Oracle: Oracle is easily the best DBMS provider. Oracle is in the business of making and selling DBMS since 1979. It thus becomes one of the most popular DBMS makers across the world. One of the best features of Oracle is that it has now included cloud storage feature in its latest release that is 12c. Now the enterprises and the institutions can store even more data! However, Oracle is quite complicated to use, and it is advisable to invest in some training to know how to get the best out of such DBMS.

Applications of Data base management system in general

(1) Banking- In bank database management system is used for storing customer information, account information, and information about bank transactions.

(2) Airlines-Database management system is used for reservations and schedule information.

(3) Universities- In universities and colleges, database management system are used for having student information, course information and about their results.

(4) Credit Card Transactions- Here database management system is used for purchase of credit cards and generation of monthly statements.

(5) Telecommunication- Record of calls made, generating monthly bill, maintaining balances on prepaid calling cards, and storing information about the communication networks.

(6) Finance- For storing information about holdings, sales, and purchases of financial instruments such s stocks and bonds.

(7) Human resources- For information about employees, salaries, payroll taxes and benefits and for generation of pay checks.

(8) Manufacturing- For information of supply chain, and for tracking production of items in factories, inventories of items in factories, inventories of items in warehouses/stores and orders for items.

(9) Sales - For customers, product, and purchase information.

- Topic2:Description of database organization

Database organization key terms

Term	Definition
Association	Refers to links between the various entities of an application. In an object-oriented database, association is denoted as references between various objects.
Database-management system (DBMS)	a collection of interrelated data and a set of programs to access those data
Database	a collection of data
File-processing system	a system which stores permanent records in various files, and needs different application programs to extract records from, and add records to, the appropriate files
Data inconsistency	various copies of the same data may no longer agree
Consistency constraints	adding boundaries to data fields to enforce them to be within a range of allowable values
Physical data independence	the user of the logical level does not need to be aware how the data is actually stored
Physical level	the lowest level of abstraction which describes how the data are actually stored
Logical level	the middle level of abstraction which describes what data are stored in the database
View level	the collection of information stored in the database at a particular moment
Schema	the overall design of the database
Instance	the collection of information stored in the database at a particular moment
Physical schema	describes the database design at the physical level
Logical schema	describes the database design at the logical level
Subschemas	describe different views of the database
Data model	a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints
Relational model	uses a collection of tables to represent both data and the relationships

	among those datarelationstables in the relational model.
entity-relationship model (E-R)	uses a collection of basic objects, called entities, and relationships among these objects
Object-based data model	combines the object-oriented data model and relational data model; extends the entity-relationship model with notions of encapsulation, methods, and object identity
Semi-structured data model	permits the specification of data where individual data items of the same type may have different sets of attributes
Extensible markup language (XML)	widely used to represent semi-structured data
Network data model	outdated predecessor to the relational data model
Hierarchical data model	outdated predecessor to the relational data model
Data-definition language (DDL)	a language used to specify a database schema by a set of definition
Data-manipulation language (DML)	a language that enables user to access or manipulate data as organized by the appropriate data model
Procedural DMLs	require a user to specify what data are needed and how to get those data
Declarative DMLs	require a user to specify what data are needed without specifying how to get those data (nonprocedural DMLs)
Query	a statement requesting the retrieval of information
Query language	the portion of a DML that involves information retrieval
Data storage and definition	a set of statements in a special type of DDL that specifies the storage structure and access methods used by the database system
Domain constraints	a certain data type can act as a restraint on the values in that domain
Referential integrity	ensuring that a value in one relation for a given set of attributes also appears in a certain set of attributes in another relation
Assertions	any condition that the database must always satisfy
Authorization	differentiate among users to change the type of access they are permitted
Read authorization	allows reading, but not modification, of data
Insert authorization	allows modification, but not deletion, or data
Delete authorization	allows deletion of data
Data dictionary	The output of the DDL stores metadata about the structure of the database, in particular the schema of the database.
Metadata	data about data

Relational database	based on the relational model and uses a collection of tables to represent both data and the relationships among those data
Application programs	programs that are used to interact with the database in this fashion
Database design	the design of the database schema
Conceptual-design phase	laying out a detailed overview of the conceptual schema of the database specification of functional requirementsusers describe the kinds of operations (or transactions) that will be performed on the data
Logical-design phase	the designer maps the high-level conceptual schema onto the implementation data model of the database system that will be used
Physical-design phase	the physical features of the database are specified
Attributes	the description of entities
Relationship	an association among several entities
Entity set	the set of all entities of the same type
Relationship set	the set of all relationships of the same type

- Topic 3: Description of data dictionary

A. Definition

Data Dictionary consists of database metadata. It has records about objects in the database.

What Data Dictionary consists of?

Data Dictionary consists of the following information –

- Name of the tables in the database
- Constraints of a table i.e. keys, relationships, etc.
- Columns of the tables that related to each other
- Owner of the table
- Last accessed information of the object
- Last updated information of the object

During a computerization project (most for wide projects), the team in charge of the software design is not the same with the one in charge of programming.

To enable the team of programmers to be in phase with the LMD produced by the designing team, it's essential to make a data dictionary which will help them to easily implement the database.

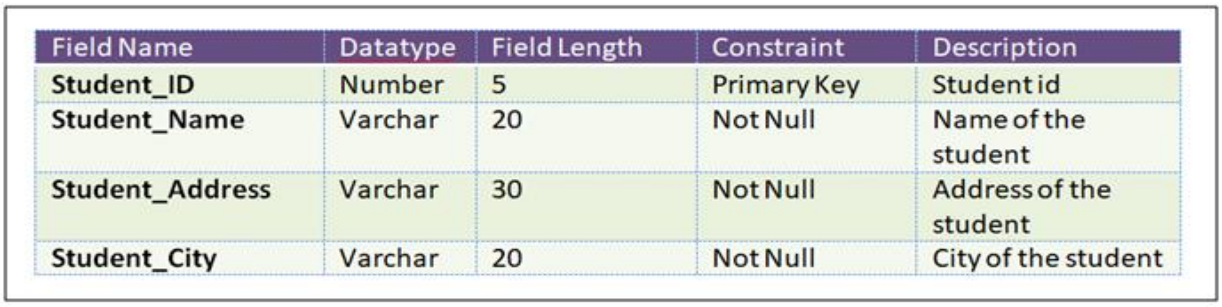
The data dictionary takes the fields of LMD and describes and organizes them into table. These are the examples of data dictionary.

Example1: personal details of a student

<StudentPersonalDetails>

Student_ID **Student_Name** **Student_Address** **Student_City**

The following is the data dictionary for the above fields:



Field Name	Datatype	Field Length	Constraint	Description
Student_ID	Number	5	Primary Key	Student id
Student_Name	Varchar	20	Not Null	Name of the student
Student_Address	Varchar	30	Not Null	Address of the student
Student_City	Varchar	20	Not Null	City of the student

Example 2:

FIELDS	DESCRIPTION	TYPE	SIZE	CONSTRAINT
ClientID	Client's Identifier	Varchar	5	Not Null
CliSurname	Client Surname	Varchar	30	Not Null
CliFirstName	Client First Name	Varchar	40	
ComNum	Command Number	Varchar	10	Not Null
ComDate	Command Date	Date	10	= Day Date [Date()]
BillID	Bill's Identifier	Varchar	10	Not Null
CarID	Car's Identifier	Varchar	15	Not Null
CarPrice	Car Price	Number	10	Not Null

B. Templates

FIELDS	DESCRIPTION	TYPE	SIZE	CONSTRAINT

Learning Unit 3 – Create Database

LO 3.1 – Create tables and attributes

- Topic 1:Selection of DBMS tools

Definition of the abbreviation "Structured Query Language (SQL)"

SQL is Structured Query Language, which is adatabase computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.SQL is widely popular because it canexecute queries against a database, retrieve data, insert records, update records, delete records from a database, create new databases, create new tables in a database, create stored procedures in a database, create views in a database and set permissions on tables, procedures, and views.

SQL Editor

When you interact with MySQL databases, you mostly do it by writing, editing, and executing SQL queries, statements, stored procedures, and scripts. A good SQL editor will help with database interaction by providing syntax highlighting, robust code completion functionality, the ability to get information about function parameters, and other features that make your coding experience more efficient.

Note: We are going to use **MySQL** as a Database Management System (**DBMS**).

Download XAMPP for Windows on this websitehttp://www.apachefriends.org/en/xampp-windows.html and use MySQL SQL Editor which contains MariaDB. There are many other SQL editors that you can use like MySQL workbench, dbForge, Studio, DBeaver.

Using XAMPP on Your Own Computer

The XAMPP package is free and is preconfigured so that the components will talk to each other. This eliminates
the hassle of the usual practice of downloading several individual components and then configuring them to work together.

