Data Base introduction for Global Marketing Management students

Author: **José María Escalante Fernández**

- **Unit 1 – Introduction to Databases**
  - **Theory**
  - **Class slides**
- **Unit 2 – Relational Database concepts**
  - **Theory**
  - **Class slides**
- **Unit 3 – Conceptual and Local design**
  - **Theory**
  - **Class slides**
- **Unit 4 – Physical design**
  - **Theory**
  - **Class slides**
- **Unit 5 – Practical database**
  - **Theory**
  - **Class slides**

ISBN: 978-84-09-67484-8

Introduction to Databases

Contents

1 What is data? 1

2 What is a database? 2

3 What is a Data Model? 3

4 What is a Database Management System (DBMS)? 6

4.1 Characteristics and functions of a DBMS 6

5 Relation and Non-Relational Databases 8

6 Why a database? 9

1 What is data?

Data is nothing but information that is collected in various formats such as numbers, text, media format (picture, voice or video), and others. Currently, due to the internet, information technologies associated, social media, IoT (internet of things), etc ... the data sources are quite varied (see Fig. 1) and the amount of data generated is impressive (see Fig. 2).

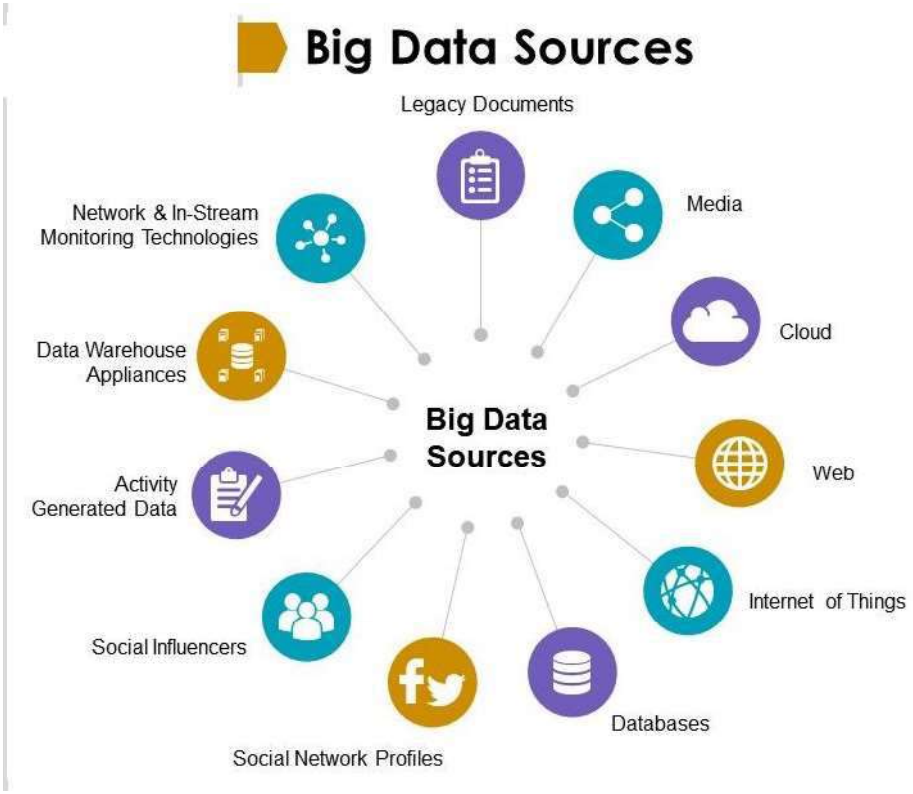
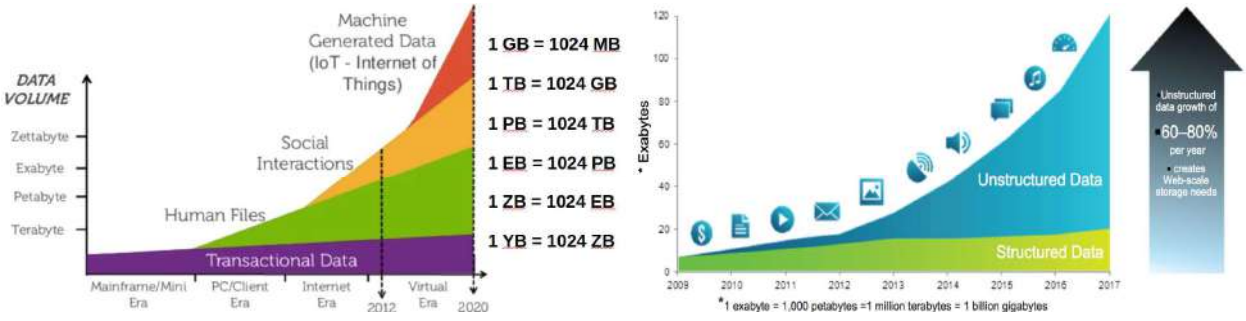


Figure 1: Data sources.



(a) As a function of kind of data.

(b) As a function of structure of data.

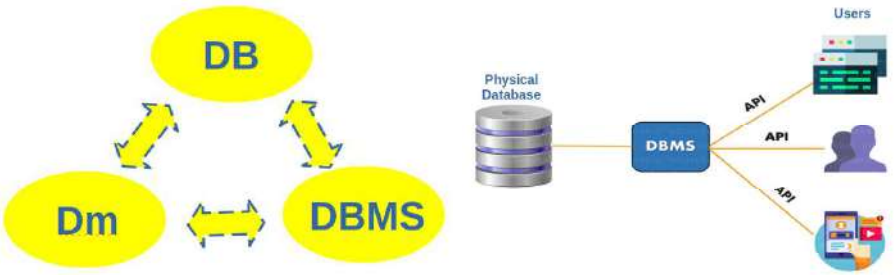
Figure 2: Current amount of data generated.

Currently we are in the era of data where the data is the new petrol. Since the emergence of social media, particularly social networks, a notable increment in the amount of data generated has been observed (Fig. 2a). So, now more than ever, it is crucial a

technology that allows a correct access, storage and management of data. This technology is the technology of databases: **Database** (**DB**; storage system, the physical place where the data is store: servers, hard disk, etc), **Database models** (**Dm**) y **Database Management System** (**DBMS**, software to manage the date and the performance of the database).

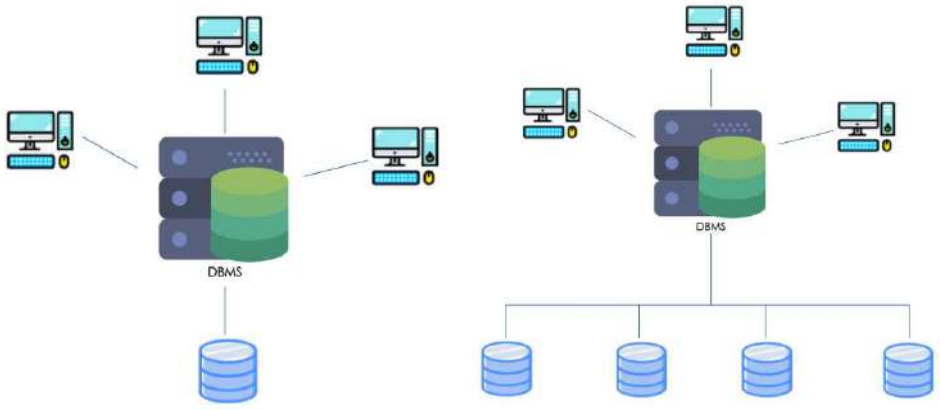
2 What is a database?

A database is a systematic and organised collection of related information that is stored in such a way that it can be easily accessed, retrieved, managed, and updated. A **Database** is based on a **Data model** and managed by a **Database Management System**. So, the DB (physical place where the data is store) and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database. Finally, we can consider **DB**, the **Dm** and the **DBMS** three different aspects of the same thing.



(a) Simple node database scheme. (b) Multiple node database scheme.

Figure 3: a) Relationship among DB, Dm and DBMS. b) **Physical database** indicate where the data is stored physically (hard disks) and **API** indicates the different tools users could use to connect to a DBMS.



(a) Simple node database scheme. (b) Multiple node database scheme.

Figure 4: Different database schemes: a) and b) show a possible schemes with simple or multiple nodes, respectively.

3 What is a Data Model?

A **Data model** or **Data design model** (**Dm**) shows the logical structure of a database, including the relationships and constraints that determine how data can be stored, accessed and related. The data model emphasizes on what kind of data is needed and how it should be organized instead of what operations will be performed on data. Data Model is like an architect's building plan, which helps to build conceptual models and set a relationship between data items. There are different types of data models, each with its pros and cons. Below, the most commonly used are shown:

- **Hierarchical model**: the hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. This data model gives rise to **Hierarchical Databases**.

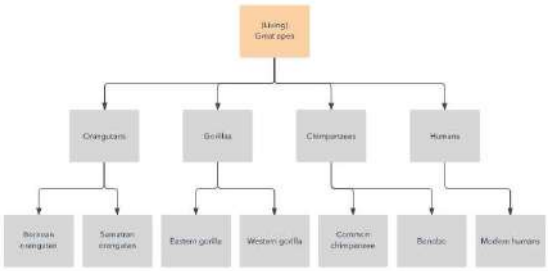


Figure 5: Example of hierarchical model.

- **Network model**

  The network model is based on the hierarchical model, but allowing many-to-many relationships between two adjacent hierarchy levels. So, this model can set up more complex relationships between different kind of records. This data model gives rise to **Network Databases**.

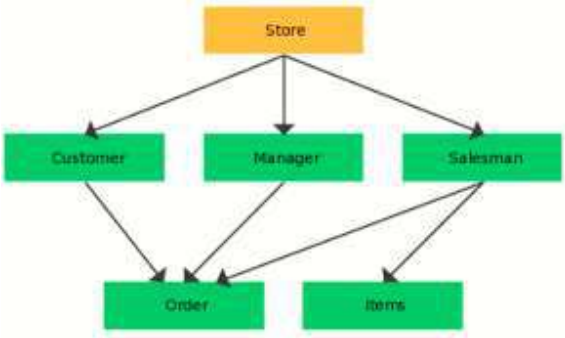
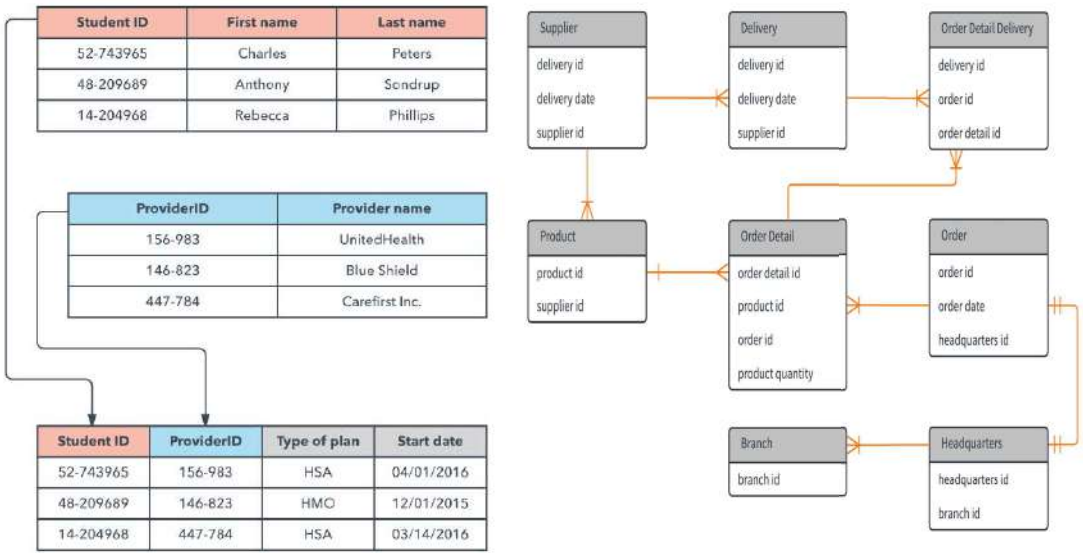


Figure 6: Example of network model.