Download and Install XAMPP
XAMP is free and needs no configuring. To download the package, go to:
http://www.apachefriends.org/en/xampp-windows.html
After installing, If XAMPP is running, you will see an icon in the Notification area like the one shown in Figure below

Figure: *The XAMPP icon*

Create a shortcut on your Desktop for XAMPP

***Figure:** Time-saving shortcuts*

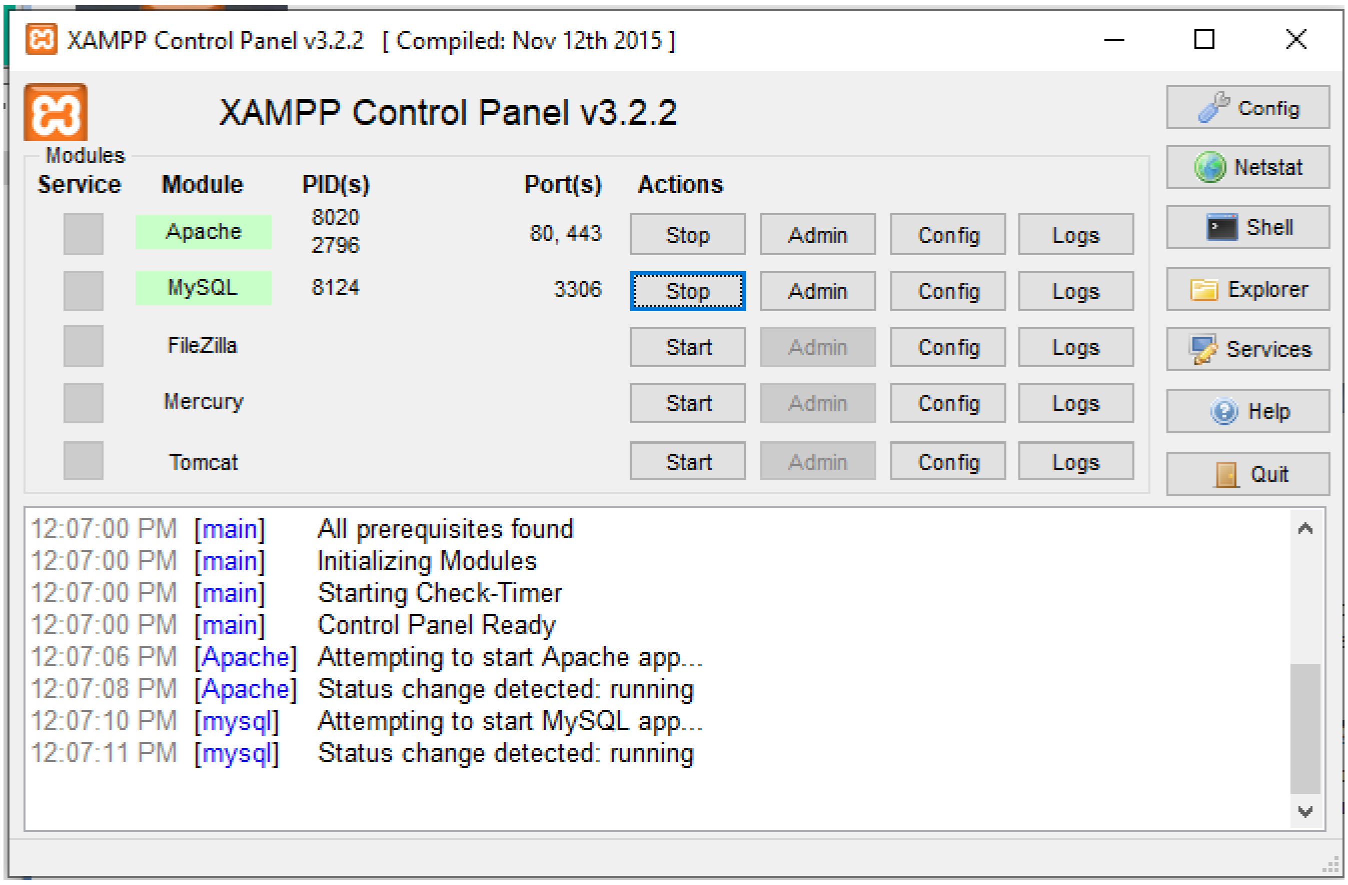
If a desktop icon was not created during the installation, I recommend that you go to the C:\xampp folder, andthen create a Desktop shortcut for the xampp-*control.exe* file.

One common problem is that Skype uses the same port as Apache. So if users have Skype running, Apache won'tstart. You can change the ports in Skype's advanced options screen. If you have web deployment Agent Services running, you will have to stop that to enable Apache to run.

Starting XAMPP

From here onward, to test your pages in XAMPP, double-click the desktop icon and check that Apache and MySQLhave started. If they have not started, click the start buttons for each and then minimize the control panel.

The XAMPP control panel is shown in Figure


XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Config
Netstat
Shell
Explorer
Services
Help
Quit
Modules