- **Relational model** The most common model, the relational model stores data into tables, also known as relations, each of which consists of columns and rows. Each column is an attribute of the table. Each row is a set of records. A particular attribute or set of attributes can be chosen to made up a primary key, which help uniquely identify each row in a table and to establish relationships with other tables (foreign key). This data model gives rise to **Relational Databases**.



(a) (b)

Figure 7: a) and b) show two different schemes of tables related in two different relational databases.

- **Object-oriented model**

  This model defines a database as a collection of objects with associated features and methods. There are several kinds of object-oriented databases. The object-oriented database model is also known post-relational database model, since this model incorporates tables, but is not just limited to tables. Such models are also known as hybrid database models. This data model gives rise to **Object-oriented Databases**.

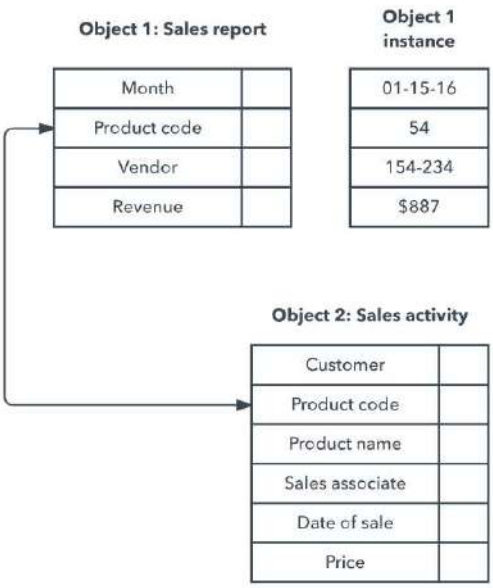


Figure 8: Example of object-oriented model.

- **Non-Relational model**

  This model is mainly used to store data that does not have a fixed structure, that is, the data does not follow a tabular structure, such as text or media format (pictures, videos or voice). This data model gives rise to **Non-Reational Databases**.

  This in turn has different non-relational models:

  – **Document oriented**: the data are stored in documents, such as JSON or XML documents.

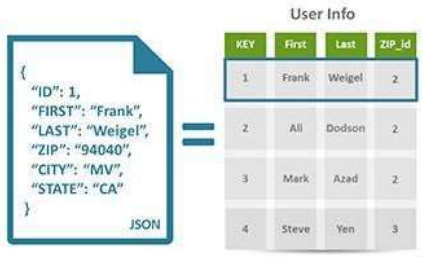


Figure 9: Example of document in a Document Oriented NoSQL Database.

  – **Column oriented**: the data is stores in columns, but unlike a relational database, the names and format of the columns can vary from one to other.

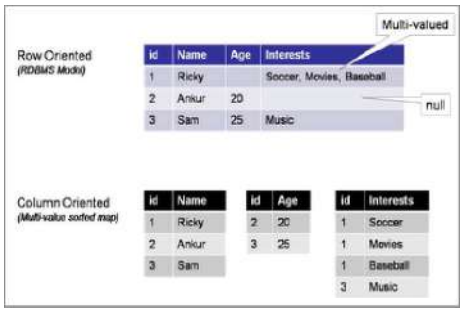


Figure 10: Example of columns in a Column Oriented NoSQL Database.

  – **Key-value oriented**: the data are stored in tuples key-value, the key is used to find the data within database.

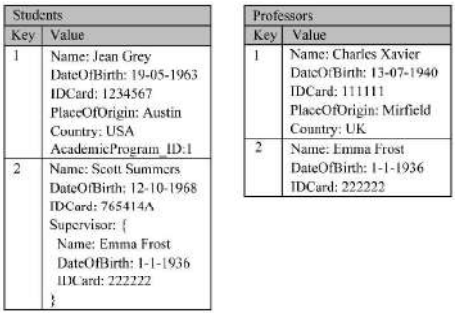


Figure 11: Example of key-value structure in the Key-Value NoSQL Database.

– **Graph**: the graph relates the data items stored to a collection of nodes and edges, the edges representing the relationships between the nodes.

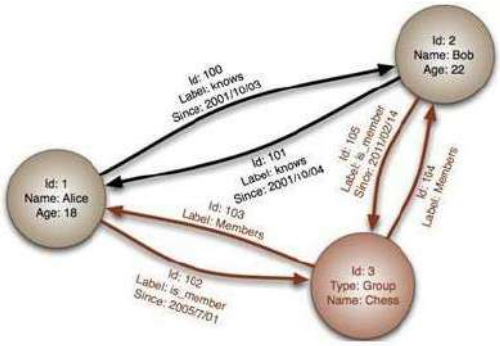


Figure 12: Example of graph structure in the Graph NoSQL Database.

4 What is a Database Management System (DBMS)?

A database typically requires a comprehensive database software program known as a database management system (DBMS). A DBMS serves as an interface between the database and its end users or programs, allowing users to retrieve, update, and manage how the information is organized and optimized. On the other hand, DBMS also facilitates oversight and control of databases, enabling a variety of administrative operations such as performance monitoring, tuning, and backup and recovery.

Deciding on a DBMS always implies choosing a specific database model. In the following table you can see some DBMS examples and its associated data model.

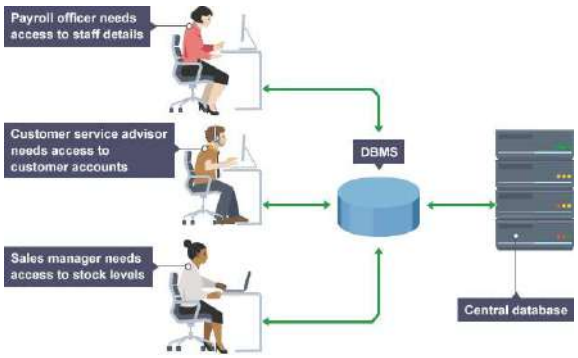


Figure 13: Basic scheme of DBMS connections.

4.1 Characteristics and functions of a DBMS

A DBMS has to meet a set of characteristic and function for a proper operation.

- **Characteristics of a DBMS**
  - **Abstraction**: this characteristic is related to the information stored in the database1. This means that the details about the physical storage of the data is not known, this type of knowledge is transparent to users.

Table 1: Example of database (DDBB)

DDBD Logo	DDBB name	DDBB model
	MySQL	Relational
	Oracle	Relational
	PostgreSQL	Combination of Relational and Object Oriented
	MongoDB	Document oriented
	Cassandra	Key-value
	Neo4j	Graph

  - **Independence**: this consists of the ability to modify the schema (physical or logical) of a database without having to make changes to the applications that use it.
  - **Redundancy**: this refers to storing the same data multiple times in different places (different nodes in a database).
  - **Consistency**: this refers to controlling the information that appears repeated (redundancy) is updated in a coherent way, that is, all the repeated data are updated simultaneously.
  - **Security**: the information stored must be safe against malicious user actions (access, copy or modification of data), for that reason the DBMS have complex systems of control and permissions for users.
  - **Integrity**: guarantee the validity of the information stored against hardware failures.
  - **Concurrency**: simultaneous access to information by different users can cause consistency problems, so DBMS must know how to manage this simultaneous access.

- **Functions of a DBMS**

  The users must perform a set of operations in the database that must be carried out efficiently and safely, for this reason the DBMS provide the following functions:

- **Data definition**: the DBMS must allow to define all object of a database.
- **Data manipulation**: the DBMS must allow to manipulate stored data, allowing: insert, modify, consult and delete the data.
- **Maintenance of permissions and roles**: the DBMS must allow to modify permissions and roles of users to access data.
- **Transaction management**: a transaction is a set of short operations for accessing and modifying parts of the database. So a DBMS must control multiple transactions in parallel in the same database.

5 Relation and Non-Relational Databases

Relational databases typically use **Structured Query Language (SQL)** to access the data, for this reason they are also known as **SQL databases**.

Currently most widespread data models for database design are **RELATIONAL** and **NON-RELATIONA MODEL**. Almost 98% of database market is covered by **RELATIONAL** and **NON-RELATIONA MODEL**

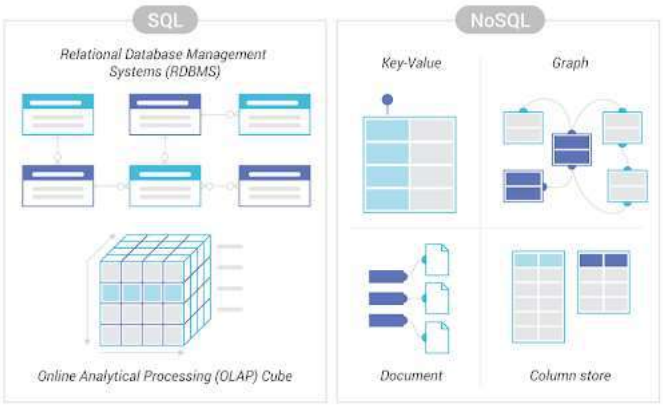


Figure 14: Comparison between SQL and NoSQL databases.

These databases have their pros and cons, arising the question, **what kind of database should I use?** SQL and NoSQL databases solve completely different and mutually exclusive scenarios. Since SQL is ideal for, NoSQL is not and vice versa.

WHEN TO USE SQL?

- **Education**: to structure information, and provide logical knowledge to the student.
- **Web developments**: to maintain data hierarchy, as long as the concurrency, storage and maintenance capacity are of considerable difficulty and the information is consistent.
- **Business**: business intelligence and analysis are topics that require the use of SQL to facilitate the consumption of information and the identification of patterns in the data.

- **Company**: because both custom software and business software have the characteristic of maintaining information with a consistent structure.

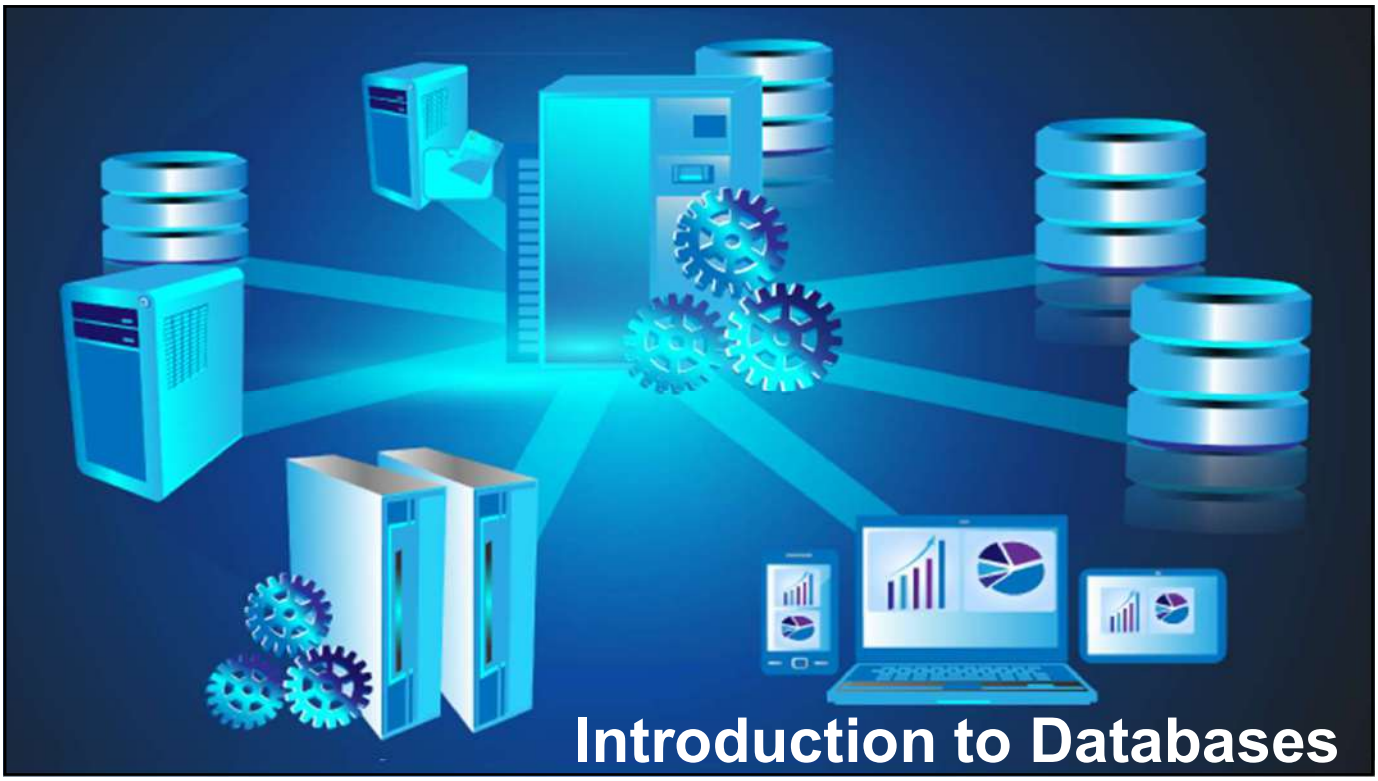
WHEN TO USE NoSQL?

- **Social networks**: almost mandatory.
- **Web development**: due to the lack of uniformity of the information found on the Internet; even though SQL can also be used.
- **Mobile Development**: due to the growing trend of Bring Your Own Device (low computing power).
- **BigData**: due to the administration of huge amounts of information and its evident heterogeneity.
- **Cloud services**: NoSQL can be adapted to almost any customer need, and its peculiarities.

Throughout the course we will focus on SQL databases since they are the ones that best adapt to the business environment where you are going to develop your career.

6 Why a database?

Use a database improves business performance and decision-making. Due to the massive data collection from Transaction, SocialMedia, IoT, etc... transforming life and industry across the globe, businesses today have access to more data than ever before. Forward-thinking organizations can now use databases to go beyond basic data storage and transactions to analyze vast quantities of data from multiple systems. Using database and other computing and business intelligence tools, organizations can now leverage the data they collect to run more efficiently, enable better decision-making, and become more agile and scalable.



1

COURSE CONTENTS BY UNIT

- **Unit 1 - Introduction to Databases**
- What is data?
- What is a database?
- What is a data model?
- What is a database management system (DBMS)?
- Characteristics and functions of a DBMS
- Why a database?
-

2

What is data? - Types

Data is nothing but information that is collected in various formats such as:

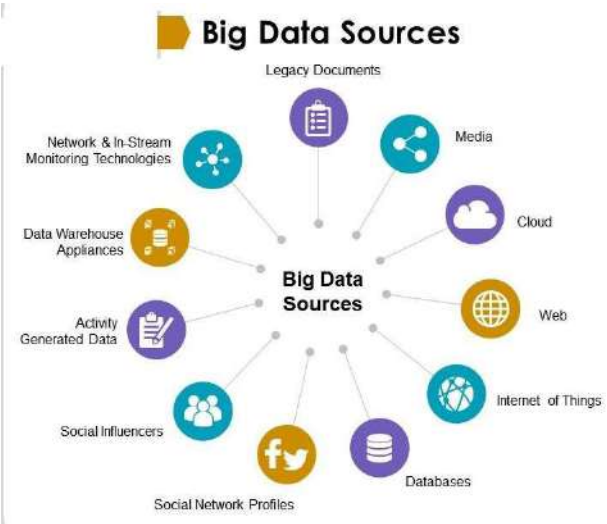
- numbers
- text
- picture
- voice
- video
- measurements
- etc.

3

What is data? - Sources

Currently, due to the internet, information technologies associated, social media, IoT (internet of things), etc ….

The data sources are quite varied.



4

What is data? - Data structure

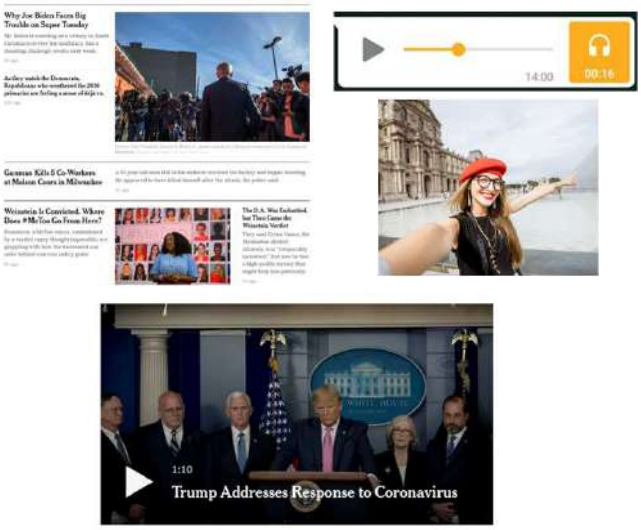
STRUCTURED

	nombre	color	edad	altura	peso	puntuacion
1:	Paco	Rojo	24	182	74.8	83
2:	Juan	Green	30	170	70.1	500
3:	Andres	Amarillo	41	169	60.0	20
4:	Natalia	Green	22	183	75.0	865
5:	Vanesa	Verde	31	178	83.9	221
6:	Miriam	Rojo	35	172	76.2	413
7:	Juan	Amarillo	22	164	68.0	902

SEMISTRUCTURED


```
{
  "marcadores": [
    {
      "latitude": 40.416875,
      "longitude": -3.703308,
      "city": "Madrid",
      "description": "Puerta del Sol"
    },
    {
      "latitude": 40.417438,
      "longitude": -3.693363,
      "city": "Madrid",
    },
    {
      "latitude": 40.407015,
      "city": "Madrid",
      "description": "Estación de
        Atocha"
    }
  ]
}
```

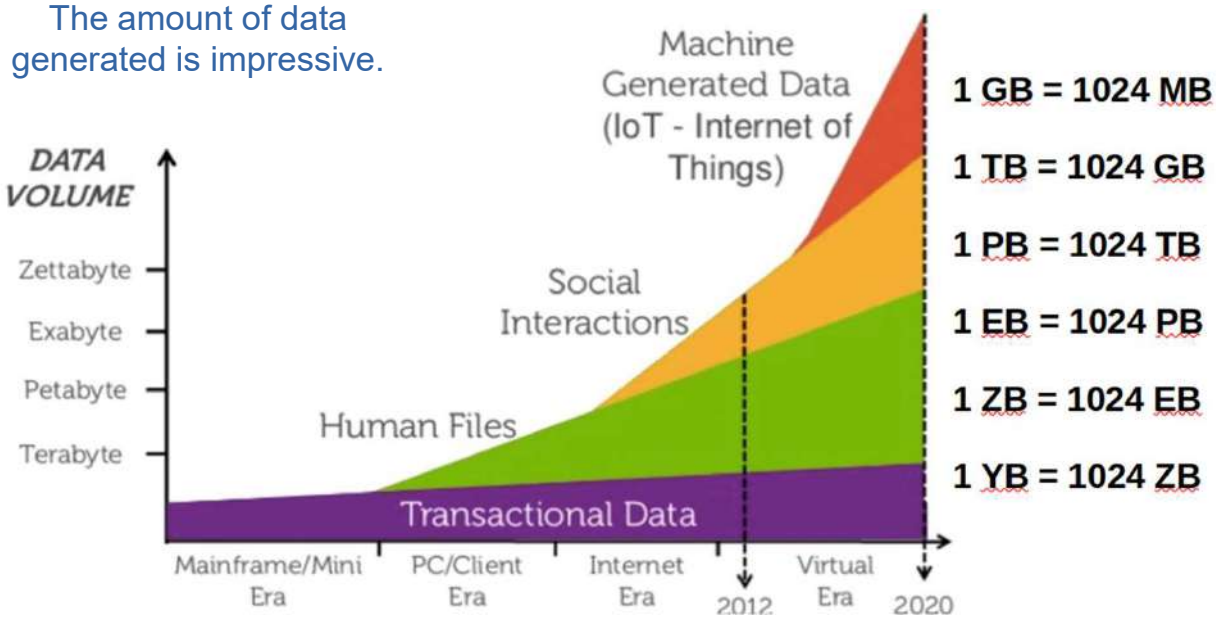

UNSTRUCTURED



5

What is data? - Amount

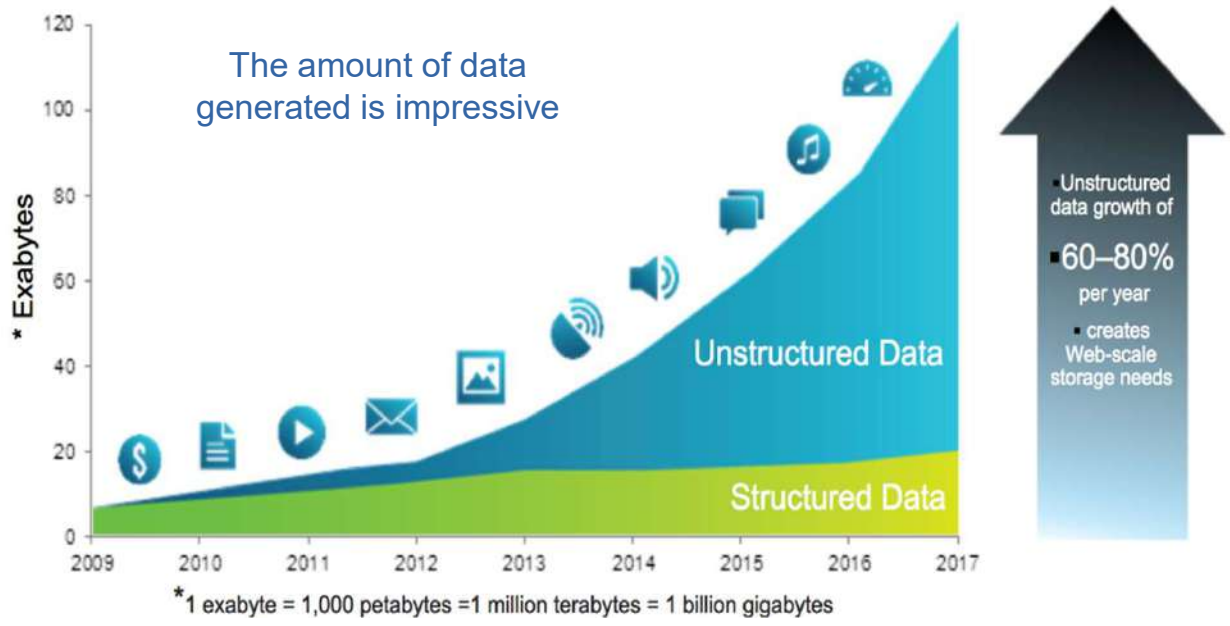
The amount of data generated is impressive.