Service	Module	PID(s)	Port(s)	Actions			
	Apache	8020 2796	80, 443	Stop	Admin	Config	Logs
	MySQL	8124	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
12:07:00 PM  [main]     All prerequisites found
12:07:00 PM  [main]     Initializing Modules
12:07:00 PM  [main]     Starting Check-Timer
12:07:00 PM  [main]     Control Panel Ready
12:07:06 PM  [Apache]   Attempting to start Apache app...
12:07:08 PM  [Apache]   Status change detected: running
12:07:10 PM  [mysql]    Attempting to start MySQL app...
12:07:11 PM  [mysql]    Status change detected: running
```

***Figure:** The XAMPP control panel*

Note that, under Service, I have shown that the first two modules are running as services, as indicated by the green color. This ensures that those modules will run as soon as you start XAMPP.

Always minimize the control panel so that you have a clear desktop for starting work on your databases. After starting Apache and MySQL, you can test your installation and examine all of the XAMPP examples and tools; to do this, enter the following address in your browser. http://localhost/ or http://127.0.0.1/

Accessing phpMyAdmin Using XAMPP

Using the address field of any browser, enter the following URL: http://localhost/phpmyadmin/ or http://127.0.01/phpmyadmin/

Be sure to include the http://; otherwise, a browser like Chrome will treat it as a search string.

Note the sections describing the use of phpMyadmin apply to any of the development platforms: XaMpp, WaMpserver, and easy php.

Note that open source programs are continually being improved and upgraded, and you may find that you have a newer version of phpMyAdmin in your XAMPP package than that used in this note. You may also see upgrade messages alerting you to a new version in the phpMyAdmin main window. Where personal data is concerned, security is paramount, so these incremental updates are a good thing for you, though they do mean that some of the screenshots in the book no longer accurately reflect what you see on screen. Don't worry if an interface looks a littledifferent from the ones shown in this note,the usage will normally besimilar.

The Familiar Bits

Within the XAMPP package, the structure of the folders and files will be familiar to Windows users, although their names may not be recognizable.

The XAMPP folders are shown in Figure

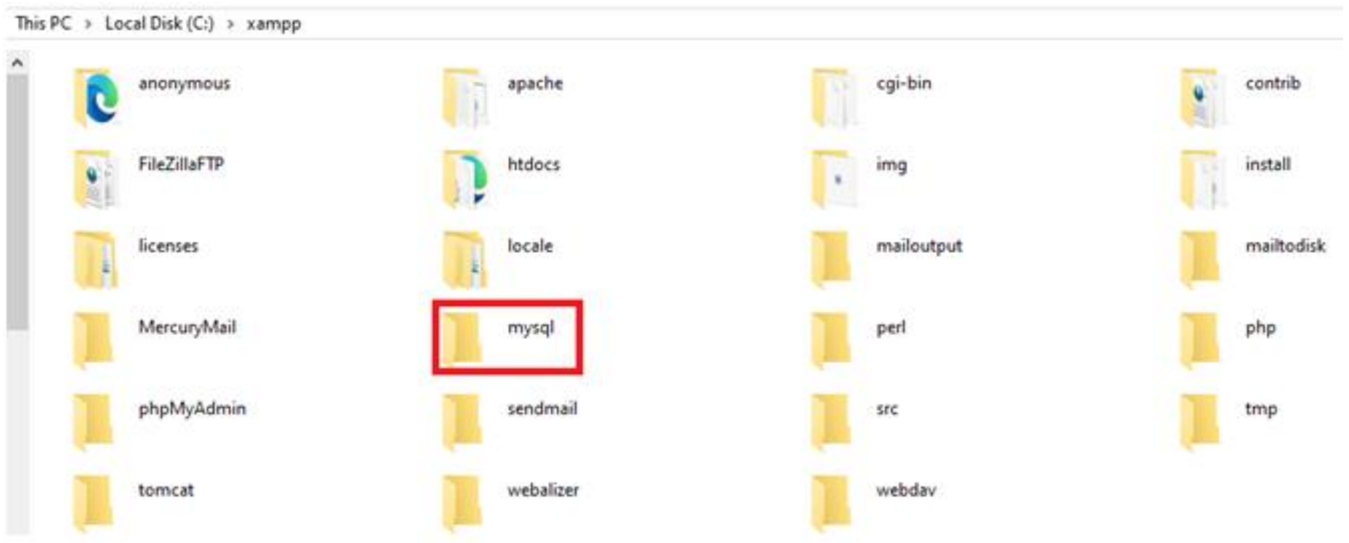


***Figure: XAMPP** folders*

Within the XAMPP folder, you will find a folder called MySQL. This folder contains a folder called data where the databases and tables will reside. Regard a database as a folder; a database must have a unique name. A file within the data folder contains all the information about the database, and it has the file type *.opt.

Tables are files; when you have created any tables, these will also live inside the folder named data and they will have the file type ***.frm**.

Now that you're familiar with the look and feel of the tools you'll be using, you're ready to move ahead. The next section will take you nearer to creating your first database and table.

Create a Database using phpMyAdmin

There is no need to start XAMPP to access phpMyAdmin, although you can if you wish. Note that you will need to have MySQL running, though. If, for some reason, you previously stopped this service, you will need to open up XAMPP to start it again. Open a browser, and then access phpMyAdmin by typing the following in the address field: http://localhost/phpmyadmin
Click the Databases tab in the top menu. You will then see the interface shown in Figure.

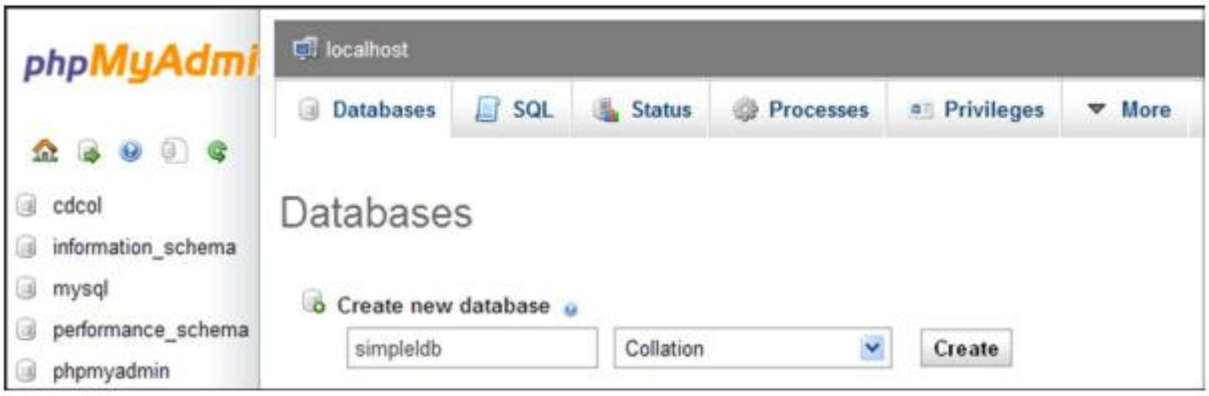
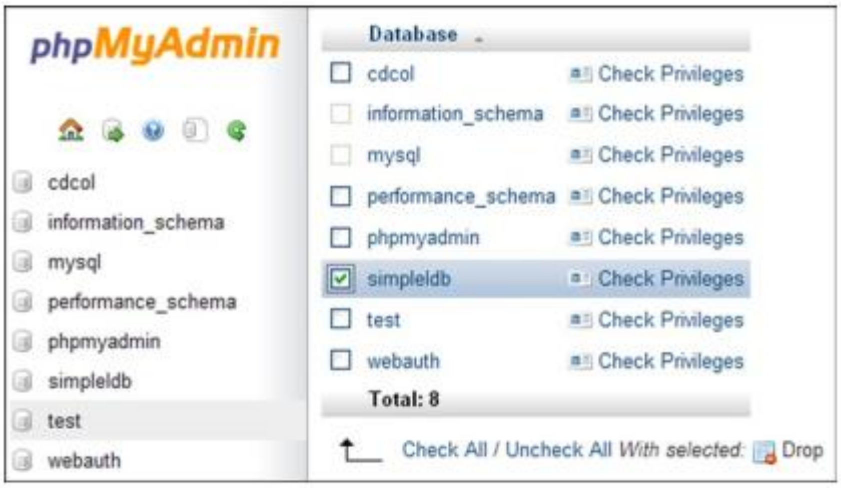


Figure: *The phpMyAdmin interface for creating the database*

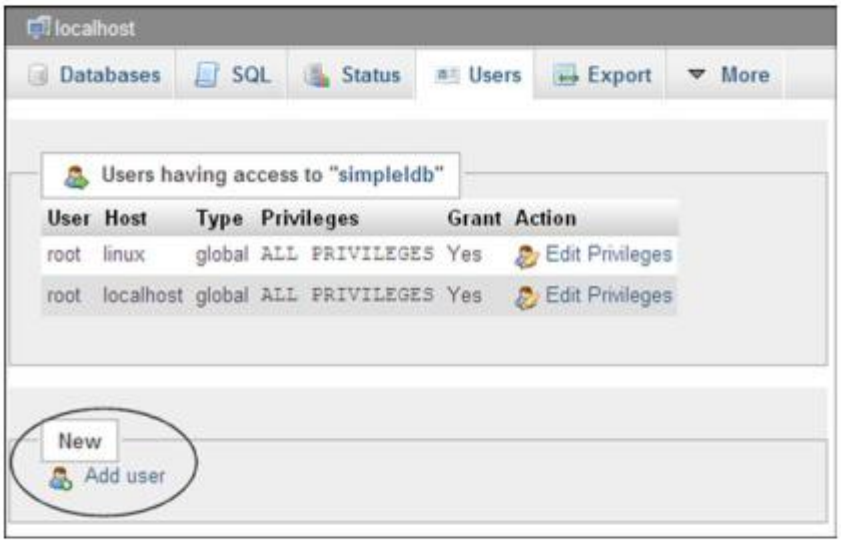