6

What is data? - Amount

The amount of data generated is impressive

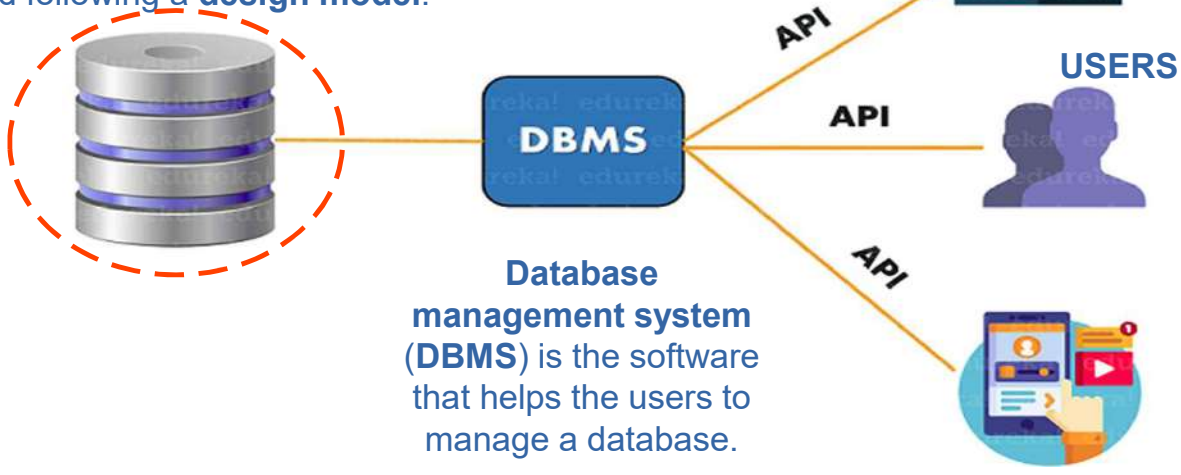


*1 exabyte = 1,000 petabytes =1 million terabytes = 1 billion gigabytes

7

What is database? - Scheme

A **database (DB)** is an organized collection of related information that is stored following a **design model**.



Database management system (**DBMS**) is the software that helps the users to manage a database.

8

What is database?

There are many different types of databases. The best database for a specific organization depends on how the organization (company, business, web, app, etc ...) will use the data.

Therefore, a database is designed according to a **Data Model** (**Dm**)or **Data Design Model** which meets the requirements requested by the organization.

9

What is data model (Dm)?

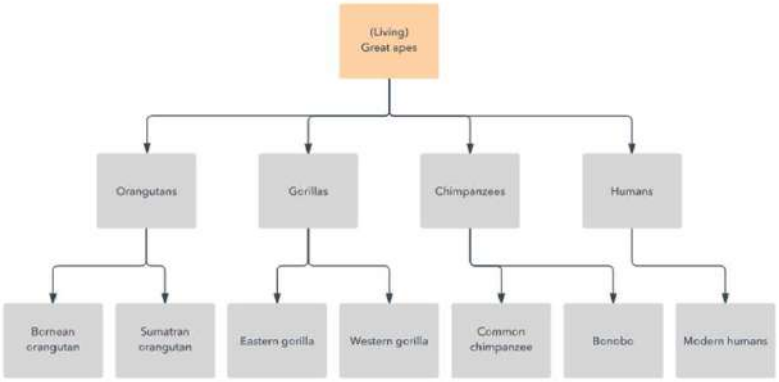
This shows the logical structure of a database, including the relationships and constraints that determines how data is stored, accessed and related.

Data Model is like an architect's building plan, which helps to build conceptual models and set a relationship between data items.

10

What is data model? - Types

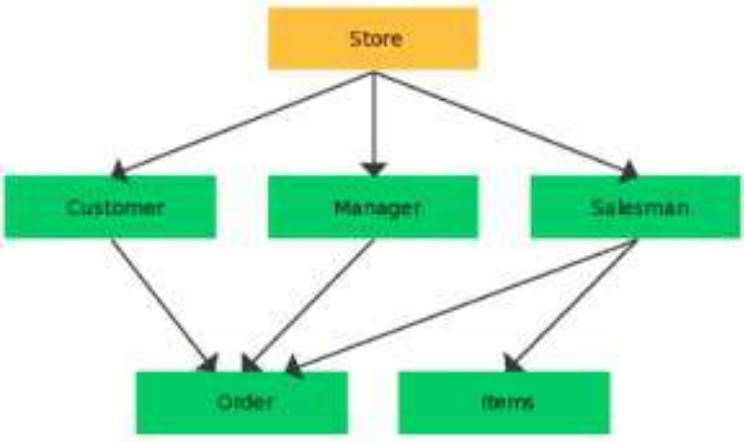
Hierarchical model: this design model gives rise to **Hierarchical Databases.**



11

What is data model? - Types

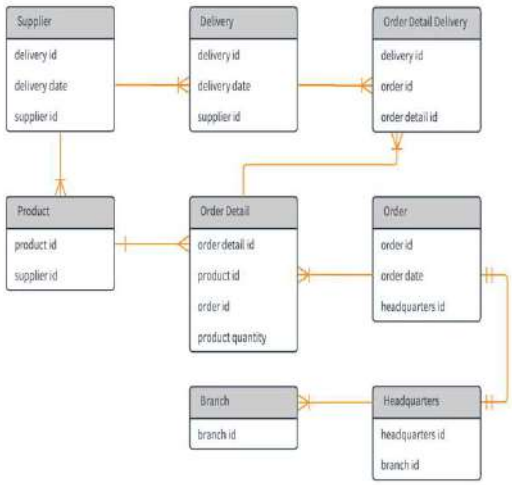
Network model: this design model gives rise to **Network Databases.**



12

What is data model? - Types

Relational model: this design model gives rise to **Relational Databases (SQL Databases).**



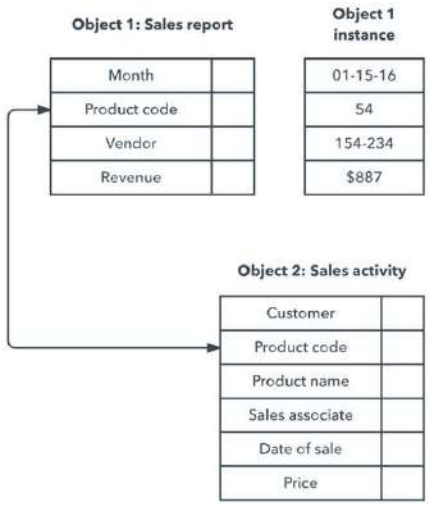
13

What is data model? - Types

Object-Oriented model: this design model gives rise to **Object-Oriented Databases.**

This model defines a database as a collection of objects with associated features and methods.

It incorporates tables, but it is not limited to tables.

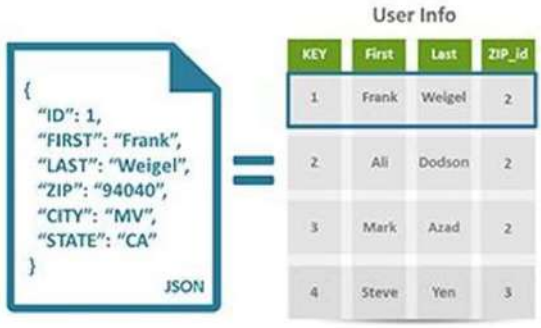


14

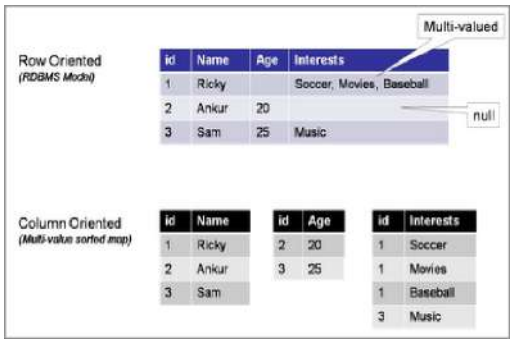
What is data model? - Types

No-Relational model: this design model gives rise to **No-Relational Databases.**

Document oriented



Column oriented



Unlike a relational database, the names and format of the columns can vary from one to other

15

What is data model? - Types

No-Relational model: this design model gives rise to **No-Relational Databases (NoSQL Databases).**

Key-Vlaue oriented

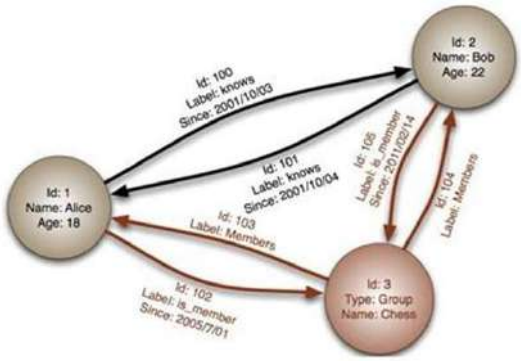
Students	
Key	Value
1	Name: Jean Grey DateOfBirth: 19-05-1963 IDCard: 1234567 PlaceOfOrigin: Austin Country: USA AcademicProgram_ID:1

2	Name: Scott Summers
	DateOfBirth: 12-10-1968
	IDCard: 765414A
	Supervisor: {
	Name: Emma Frost
	DateOfBirth: 1-1-1936
	IDCard: 222222
	}

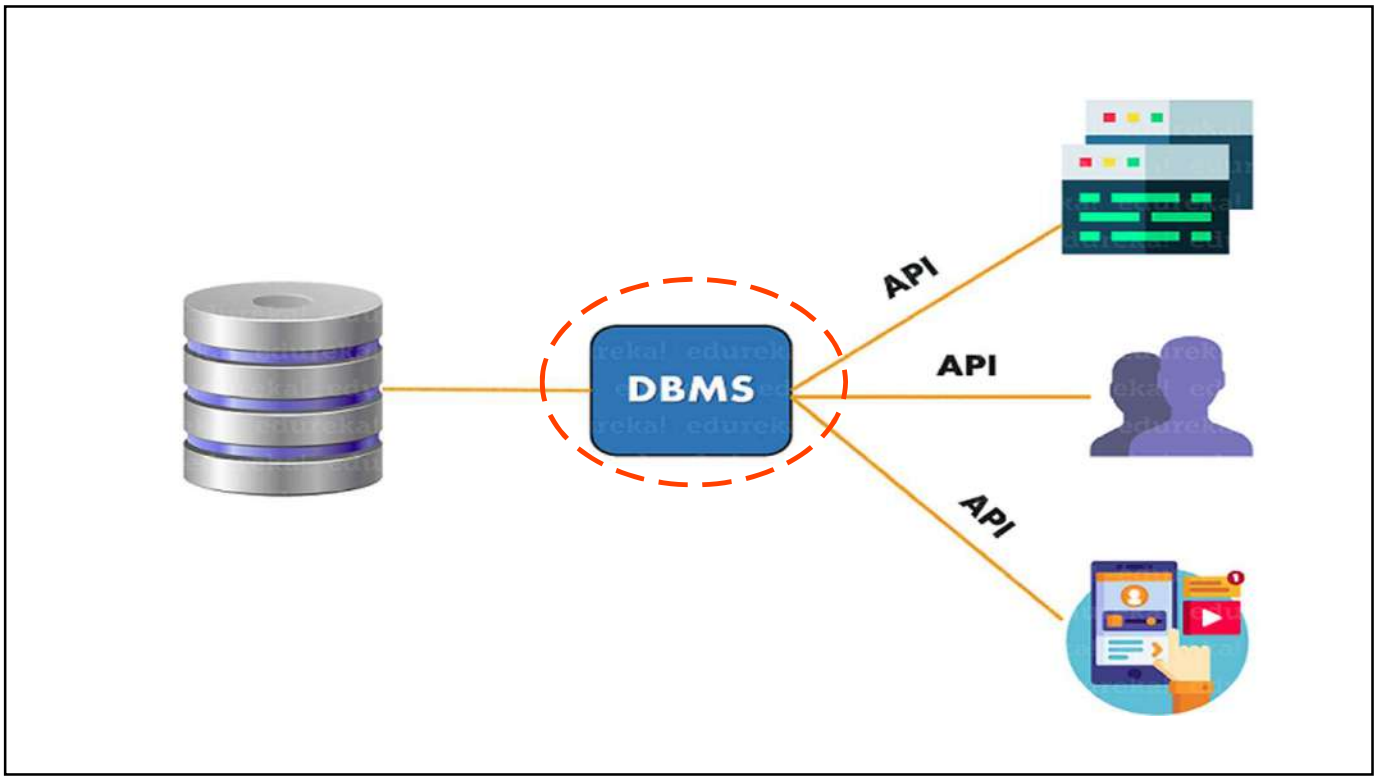
Professors	
Key	Value
1	Name: Charles Xavier DateOfBirth: 13-07-1940 IDCard: 111111 PlaceOfOrigin: Mirfield Country: UK
2	Name: Emma Frost DateOfBirth: 1-1-1936 IDCard: 222222