Type a name for the database. For this example, it will be **simpleldb,** all in lowercase except for the uppercase letter I in the last three letters. Then click the Create button (ignore the Collation field). After you click the Create button, the top part of the interface does not change. However, lower down you will see a list of items with check boxes. Figure shows the lower part of the page and a list with check boxes.



	Database	
☐	cdcol	Check Privileges
☐	information_schema	Check Privileges
☐	mysql	Check Privileges
☐	performance_schema	Check Privileges
☐	phpmyadmin	Check Privileges
☑	simpleldb	Check Privileges
☐	test	Check Privileges
☐	webauth	Check Privileges
	Total: 8	

Figure: *In the lower part of the page, select the box next to the name of your new database*

When you select the box next to your new database as shown in Figure, click Check Privileges and you will be taken to a screen where you will see a list of users that have access to the database. To make the database secure, you must add a username and password. Click the words Add User as shown in Figure.

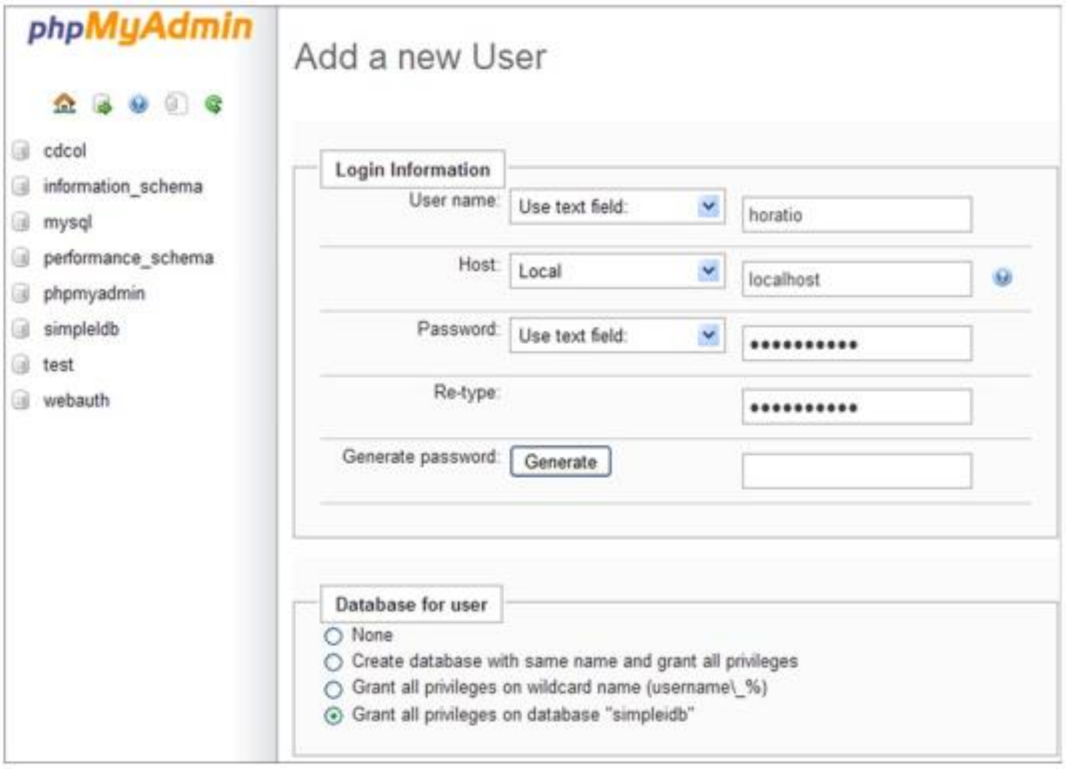


Users having access to "simpleldb"

User	Host	Type	Privileges	Grant	Action
root	linux	global	ALL PRIVILEGES	Yes	Edit Privileges
root	localhost	global	ALL PRIVILEGES	Yes	Edit Privileges

Figure: *The Add user icon is circled in this screenshot*

Clicking Add User will load the *Add a new User* screen, shown next in Figure.



phpMyAdmin

cdcol
information_schema
mysql
performance_schema
phpmyadmin
simpleldb
test
webauth

Add a new User

Login Information

User name:	Use text field:	horatio
Host:	Local	localhost
Password:	Use text field:	●●●●●●●●●●●
Re-type:		●●●●●●●●●●●
Generate password:	Generate	

Database for user

- ○ None
- ○ Create database with same name and grant all privileges
- ○ Grant all privileges on wildcard name (username_%)
- ● Grant all privileges on database "simpleidb"

Figure 1-14: *This screen enables you to add a user and a password*

Caution adding a username and password is absolutely essential; otherwise, your database will be insecure and vulnerable to attack by unscrupulous individuals or their robots. This is the most important habit to cultivate. be sure to record the user and password details in your note book. Keeping a detailed record will save you hours of frustration later.

Using the pull-down menus, accept the default Use text field in the first field and enter the username in the field to right of it. In the second field labelled Host, select local. The word localhost will appear in the field on the right.

Localhost is the default name for the server on your computer. Enter a password in the third field, and confirm yourpassword by retyping it in the lower field. The Generate Password button will create a random strong password if you want something unique.

Scroll down, and where it says *Global privileges (Check All/Uncheck All)*, click Check All.You need to be able to deal with every aspect of the database; therefore, you need all the privileges. If you add other users, you need to restrict their privileges by deselecting boxes such as Drop, Delete, and Shutdown. Scroll down to the bottom of the form, and click the Add User button (or the Go button on some versions). You have now created the database and secured it against attack. The database can be regarded as an empty folder that will eventually contain one or more tables.

Note If you get lost when using phpMyadmin and can't see what you should do next, always click the little house at the top of the left panel. Hover over the icon to ensure that it is the home button.

Now we will create our first table.

- Topic 2: Application of "Create table" command

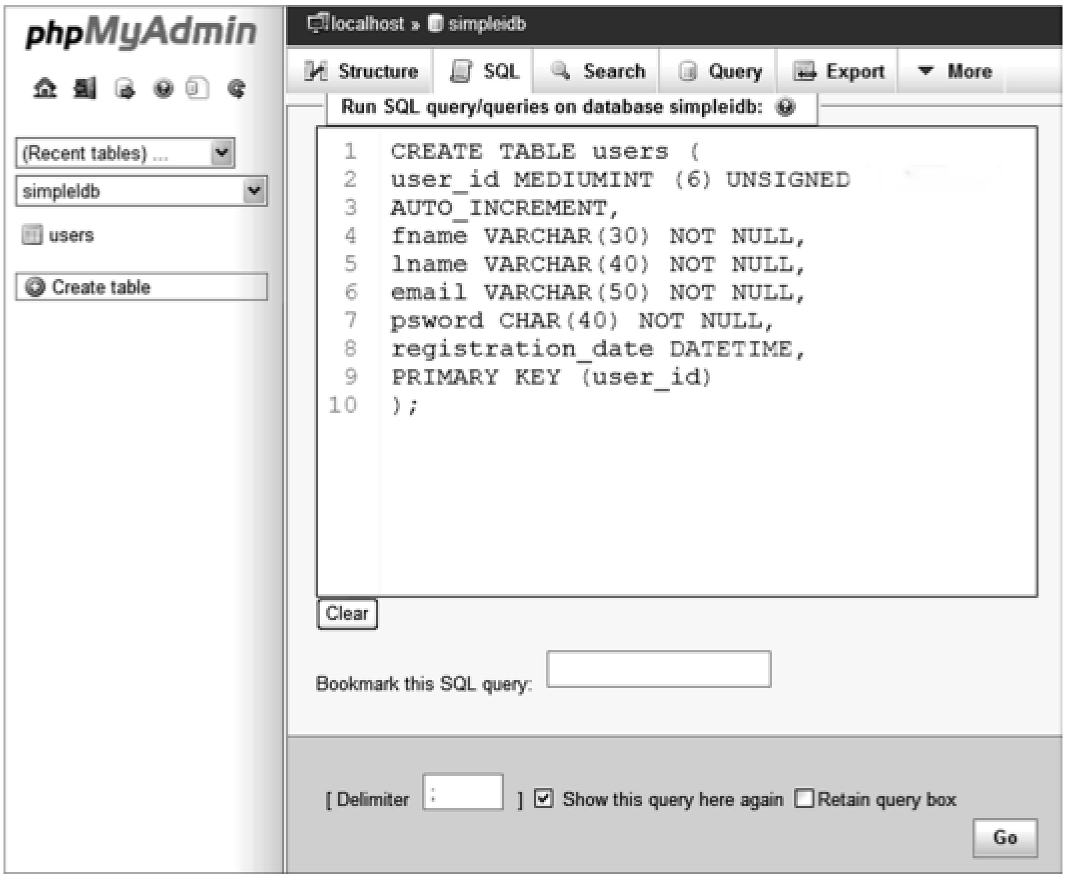
Syntax

```
CREATE TABLE table_name(
column1 datatype,
column2 datatype,
column3 datatype,
.....
columnNdatatype,
PRIMARY KEY( one or more columns )
);
```

Now we will create a table named **users** in the **simpleIdb**database using the SQL window. Click the **simpleIdb** database in the left panel of phpMyAdmin. If the**simpleIdb** database does not show, refresh the page so that it does appear. Open the SQL window, and enter this:

```
CREATE TABLE users (
user_id MEDIUMINT (6) UNSIGNED
AUTO_INCREMENT,
fname VARCHAR(30) NOT NULL,
lname VARCHAR(40) NOT NULL,
email VARCHAR(50) NOT NULL,
psword CHAR(40) NOT NULL,
registration_date DATETIME,
PRIMARY KEY (user_id)
);
```

Figure shows the details entered into the SQL window.



Run SQL query/queries on database simpleidb:

```
 1  CREATE TABLE users (
 2  user_id MEDIUMINT (6) UNSIGNED
 3  AUTO_INCREMENT,
 4  fname VARCHAR(30) NOT NULL,
 5  lname VARCHAR(40) NOT NULL,
 6  email VARCHAR(50) NOT NULL,
 7  psword CHAR(40) NOT NULL,
 8  registration_date DATETIME,
 9  PRIMARY KEY (user_id)
10  );
```

Bookmark this SQL query:

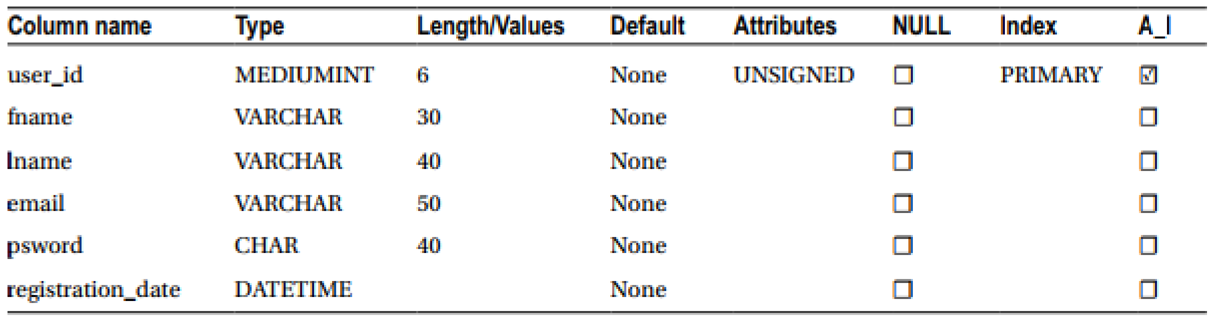
[Delimiter ;] [✓] Show this query here again [ ] Retain query box

***Figure:** Creating a table in the SQL window of phpMyAdmin*

Note that the brackets are all normal brackets, not curly brackets. Press the Enter key after each line, and remember to put the closing bracket and the semicolon at the end of the last line. Each item is separated by a closing comma (lines 3 through 8); if your table has six columns, you should have six commas. Click the Go button, and the table will be created.

Tip: I encourage you to explore the SQL topic just described. The ability to work with SQL will be a very useful alternative. You may wish to refer to the tutorial on: http://dev.mysql.com/doc/refman/5.0/en/tutorial.html.

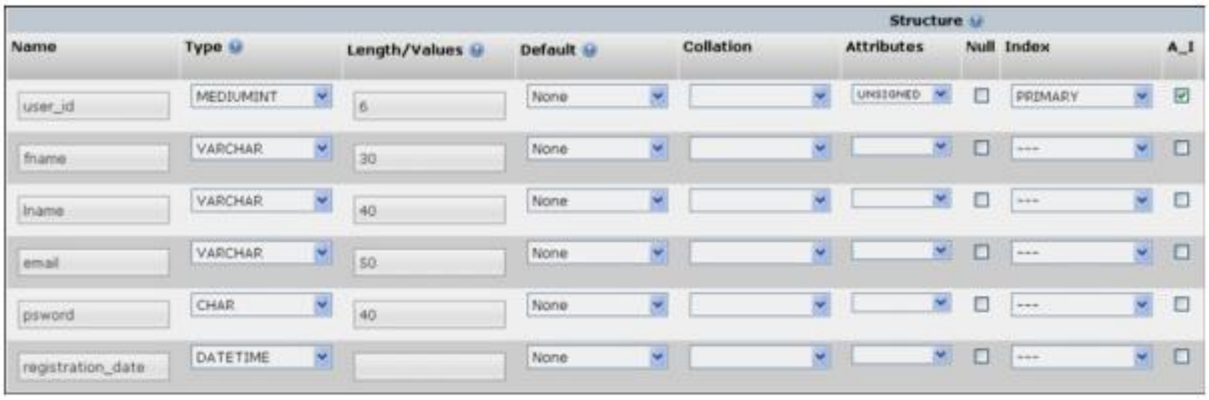
- Topic 3: Application of data type on fields



Column name	Type	Length/Values	Default	Attributes	NULL	Index	A_I
user_id	MEDIUMINT	6	None	UNSIGNED	☐	PRIMARY	☑
fname	VARCHAR	30	None		☐		☐
lname	VARCHAR	40	None		☐		☐
email	VARCHAR	50	None		☐		☐
psword	CHAR	40	None		☐		☐
registration_date	DATETIME		None		☐		☐

***Table:** The attributes for the users table*

Accept all the default settings for each item except for the user_id. Here, you will need to select UNSIGNED, PRIMARY, and the type; also select the A_I box. The various categories under the heading Type will be explained later; the heading Length/Values refers to the maximum number of characters. The Length/Values for the registration_date is left blank because the length is predetermined. Do not enter anything under the headings Default and NULL. The attribute UNSIGNED means that the user_id integer cannot be a negative quantity. The Index for the user_id is the primary index, and A_I means Automatically Increment the id number; as each user is registered to the database, he or she is given a unique number. The number is increased by one as each new user is added. The screen for specifying the attributes is shown in Figure.



					Structure			
Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I
user_id	MEDIUMINT	6	None		UNSIGNED	☐	PRIMARY	☑
fname	VARCHAR	30	None			☐	---	☐
lname	VARCHAR	40	None			☐	---	☐
email	VARCHAR	50	None			☐	---	☐
psword	CHAR	40	None			☐	---	☐
registration_date	DATETIME		None			☐	---	☐

Figure: *This screen allows you to specify column titles and the type of content*

The rows represent columns and they are very wide; you may have to scroll horizontally to enter some of the information. You will find more options as you scroll right, but we will not need them for this tutorial.

So how do you fill out the fields? Enter the six column titles in the fields on the left under the heading Name.

Enter the type of column in the second column of fields under the heading Type. Select them from the Pull-down menus. The types used in this table are as follows:

• **MEDIUMINT** can store integers ranging from minus 8,388,608 to plus 8,388,607. You could choose the next smallest category SMALLINT if the number of users will never exceed 65,535.
•**VARCHAR** specifies a variable-length string of characters from 1 to 255 long.

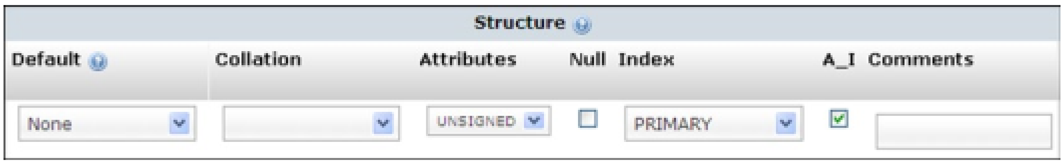
• **CHAR** is a string of characters traditionally used for passwords. Be sure to give this 40 characters so that your database is able to encrypt the password using the function SHA1('$p').

MySQL then converts a password into an encrypted string of 40 characters. A user's password can be, say, 6 to 12 characters long, but it will still be stored in the database as an encrypted40-character string. Incidentally, SHA stands for Secure Hash Algorithm.

•**DATETIME** stores the date and time in the format **YYYY-MM-DD-HH:MM:SS.** Enter the number of characters in the fourth column of fields under the heading Length/Values. Under the heading Default, accept the default *None*. This field is where you can enter a default value if you wish.

Now scroll right. Under the heading *Attributes*, use the drop-down menu to select UNSIGNED for user_id. This ensures that the integer range becomes zero to 16,777,215. This is because a negative quantity is not applicable for the user_id.

The next two entries concern only the user_id. Scroll right so that you can see the headings shown in Figure below.



Structure

Default	Collation	Attributes	Null	Index	A_I	Comments
None		UNSIGNED	☐	PRIMARY	✓	

Figure:*Two extra entries for the user_id column*

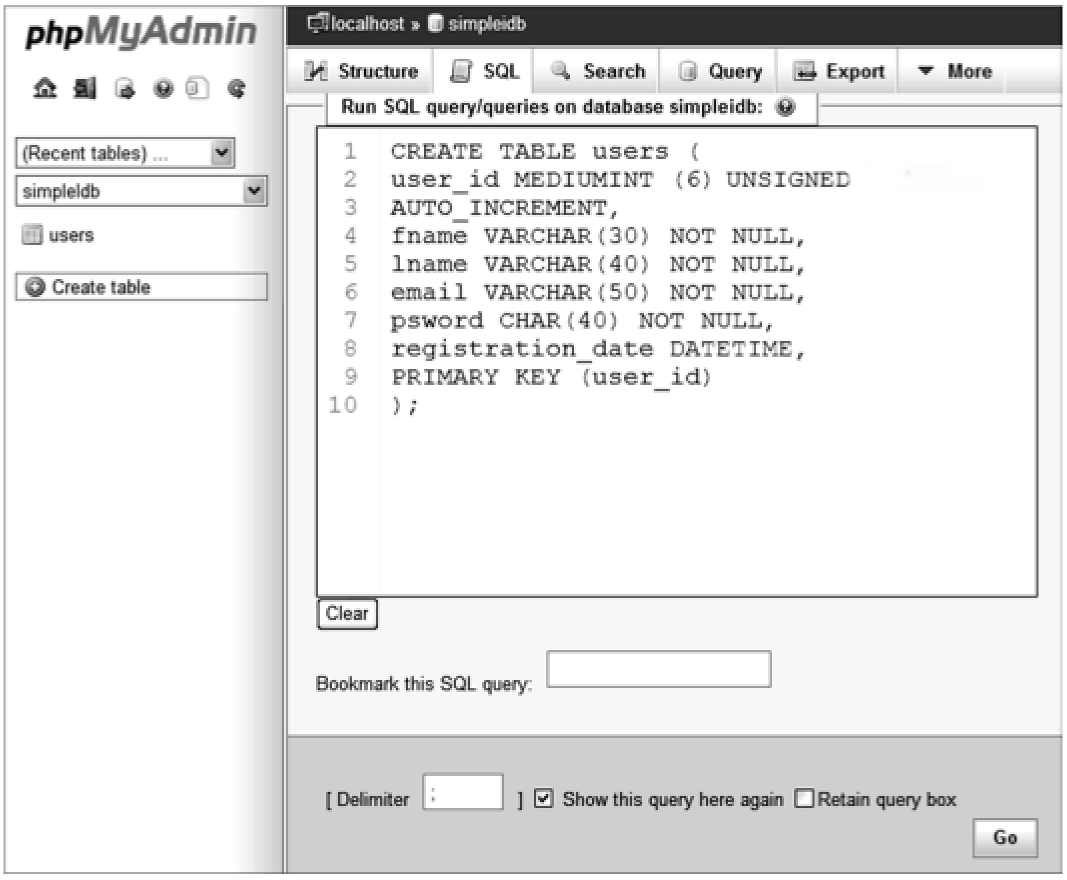
For the user_id, under the heading Index, click the drop-down menu to enter PRIMARY. The user_id should always be a primary index.

Under the heading *A_I*, select the topmost check box so that the user_id number is automatically incremented when each new record is added to the database. Now scroll to the bottom and click the SAVE button.

Caution If you forget to select the a_I box for user_id, you will receive an error message when you later try to enter the second record. the message will say that you are trying to create a duplicate value "0" for id_user.

Some people prefer a blend of GUI and command-line for programming a table, phpMyAdmin allows you to do this by means of SQL. However, this book will mainly use the phpMyAdmin GUI. The SQL alternative is described next. You can skip this section if you wish, but I recommend that you come back to it at some future date because you will undoubtedly come across SQL in other more advanced manuals.

The SQL Alternative



Run SQL query/queries on database simpleidb:

```
CREATE TABLE users (
user_id MEDIUMINT (6) UNSIGNED
AUTO_INCREMENT,
fname VARCHAR(30) NOT NULL,
lname VARCHAR(40) NOT NULL,
email VARCHAR(50) NOT NULL,
psword CHAR(40) NOT NULL,
registration_date DATETIME,
PRIMARY KEY (user_id)
);
```

Bookmark this SQL query:

[Delimiter ;] [✓] Show this query here again [ ] Retain query box

***Figure:**SQL table creation commands*

- Topic3: Modification of tables using commands

Every table should be contained in a database that is why before looking at modification of table we should create database. Let's see some the commands that can be used to create and manipulate the database.

No	Command &description
1	**CREATE:** Creates a new table, a view of a table, or other object in the database.
2	**ALTER:** Modifies an existing database object, such as a table.
3	**DROP:** Deletes an entire table, a view of a table or other objects in the database
4	**RENAME:** Renames a database or a tableby giving another name.
5	**TRUNCATE:** Is used to delete complete data from an existing table.

No	Command	syntax
1	Create a database	**CREATE database database_name;**
2	Rename a database	**Note**: there is no single command to rename a database
3	Drop a database	**DROP database database_name;**

Examples:

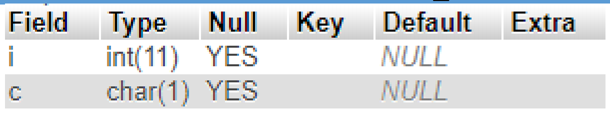
Command	Example
CREATE database	**CREATE database CUSTOMERS**;
DROP database	DROP database**CUSTOMERS**;
RENAME database	USEmaster; GO ALTERDATABASE CUSTOMERS SET SINGLE_USER WITHROLLBACKIMMEDIATE GO ALTERDATABASE CUSTOMERS MODIFYNAME = CLIENTS; GO ALTERDATABASE CLIENTS SET MULTI_USER GO

✓ **Alter**

The MySQL **ALTER** command is very useful when you want to change a name of your table, any table field or if you want to add or delete an existing column in a table.

Let us begin with the creation of a table called **testalter_tbl**.

```
create table testalter_tbl
  (
i INT,
c CHAR(1)
   );
SHOW COLUMNS FROM testalter_tbl;
```



```
Field  Type     Null  Key  Default  Extra
i      int(11)  YES        NULL
c      char(1)  YES        NULL
```

✓ **Dropping, Adding or Repositioning a Column**

If you want to drop an existing column i from the above MySQL table, then you will use the **DROP** clause along with the **ALTER** command as shown below –

Syntax: ALTER TABLE table_name DROP column_name;

```
ALTER TABLE testalter_tbl DROP i;
```

A **DROP** clause will not work if the column is the only one left in the table.

To add a column, use ADD and specify the column definition. The following statement restores the **i** column to the testalter_tbl–

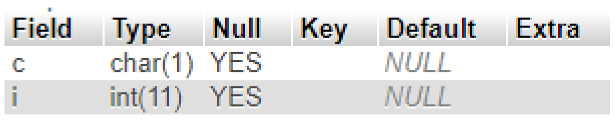
Syntax:

ALTER TABLE table_name ADD column_namedatatype;

```
ALTER TABLE testalter_tbl ADD i INT;
```

After issuing this statement, testalter will contain the same two columns that it had when you first created the table, but will not have the same structure. This is because there are new columns that are added to the end of the table by default. So even though **i** originally was the first column in mytbl, now it is the last one.

```
SHOW COLUMNS FROM testalter_tbl;
```



Field	Type	Null	Key	Default	Extra
c	char(1)	YES		*NULL*	
i	int(11)	YES		*NULL*	

To indicate that you want a column at a specific position within the table, either use FIRST to make it the first column or **AFTER col_name** to indicate that the new column should be placed after the col_name.

Try the following **ALTER TABLE** statements, using **SHOW COLUMNS** after each one to see what effect each one has –

```
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT FIRST;
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

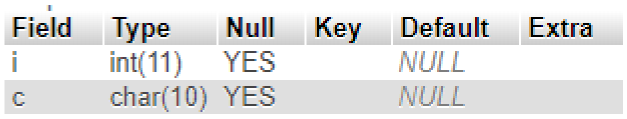
The FIRST and AFTER specifiers work only with the ADD clause. This means that if you want to reposition an existing column within a table, you first must **DROP** it and then **ADD** it at the new position.

- ✓ **Altering (Changing) a Column Definition or a Name**

To change a column's definition, use**MODIFY** or **CHANGE** clause along with the ALTER command.

For example, to change column **c** from CHAR (1) to CHAR (10), you can use the following command:

```
ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```



Field	Type	Null	Key	Default	Extra
i	int(11)	YES		*NULL*	
c	char(10)	YES		*NULL*	

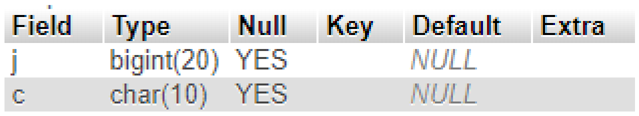
With **CHANGE**, the syntax is a bit different. After the CHANGE keyword, you name the column you want to change, and then specify the new definition, which includes the new name.

Try out the following example –

```
ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

If you now use CHANGE to convert **j** from **BIGINT** back to **INT** without changing the column name, the statement will be as shown below –

```
ALTER TABLE testalter_tbl CHANGE j j INT;
```



Field	Type	Null	Key	Default	Extra
j	bigint(20)	YES		NULL	
c	char(10)	YES		NULL	

The Effect of ALTER TABLE on Null and Default Value Attributes – When you MODIFY or CHANGE a column, you can also specify whether or not the column can contain NULL values and what its default value is. In fact, if you don't do this, MySQL automatically assigns values for these attributes.

The following code block is an example, where the **NOT NULL** column will have the value as 100 by default.

```
ALTER TABLE testalter_tbl

MODIFY j BIGINT NOT NULL DEFAULT 100;
```

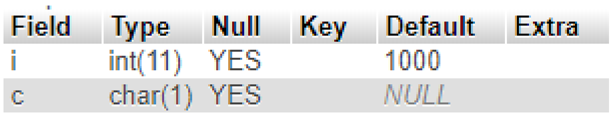
If you don't use the above command, then MySQL will fill up NULL values in all the columns.

- ✓ **Altering (Changing) a Column's Default Value**

You can change a default value for any column by using the **ALTER** command.

Try out the following example.

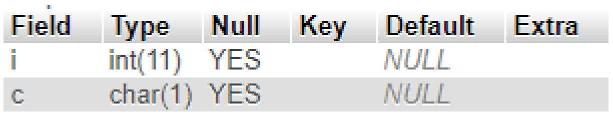
```
ALTER TABLE testalter_tbl ALTER i SET DEFAULT 1000;
SHOW COLUMNS FROM testalter_tbl;
```



Field	Type	Null	Key	Default	Extra
i	int(11)	YES		1000	
c	char(1)	YES		NULL	

You can remove the default constraint from any column by using DROP clause along with the **ALTER** command.

```
ALTER TABLE testalter_tbl ALTER i DROP DEFAULT;
SHOW COLUMNS FROM testalter_tbl;
```



Field	Type	Null	Key	Default	Extra
i	int(11)	YES		NULL	
c	char(1)	YES		NULL	

- ✓ **Renaming (Altering) a Table**

To rename a table, use the **RENAME** option of the **ALTER TABLE** statement.

Syntax:

ALTER TABLE old table_name RENAME TO new table_name.

Try out the following example to rename **testalter_tbl** to **alter_tbl**.

```
ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

You can use the ALTER command to create and drop the INDEX command on a MySQL file.

- ✓ **The TRUNCATE TABLE Statement**

What if we only want to delete the data inside the table, and not the table itself?

Then, use the TRUNCATE TABLE statement:

```
TRUNCATE TABLE table_name;
```

Example /try it by yourself to see the result