Tuples key-value. The value can be semistructured a data.

Graph



16



17

What is DBMS?

A DBMS is a software which serves as an interface between the database and its end users or programs. This allows users to:

- **Access**
- **Retrieve**
- **Update**
- **Manage**
- **Organized**
- **Optimized**

the information.

On the other hand, DBMS also facilitates oversight and control of databases, enabling a variety of **administrative operations** such as performance **monitoring**, **tuning**, **and backup and recovery.**

18

What is DBMS?-Characteristics

ABSTRACTION: details of physical storage are unknown for the normal user.

INDEPENDENCE: modify the database scheme without affecting the applications that depend on it

REDUNDANCY: the data is stored multiple time in different localization.

CONSISTENCY: the data repeated is updated simultaneously.

SECURITY: the data is safe against malicious activities.

INTEGRITY: the validity of the stored data is guaranteed

CONCURRENCY: a simultaneous access from different places is allowed without problem of consistency.

19

What is DBMS?-Functions

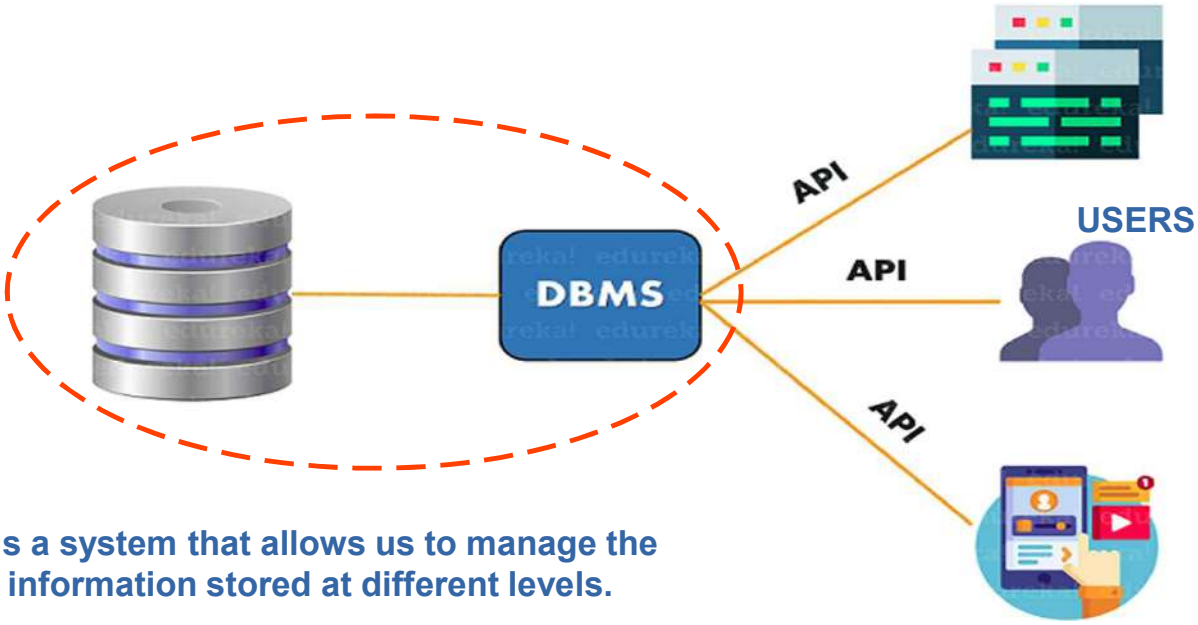
DATA DEFINITION: allow to define new kind of data by allowed users.

DATA MANIPULATION: allow the manipulation of data by allowed users.

MAINTENANCE Of PERMISSIONS AND ROLES: control and keep update the permissions and roles of the users.

TRANSACTION MANAGEMENT: allow multiple data transaction in parallel done by different user at the same time, without losing of consistency.

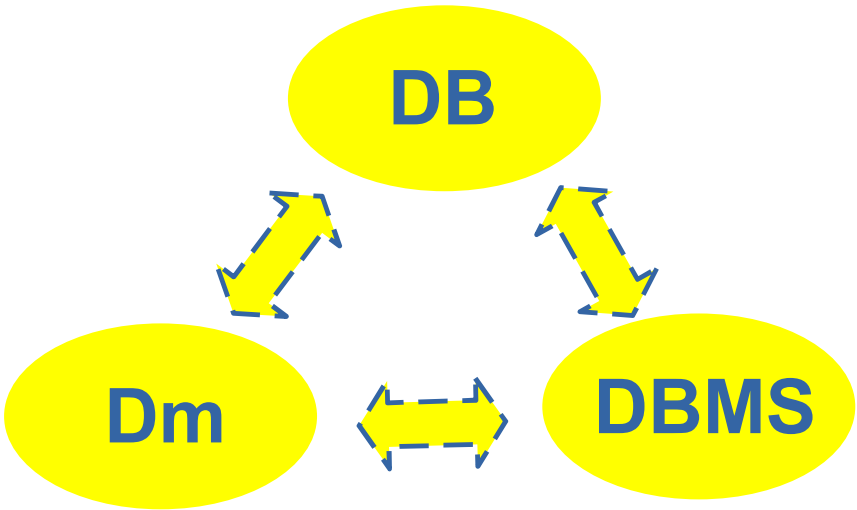
20



It is a system that allows us to manage the information stored at different levels.

21

DB: database
Dm: data model
DBMS: database management system



They are a three different aspect of the same thing.

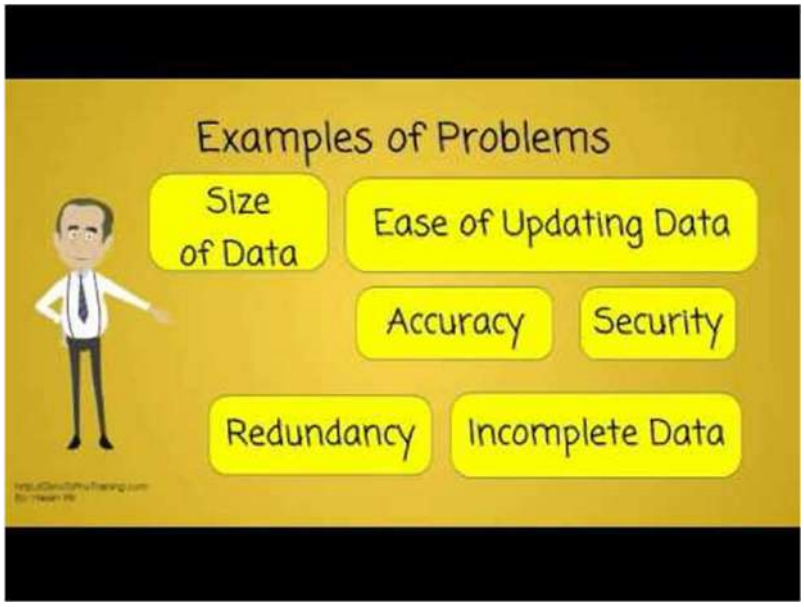
22

Table 1: Example of database (DDBB)

DBMS logo	DBMS	DB
MySQL	MySQL	Relational
ORACLE	Oracle	Relational
PostgreSQL	PostgreSQL	Combination of Relational and Object Oriented
mongoDB	MongoDB	Document oriented
cassandra	Cassandra	Key-value
neo4j	Neo4j	Graph

23

Why a Database?



https://www.youtube.com/watch?v=djEZeF4KTaM

24

Important Concepts

- What is data?
- What is a database?
- What is a data model?
- What is a database management system (DBMS)?
- Any characteristics and functions of a DBMS?

25

Relational Database concepts

Contents

1 Introduction **1**

2 Basic Concepts **2**
2.1 Relations (or tables) 3
2.2 Relationships 4
2.3 Keys 5

3 Design model in a relational database **6**

4 Types of RDBMS **7**

5 Local and in the cloud **8**
5.1 Cloud computing 9
5.2 Cloud service models 11
5.3 Cloud Database 11

1 Introduction

As we saw in unit 1, there are different **Data Models** that give rise to different **Databases**. From all those kinds of databases seen, almost the 95% of the market of databases is covered by two kinds [see Fig 1]:

- **Relational Databases** (**SQL-databases**)
- **Non-Relational Databases** (**NoSQL-databases**)

Remember that **SQL** (**Structured Query Language**) specific language used to manage data held in a **RELATIONAL DATABASE**. It is a programming language used to communicate with data stored in a **Relationa Database Management System** (**RDBMS**, a DBMS used with Relational Database). SQL syntax is similar to the English language, which makes it relatively easy to write, read, and interpret. Many RDBMSs use SQL or variations of SQL for accessing the data in tables.

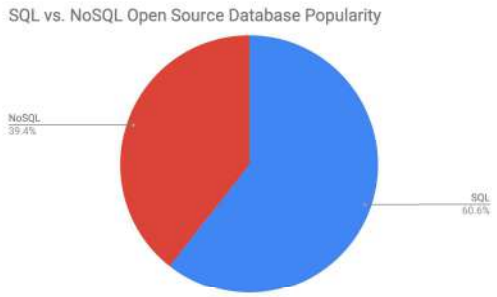


Figure 1: Database market.

The main reason to use one database or another is the type of data stored. So, Relational Databases are used to store structured data and Non-Relational Databases are used to store semistructured and unstructured data (see Figs. 2 and 3).