```
TRUNCATETABLECUSTOMER;
```

You can also use DROP TABLE command to delete complete table but itwould remove complete table structure form the database and you wouldneed to re-create this table once again if you wish you store some data.

SQL - ALTER TABLE Command with constraints

The SQL **ALTER TABLE** command is used to add, delete or modify columnsin an existing table. You should also use the ALTER TABLE command to addand drop various constraints on an existing table.

Below is table of some syntax for SQL **ALTER TABLE:**

Constraint	**ADD**	**DROP**
Primary key constraint	ALTER TABLE table_name ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);	ALTER TABLE table_name DROP CONSTRAINT MyPrimaryKey; Or in MySQL ALTER TABLE table_name DROP PRIMARY KEY;
Unique key constraint	ALTER TABLE table_name ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);	ALTERTABLEtablename DROPCONSTRAINTUC_tablename;
Not null constraint	ALTER TABLE table_name MODIFY column_namedatatype NOT NULL;	
Check constraint	ALTER TABLE table_name ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);	ALTERTABLEtable_name DROPCONSTRAINTCHK_Columnname;
column	ALTER TABLE table_name ADD column_namedatatype;	ALTER TABLE table_name DROP COLUMN column_name;
Data type	ALTER TABLE table_name MODIFY COLUMN column_namedatatype;	NOT APPLICABLE

LO 3.2 – Create views

- Topic 1: Creation of views

Design Views

Means of accessing a subset of database as if it were a table, the view may be:

- Restricted to named columns, change column names, derive new columns, give access to a combination of related tables.
- Evaluate table de-normalization.

SQL - Using Views-

A view is nothing more than a SQL statement that is stored in the databasewith an associated name.

A view is actually a composition of a table in theform of a predefined SQL query.A view can contain all rows of a table or select rows from a table. A viewcan be created from one or many tables which depend on the written SQLquery to create a view.

Views, which are a type of virtual tables allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.To create a view, a user must have the appropriate system privilege according to the specific implementation.

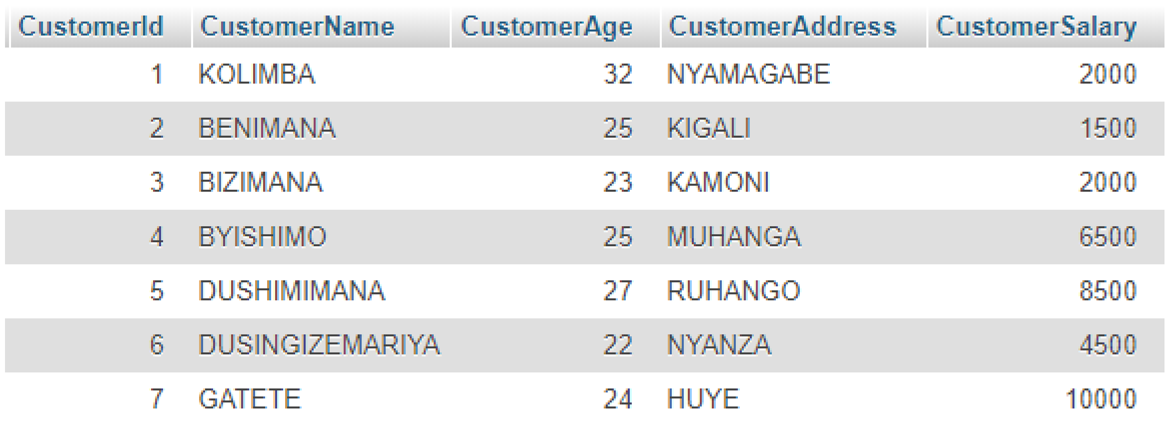
The basic **CREATE VIEW** syntax is as follows –

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in a similar wayas you use them in a normal SQL SELECT query.

Example

Consider the CUSTOMERS table having the following records –



CustomerId	CustomerName	CustomerAge	CustomerAddress	CustomerSalary
1	KOLIMBA	32	NYAMAGABE	2000
2	BENIMANA	25	KIGALI	1500
3	BIZIMANA	23	KAMONI	2000
4	BYISHIMO	25	MUHANGA	6500
5	DUSHIMIMANA	27	RUHANGO	8500
6	DUSINGIZEMARIYA	22	NYANZA	4500
7	GATETE	24	HUYE	10000

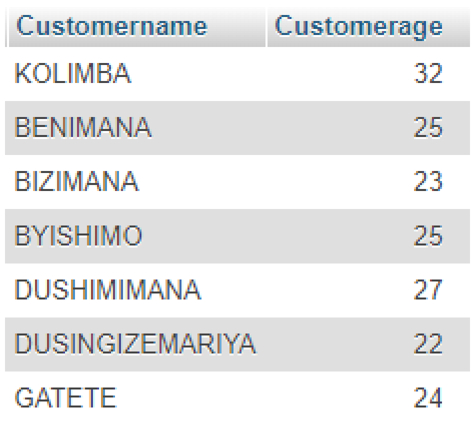
Following is an example to create a view from the CUSTOMERS table. Thisview would be used to have customer name and age from the CUSTOMERS table.

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in a similar way as you query an actual table. Following is an example for the same.

```
SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result.



Customername	Customerage
KOLIMBA	32
BENIMANA	25
BIZIMANA	23
BYISHIMO	25
DUSHIMIMANA	27
DUSINGIZEMARIYA	22
GATETE	24

- ✓ **Queries**

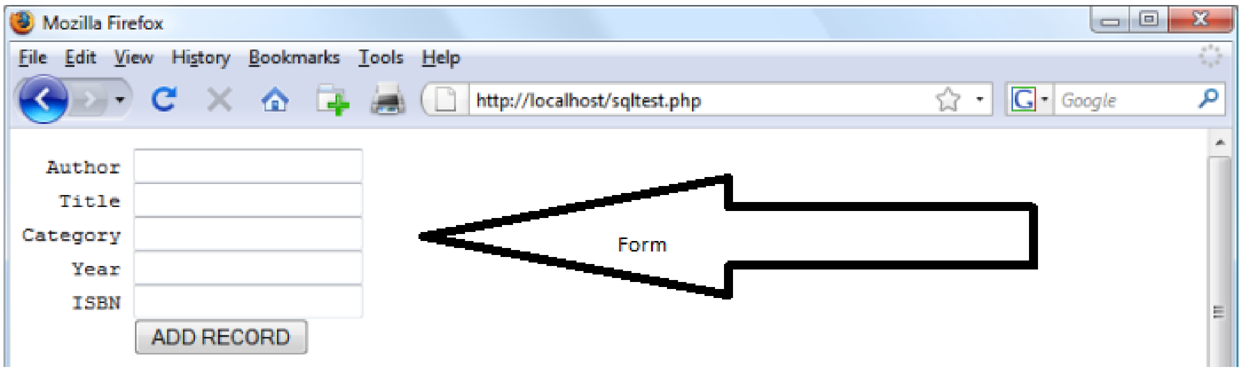
```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

- ✓ **Website forms**

Building forms:

Handling forms is a multipart process. First a form is created, into which a user canenter the required details. This data is then sent to the web server, where it is interpreted,often with some error checking. If the PHP code identifies one or more fields that requirere-entering, the form may be redisplayed with an error message. When the code is satisfied with the accuracy of the input, it takes some action that usually involves the database, such as entering details about a purchase.

Accessing MySQL Using PHP


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/sqltest.php
Google
Form

Author	
Title	
Category	
Year	
ISBN	
	ADD RECORD

It first checks for any inputs that may have been made and then inserts new data into the classics table of the publications database row from it, according to the input supplied. Regardless of whether there was input, the program then outputs all the rows in the table to the browser.

LO 3.3 – Proper management of access control of database

- Topic 1:Types of database users

What are the different types of database users?

1) Application Programmer:

A technical person who writes database application programs. They are also known as Software Developer or Software Engineer. They write programs by using computer programming languages and development tools.

Languages: Java, COBOL, PASAL, PL/SQL etc are used.

Development tools: Eclipse, JDeveloper, SQL Developer etc are used.

Application programs that interact with database are developed as per the user requirements and include functionalities of creating, reading, updating and deleting data from the database. These are commonly known as CRUD operations.

2) Naive Users:

Naive users use application programs written by application programmer for interacting with the database. These users don't have much knowledge about database but thy use predefined applications to access required data from the database.

Clerk working at the LIC office to maintain customer policy data is a naive user.

Student filling information for online university registration is a naive user.

Operator doing a railway reservation is a naive user.

3) Sophisticated users:

These users interact with database by using some query language or by some analytical tool. System Analyst or Data Analysts are sophisticated users. Commonly use SQL and MS Excel for data analysis.

4) Specialized Users:

These are scientist or engineers who can create their own application programs for some specific needs and is different from the programs written by application programmers. They are comfortably familiar with database systems.

5) Database Administrator:

Also known as DBA. They perform the complete administration of database systems.

Database Administrators

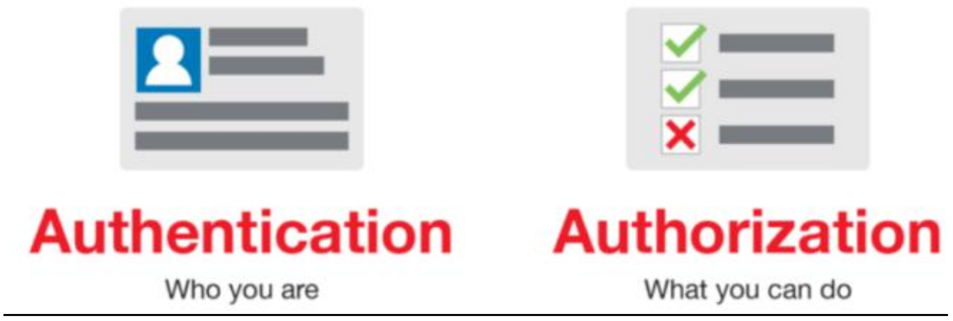
The life cycle of database starts from designing, implementing to administration of it. A database for any kind of requirement needs to be designed perfectly so that it should work without any issues. Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database. The database grows as the data grows in the database. When the database becomes huge, its performance comes down. Also accessing the data from the database becomes challenge. There will be unused memory in database, making the memory inevitably huge. These administration and maintenance of database is taken care by database Administrator – DBA. A DBA has many responsibilities. A good performing database is in the hands of DBA.

- **Installing and upgrading the DBMS Servers: –** DBA is responsible for installing a new DBMS server for the new projects. He is also responsible for upgrading these servers as there are new versions comes in the market or requirement. If there is any failure in upgradation of the existing servers, he should be able revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hot fixes/ patches to the DBMS servers.
- **Design and implementation: –** Designing the database and implementing is also DBA's responsibility. He should be able to decide proper memory management, file organizations, error handling, log maintenance etc for the database.
- **Performance tuning: –** Since database is huge and it will have lots of tables, data, constraints and indices, there will be variations in the performance from time to time. Also, because of some designing issues or data growth, the database will not work as expected. It is responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs works in fraction of seconds.
- **Migrate database servers: –** Sometimes, users using oracle would like to shift to SQL server or Netezza. It is the responsibility of DBA to make sure that migration happens without any failure, and there is no data loss.
- **Backup and Recovery: –** Proper backup and recovery programs needs to be developed by DBA and has to be maintained him. This is one of the main responsibilities of DBA. Data/objects should be

backed up regularly so that if there is any crash, it should be recovered without much effort and data loss.

- **Security: –** DBA is responsible for creating various database users and roles, and giving them different levels of access rights.
- **Documentation: –** DBA should be properly documenting all his activities so that if he quits or any new DBA comes in, he should be able to understand the database without any effort. He should basically maintain all his installation, backup, recovery, security methods. He should keep various reports about database performance.

- Topic 2:User authentication and authorization



Database Authentication

Definition - What does Database Authentication mean?

Database authentication is the process or act of confirming that a user who is attempting to log in to a database is authorized to do so, and is only accorded the rights to perform activities that he or she has been authorized to do.

The concept of authentication is familiar to almost everyone. For example, a mobile phone performs authentication by asking for a PIN. Similarly, a computer authenticates a username by asking for the corresponding password.

In the context of databases, however, authentication acquires one more dimension because it may happen at different levels. It may be performed by the database itself, or the setup may be changed to allow either the operating system, or some other external method, to authenticate users.

For example, while creating a database in Microsoft's SQL Server, a user is required to define whether to use database authentication, operating system authentication, or both (the so-called mixed-mode authentication). Other databases in which security is paramount employ near-foolproof authentication modes like fingerprint recognition and retinal scans.

A. Security issues

Security authentication vs. authorization

Authentication and authorization are often discussed in tandem. However, it’s important to recognize key distinctions between the two. In the login process as a whole, one step can’t be completed without the other. Still, authentication must always come first.

How are authentication and authorization different?

The easiest way to understand this relationship is by asking yourself the questions, “Who are you?” and “What are you allowed to do?”
Just like it would be impossible to determine what an individual is allowed to do if you’re not sure who they are, your registration and login systems cannot authorize users if they’ve yet to be authenticated. In other words, authentication identifies users while authorization determines their permissions. While they’re entirely different processes, they work together to ensure a seamless login experience for users and neither one can be completed without the other!

How do authentication and authorization work together?

Security authentication and authorization should be incorporated into any website or application, although it’s especially vital for those that process online transactions or personal information.

Since anyone with the “key” can gain access, it’s vital that companies implement a strong authentication strategy to keep unauthorized users from accessing accounts without permission.

Boosting security authentication and authorization

Authentication and authorization keep internal accounts organized and help catch unauthorized activity before it becomes a serious threat. **One of the main steps we recommend to protect against breaches is**

to make sure every account has only the permissions they need. That way, your team can spot any unusual behavior early on and take the necessary steps to correct or shut it down.
Strong security authentication protocols prevent cybercriminals from gaining access to your accounts. Having a secure authentication method will make it more difficult for hackers to crack a users' key and gain access to their information.

B. **Preventions from unauthorized users' access**.

There is couple of ways that you can restrict access to a database:

1. Using LOGON Trigger - but only temporarily (enables before upgrade and then disables it).You can **shutdown IIS on the webservers** so that no connections are made using the application. This is called "Application downtime"
2. Keep database in single user mode using (Note this will be risky as if there is any other connection to the database then you might end up waiting or refused connection.)
   Alterdatabasedatabase_namesetsingle_userwithrollback immediately

Seven (7) Database Security Best Practices
Databases - by definition - contain data, and data such as credit card information is valuable to criminals. That means databases are an attractive target to hackers, and it's why database security is vitally important.

Useful database security best practices that can help keep your databases safe from attackers:

- **Ensure physical database security:**In the traditional sense this means keeping your database server in a secure, locked environment with access controls in place to keep unauthorized people out.
- **Use web application and database firewalls:**Your database server should be protected from database security threats by a firewall, which denies access to traffic by default.
- **Harden your database to the fullest extent possible:** Clearly it's important to ensure that the database you are using is still supported by the vendor or open source project responsible for it, and that you are running the most up-to-date version of the database software with all database security patches installed to remove known vulnerabilities.
- **Encrypt your data:** It is standard procedure in many organizations to encrypt stored data, but it's important to ensure that backup data is also encrypted and stored separately from the decryption keys. (Not, for example, stored in encrypted form but alongside the keys in plaintext.) As well as encrypting data at rest, it's also important to ensure confidential data is encrypted in motion over your network to protect against database security threats.
- **Minimize value of databases:**Attackers can only get their hands on what is stored in a database, so ensure that you are not storing any confidential information that doesn't need to be there.

- **Manage database access tightly:** You should aim for the least number of people possible to have access to the database. Administrators should have only the bare minimum privileges they need to do their job, and only during periods while they need access by **using grant and revoke.**
- **Audit and monitor database activity:** This includes monitoring logins (and attempted logins) to the operating system and database and reviewing logs regularly to detect anomalous activity.

- Topic3:Various access methods

Five(5) kinds of data structure and 16 kinds of data access method

So I'm going to explain a bit about *data access methods,* as well as the sub-topic of *data structures.* First, let's define the concept of *data access method* in three steps:

1. The basic mission of a database management system is to, upon request, return part of the database to the requester. In a relational database, what's returned will be a record or set of records; in an object-oriented database it will be an object or set of objects; and so on.
2. In a few major brands of data warehouse appliance, the DBMS may call pretty much the whole database into memory, then find what it wants from there. Most other DBMS are more finicky, and try to retrieve from disk only the data requested. More precisely, they usually retrieve data blocks of a fixed size, and try to bring back only the blocks that may contain the information being looked for.
3. How they make these selections is their *data access method.* More precisely, it's the part of the data access method that deals with getting data off of disk. Often, there will be other parts that govern what happens to the data once it's in memory.

At their core, most data access methods consist of two things:

- A *data structure*.
- (In some cases) A design for *indexes* that point into that data structure.

There are many kinds of data structure, with some of the most important being:

- **Simple files.** A single file can be stored as, well, a file. Or maybe as a BLOB or CLOB (Binary/Character Large object) managed by a DBMS. This used to be the way all data was managed, but now it's mainly used in three contexts:
  - For surviving old-style legacy applications.
  - For single documents, images, videos, sound files, and the like.
  - To hold something whose contents will be indexed by a DBMS or similar technology.
- **Arrays with fixed-length entries.** There's nothing faster than getting data from a completely regular array. Just calculate the address and you're there. But lack of compression is a huge issue, unless you're dealing with purely numeric data that can't be compressed anyway.
- **Tables with variable-length entries.** The bread-and-butter data structure of modern database management is the table, storing numeric, date, or character-string data. By default, entries take up however many bytes are needed to express them, but various kinds of compression are increasingly used on top of that.
- **Linked list (hierarchy or network).** Before there were relational or other index-based database management systems, there were linked-list products. Each data value would be accompanied by a pointer to the next one you were likely to want. Specifically, you were linked to parents,

children, and next-records. These systems were called *hierarchical* if there only was one parent (e.g., IBM's IMS and it's cousins), or *network* if there could be more (e.g., Cullinet's IDMS). The CODASYL standard covered network DBMS, but people soon stopped caring, as index-based DBMS obsoleted linked-list ones.

- "**It's all in the index.**" In some cases, data is wholly copied into the index used to find it, and the underlying data isn't really managed at all. This is particularly common in text search, data federation, and columnar RDBMS.

We have a huge number of possible data access methods, and each major kind can have a broad variety of tweaks.

1. Simple file manager.
2. Classic relational b-tree.
3. Hash index.
4. Bitmap.
5. Other columnar.
6. Star.
7. Geospatial r-tree.
8. Hierarchy-of-arrays MOLAP.
9. Inverted index.
10. Classic full-text index.
11. Classic linked-list.
12. Classic object-oriented.
13. Native XML (all).
14. Tree.
15. File plus calculated metadata (e.g., for still image).
16. Simple key/value lookup.

References:

1. Prettyman, A. W. (2018). *Practical PHP7,MySQL 8, and MariaDB Website Database.* Key West, Florida,USA: Appress Media LLC.

2. Publication, M. R. (2008, January 31). *Monash Research Publication*. Retrieved July 20, 2020, from www.dbms2.com: http://www.dbms2.com/2008/01/31/5-kinds-of-data-structure-and-16-kinds-of-data-access-method/

3. Rubens, P. (2016, August 23). *7 Database Security Best Practices*. Retrieved July 31, 2020, from esecurityplanet: http://www.esecurityplanet.com

4. Tutorialspoint. (2020, July 31). *MySQL ALTER command*. Retrieved July 32, 2020, from www.tutorialspoint.com: https://www.tutorialspoint.com/mysql/mysql-alter-command.html

5. Architect, E. (2011). *Data Modeling From Conceptual Model to DBMS*. Retrieved July 31, 2020, from sparxsystems: http://www.sparxsystems.com