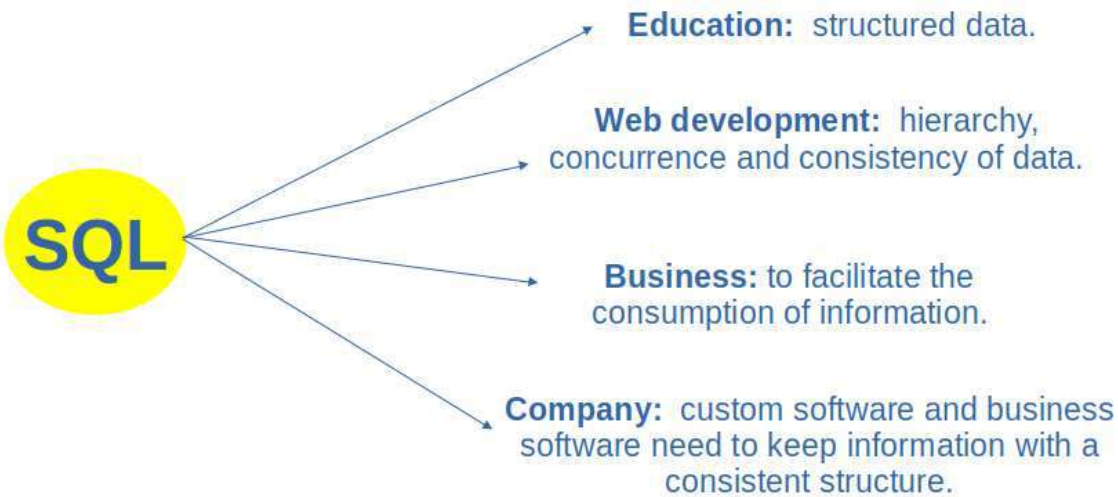


Figure 2: Main uses of SQL Database.

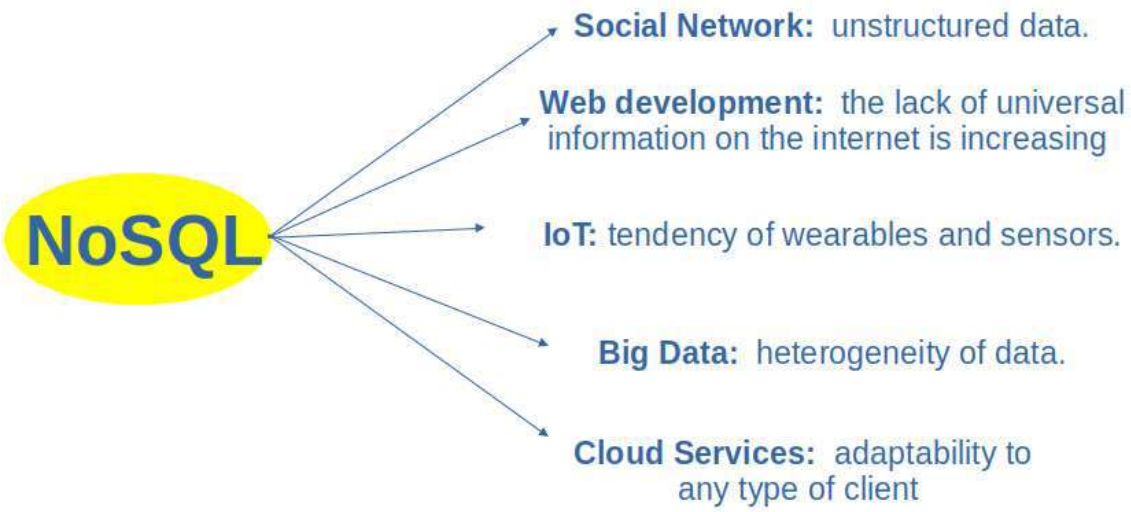


Figure 3: Main uses of NoSQL Database.

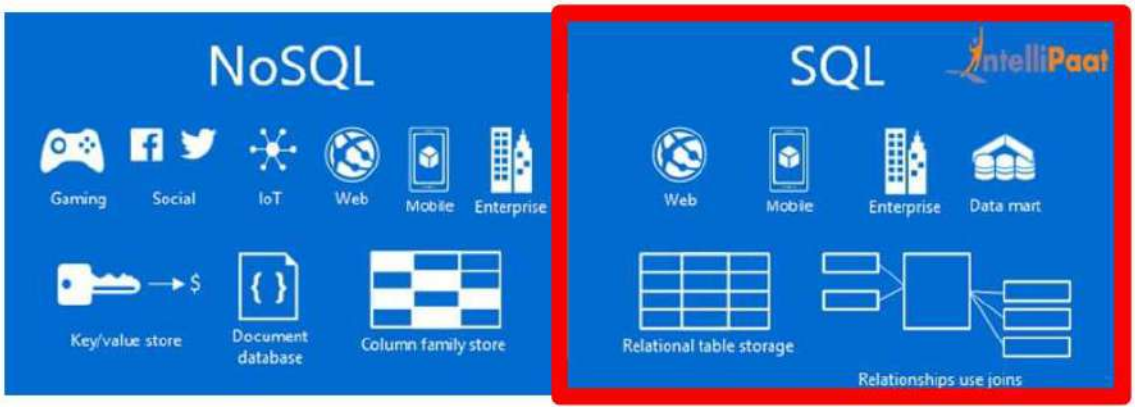


Figure 4: Comparative of main uses of SQL and NoSQL Database.

For the purpose of this course, we are going to focus on Relational Databases (red box in Fig. 4), both in the design as in the implementation and development using a Relational Database Management System.

2 Basic Concepts

As we know **Relational Database** are databases based on the relational model. This model is an intuitive and direct way of representing data in relations, where a relation is nothing but a table of values. In this model the data is organized and stored in tables. The tables can be linked or related to each other, relating the data among them and providing access to data through previously established relationships between them.

A Relational Database has three main components (see Fig. 5):

- **Relations (or tables)**
- **Relationships**

- **Keys**

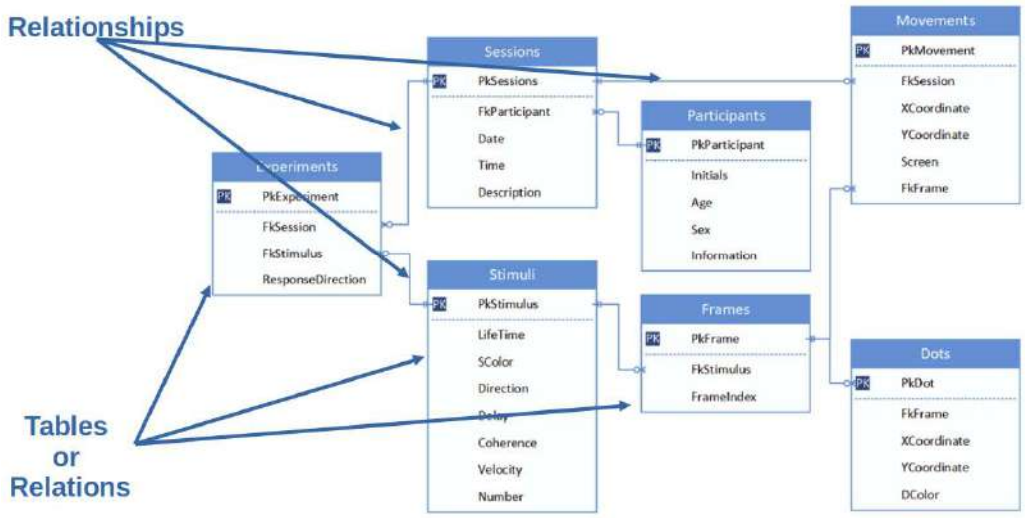


Figure 5: An example of relationa datbase, where we can see **Relations**, **Relationships** and **Primary keys (PK)** indicated.

2.1 Relations (or tables)

Relations or tables have a set of concepts related to them (see Fig. 6):

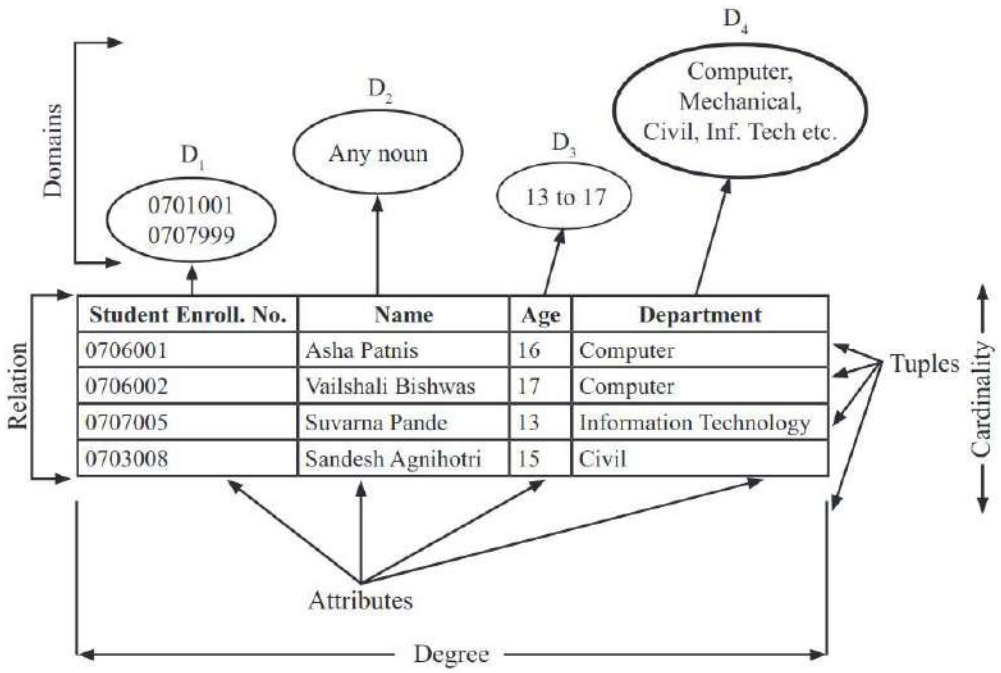


Figure 6: Concept of SQL database rable

- **Relation**: it is nothing but a table of values.
- **Domains**: range of valid values represented by an attribute.

- **Attributes**: each column in a table, to some extent the attributes define a table.
- **Tuple**: every single row of a table, which contains a single record.
- **Cardinality**: the total number of rows present in the table.
- **Degree**: the total number of attributes which in the relation.

2.2 Relationships

A **Relationships** has a characteristic called CARDINALITY. The cardinality indicates how the rows of different tables are related among them when a relationship exists between tables. IMPORTANT!!!! Do not confuse this CARDINALITY with the CARDINALITY of a table, they are different.

We are going to see 3 different types of cardinalities:

- **One to One**: a row of a tables is related to a row of another tables.

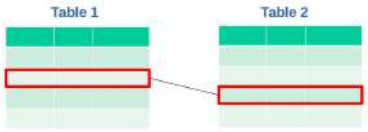


Figure 7: Cardinality **One to One**.

- **One to Many (or Many to One)**: one row in one table is related to many rows in another table and vice versa.

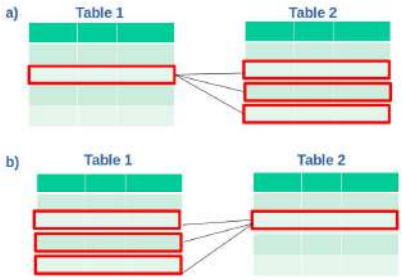


Figure 8: Cardinality: a) **One to Many** and b) **Many to One**.

- **Many to Many**:many rows in one table are related to many rows in another table.

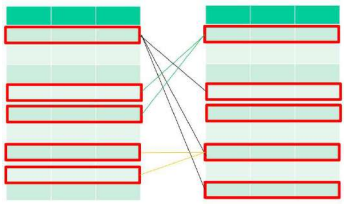


Figure 9: Cardinality: a) **One to Many** and b) **Many to One**.

2.3 Keys

A **key** in a Relational Database is an attribute or set of attributes that could help us to:

- to identify a row (tuple) in a relation (table)
- to relate tables to each other
- to identify a relation in a Relational Database.
- to identify a record in a table.

, for the purpose of this course we are going to see three different kinds of keys:

- **Candidate key**: it a set of attributes whose properties are interesting to be used them as a primary key, so they are potential attributes to be chosen as the primary key.
- **Primary key**: it is a attribute or set of attributes in a table that uniquely identifies each row in that table. A primary key can be made up using one or more attributes, depending on needs.
- **Foreign key**: it is an attribute that is the primary key in another table. Foreign keys are used to relate tables to each other.

Figures 10 and 11 show two examples comparing the concepts of primary, candidate and foreign key.

Figure 10: Example of how to choose a primary key from different candidate keys.

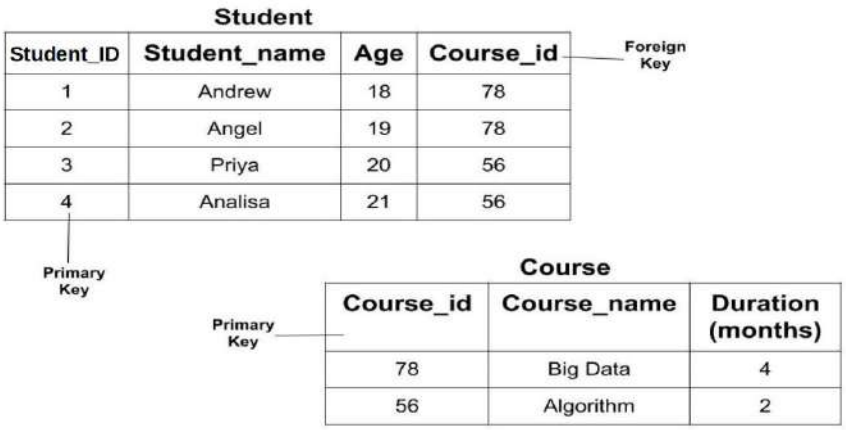


Figure 11: Example of how a pforeign key is used to relate two tables

IMPORTANT

- A primary key can be made up of an attribute or set of attributes.
- There cannot be two rows in the same table that have the same primary key value.
- Each row and record must be uniquely identified.

3 Design model in a relational database

Relational model, which as we know gives rise to Relational Databases, has three stages of design to help its implementation in a **RELATIONAL DBMS** (**RDBMS**). These phases are:

- **CONCEPTUAL**: in this stage we identify what possible tables the database will contain, and it also identifies the possible relationships between the different tables. At least, we identify the main tables and relationships although can appear other, depending on the needs of the database.

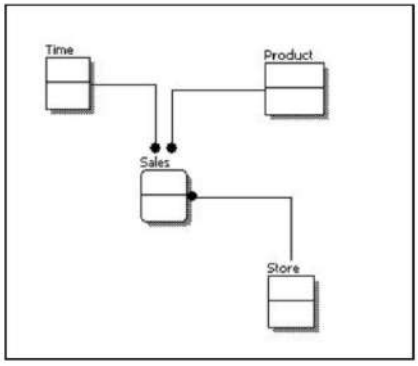


Figure 12: Conceptual stage of a relational database to store the data about sold products, considering the tables: products, time (of sale), stores and sales.

- **LOGICAL**: in this stage we describe the data in as much detail as possible, indicating: attributes, primary key and cardinality, but without regarding to how they will be physical implemented in the RDBMS.

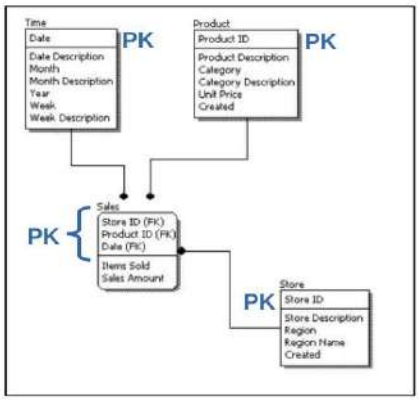


Figure 13: Logical stage of a relational database to store the data about sold products.

- **PHYSICAL**: this stage describes how the model will be implemented and developed in the RDBMS, giving more information about attributes.

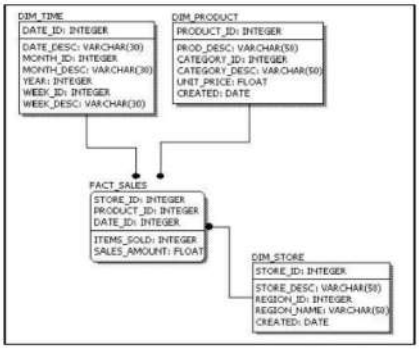


Figure 14: Logical stage of a relational database to store the data about sold products.

To implement these stages we are going to use a tool very useful called **Entity-Relationship Model**. This one help us to represent the database structure, how the data will be stored in a RDBMS and to set the relationship between data. We will see these three stages or phases in more detail in units 3 and 4.

4 Types of RDBMS

A relational database management system (RDBMS) is a software that allow us to create, update, and administer a relational database. In a RDBMS the data is stored, organized and accessible according to our relational model. The market-leading RDBMSs are:

- **MySQL**: it is the most popular open source SQL database. It is typically used for web application development.
- **SQLite**: it is a popular open source SQL database. The data can be stored locally without connection to a server, which is ideal for small devices (cellphones, iPads and other electronic gadgets).
- **PostgreSQL**: it is an open-source relational- object oriented database. It is also typically used for web application development.
- **Oracle**: it is the most popular no open source SQL database. It is noted for its flexibility, reliability and security. It is typically used for large applications such as companies or national institutions databases.
- **Microsoft SQL Server** : it is another no open source SQL database. Recent enhancements include a web edition and new business intelligence analytics tools. Along Oracle, it is one of the most used in large companies.

As we can see, the choice of a RDBMS depends on the type of application and the amount of data.

5 Local and in the cloud

The efficient management and storage of data has been a classic problem for information technologies. Not long ago the local storage was the only option for saving precious files such as images, documents and videos. This involved having local servers, which required physical space, configuration, and maintenance, meaning an investment of equipment, infrastructure and trained personnel. At this point, let's consider the following situation:

> **Suppose you have a company where a new line of business for the sale of goods and services is opening. It is thought that this line will generate a quantity of important data, which must be stored. They are potentially useful since from the analysis of this data you can see the possible problems or new business models.**
>
> ***What type of server or servers would you choose?***

Theoretically a large amount of data will be stored and analysed on these servers, therefore servers with large capacity should be considered. In addition, this implies having a technical infrastructure (place, connection, electrical supply, refrigeration, security, ...) and a trained staff in various fields of IT technologies. Therefore, the company needs to make a significant initial investment in all of them. If the estimations are correct the company will have a return on the money invested more or less important.

Now, we are going to consider the situation, where the company makes a significant initial investment but the results are not as expected. Either because the data collected is not as much as expected or the data analysis does not give significant results, in any case a significant initial investment has been made in an infrastructure (equipment and personnel) that does not provide the expected results.

We can also have the opposite case, the company underestimates the amount of data to collect and its importance. So the company makes a small investment choosing small servers and a reduced team of qualified personnel, but in this case the amount of data collected exceeds the capacity of the servers and the staff is not prepared for the management and analysis of so much amount of data. This implies that the company needs to change the server and hires more trained personal, but while this happens the company is losing money because it can not store neither analyse the data.

In any case, the company has the danger of overestimating or underestimating the computational needs of the business, having a high probability of losing money in one way or another.

Beside, there are two crucial points currently related to the data storage:

- the current amount of data generated continuously.
- the necessity of accessing to the data from any point and at any time.

All above implies that the storage of this data must meet a set of requirements, which are:

- **Implementation**: the implementation or start-up of a server varies a lot according to its nature of the company's needs. The larger the server or server system, the more complex it will be the management, configuration and maintenance.

- **Scalability**: as a company grows, it generates a greater volume of information that needs a space to stay safely.
- **Security**: servers servers are the main target of computer viruses and data theft. Therefore, security protocols must be strict if we want to guarantee the protection of our company.
- **Availability**: servers must be available 24/7, preventing the loss of services and data due to disconnection or failure of the servers
- **Accessibility**: allow access to the servers and their services from anywhere and at any time without restrictions and without putting the security of the company and data at risk.

at this point the question is:

Is there any solution to reduce the risk to a minimum?

The answer is "**Yes, there is**". It is known as "**Cloud Computing**", colloquially "**the cloud**", but what is "**the cloud**"?

5.1 Cloud computing

Although, previously we have refereed to the cloud in relation to data storages, the concept of the cloud is broader. Cloud Computing is a model that allows ubiquitous and on-demand access to a set of configurable computing and storage resources, which can be rapidly provisioned and released with minimal interaction with the provider.

Cloud Computing allows organizations to outsource some of their computing and storage to a third party provider, which typically offers a pay-per-use model. A Cloud provider allows offering access to its resources (storage and computing, among others) to multiple clients, allowing clients to reduce their costs (investment in hardware, cooling of equipment, conditioning of a premises to store servers, etc.). At this point a clarification is necessary, when we speak of **Cloud Provider** we refer to both **NO Public Providers** and **Public Providers**.

Cloud Computing is a paradigm that allows dynamic provisioning of resources, generally computing and storage, typically from a third party, through a pay-per-use model, in the case of using a public Cloud. **The underlying idea is to outsource. Just like you do not manufacture your own energy, but consume it from an external provider (which in turn bills the customer based on the consumption made)**, with Cloud Computing you can outsource (all or part of, depending on needs) the computation and storage to a external provider.

How does the concept of cloud computing respond to the previously mentioned requirements?

- **Implementation**: with cloud computing the company does not worry about the implementation (management, configuration and maintenance) of the servers. The cloud service provider takes care of all this.
- **Scalability**: cloud computing offers flexibility to companies that are growing or have specific needs, allowing a management of storage and services simple and agile.

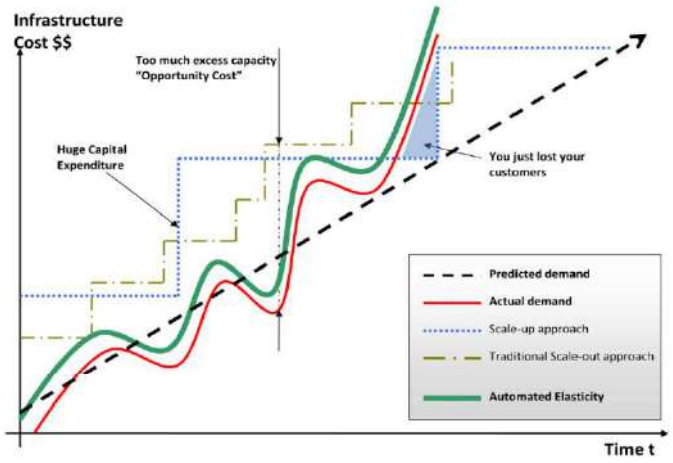


Figure 15: Dynamic adjustment of infrastructure to variable resource demand.

- **Security**: servers are the main target of computer viruses and data theft. Therefore, security protocols must be strict if we want to guarantee the protection of our company. The great advantage of servers in the cloud is that they always have the latest versions in terms of security against the latest malware threats. On the other hand, the information stored there is encrypted, which makes it difficult to access sensitive information. In addition, the information is replicated in different places so that the total theft of information is impossible.
- **Availability**: This guarantees availability by offering an automatic backup to the servers. That is, servers in the cloud can fail just like local ones, however, if a server in the cloud fails, the information jumps to another in a matter of seconds without affecting or hindering the work of the teams.
- **Accessibility**: it allows from anywhere and at any time to access the data and computing resources, without restrictions and without putting the security of the company at risk.

Servidor en la nube	Servidor local
Precios más bajos y escalables	Altos costes de equipo y servicios
Actualizaciones automáticas	Costes de actualización y renovación
99.9 % accesibilidad	Susceptible de sufrir problemas o fallos
Sin coste de infraestructura ni soporte	Necesidad de un espacio físico
Sin necesidad de backup	Respaldo manual
Sin consumo energético	Alto consumo energético.
Información disponible 24/7/ 365	Coste por acceso remoto
Altos estándares de seguridad	La seguridad depende de la empresa
Pago por servicio SaaS	Coste de servidor + configuración
Escalabilidad infinita	Limitado al crecimiento de la empresa

prodware

Figure 16: Local and cloud comparative.

5.2 Cloud service models

In addition, regardless of the deployment models, there are different service models that determine the type of service that a given Cloud provider offers its users. This is not an exhaustive classification, but there is considerable consensus in the community on this (see Fig. 17):

- **Infrastructure as a Service (IaaS)**: this model is oriented to system administrators. With this approach, the provider offers access to computing and storage resources through a pay-per-use model. For instance: Amazon EC2 or Amazon Bucket.
- **Platform as a Service (PaaS)**: this model is aimed at application developers. The developer uses a development kit (Softtware Development Kit, SDK) to build an application that uses the multiple services of the Cloud platform. It is a higher level of abstraction where the developer does not worry about the operation of servers. For instance: MicrosoftAzure, Google CloudPlatform o Heroku.
- **Software as a Service (SaaS)**: this model is geared towards end users, who use a web browser to access end applications. This is the case for web applications such as GMail, Office 365 or Google Docs.

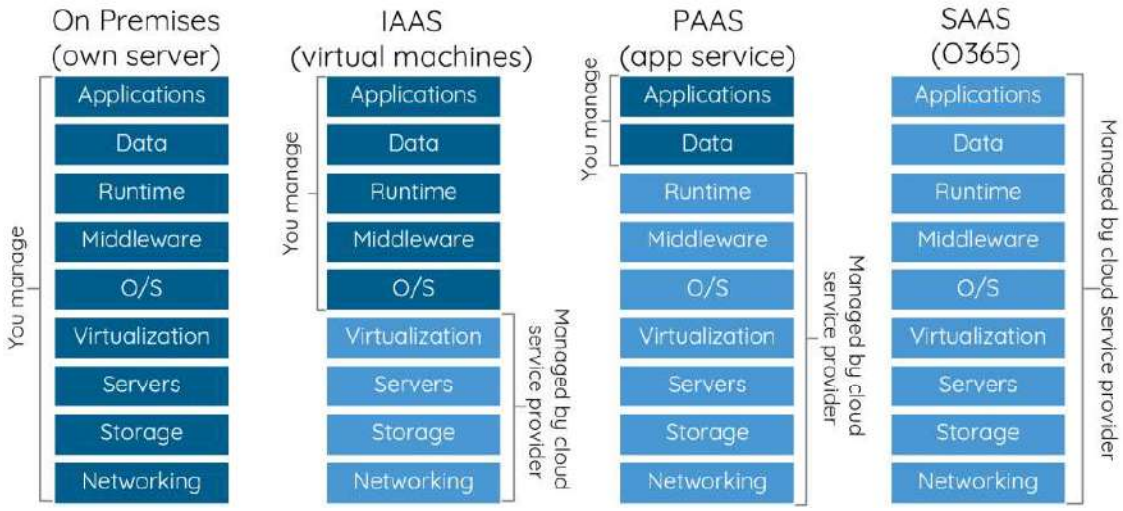


Figure 17: Local and cloud comparative.

5.3 Cloud Database

Cloud Database is a database that runs on a cloud computing platform. So, the database is provided as a service, considering Cloud Databases as SaaS. The data is remotely maintained and managed. This service allows us, from any location and on-demand, to access the data.

There are three main factors involved to decide between **On-Premise Database** and **Cloud Database**:

- **Hardware cost**
  - Servers: power computing and storage.

  - Limitation of our own server: restricted to one server and not to adapt to demand.
  - Infrastructure: physical place, network, energy and refrigeration.
- **Software cost**
  - Payment of a licence
- **Operational cost**
  - Qualified personal
  - Services running 24/7
  - Hidden cost: back ups, updates, configurations and interoperability

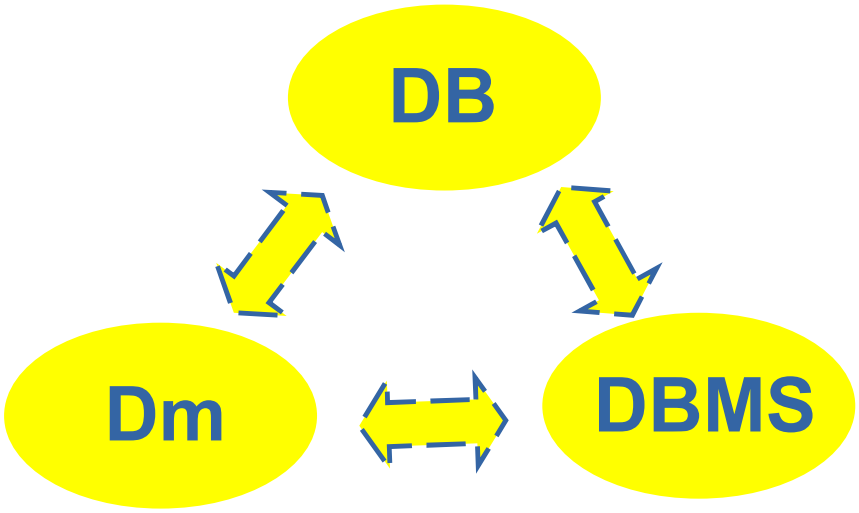
1

Unit 2 – Relational Database concepts

- Basic concepts
- Design model in a relational database
- Types of Relational DBMS (RDBMS)
- Local and in the cloud
- Cloud computing

2

DB: database
Dm: data model
DBMS: database management system



They are a three different aspect of the same thing.

3

Data model ⟷ Database

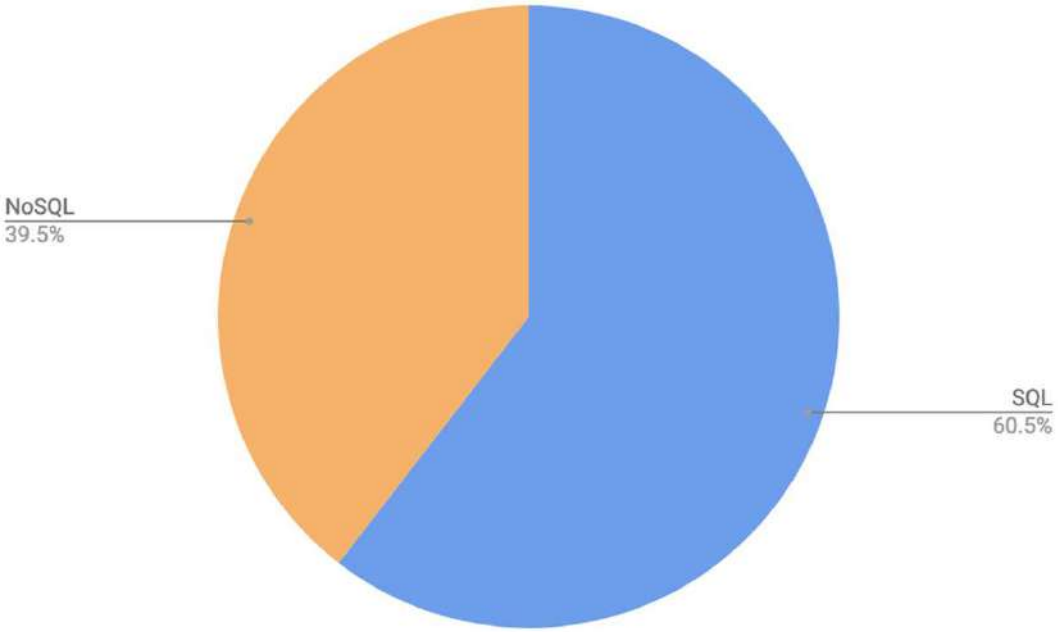
- Hierarchical model → Hierarchical Databases
- Network model → Network Databases
- Relational model → Relational Databases (SQL-databases)
- Object-Oriented model → Object-Oriented Databases
- No-Relational model → No-Relational Databases (NoSQL-databases)

SQL (Structured Query Language) specific language used in programming and designed for managing data held in a RELATIONAL DATABASE.

Relational Database ⟷ SQL Database

4

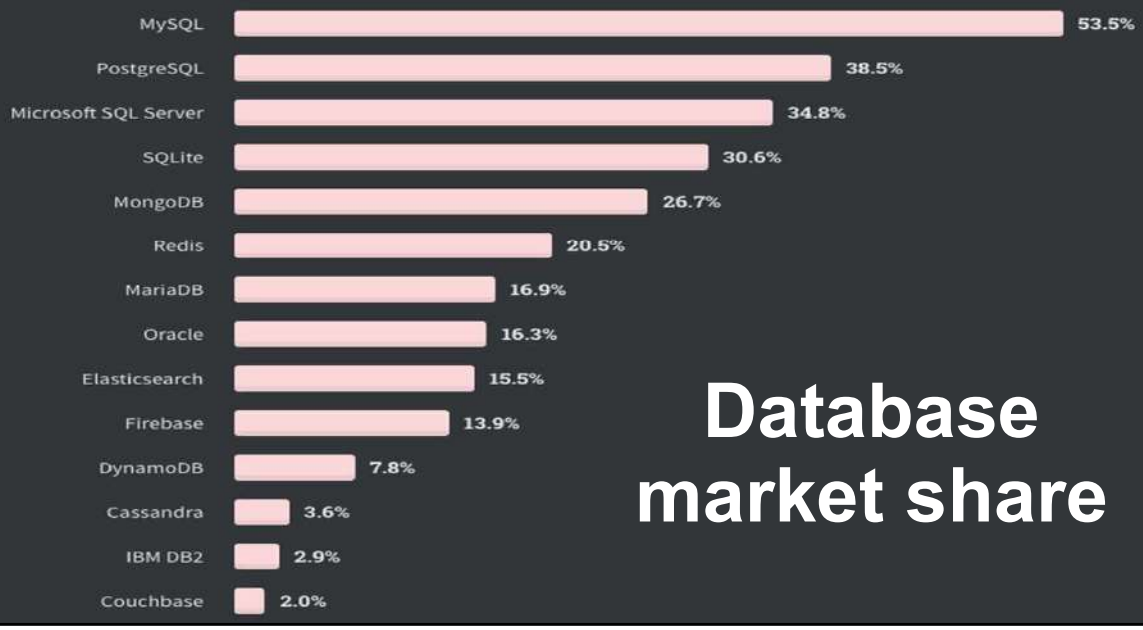
Database market share



5

All Respondents | Professional Developers | 41,811 responses; select all that apply

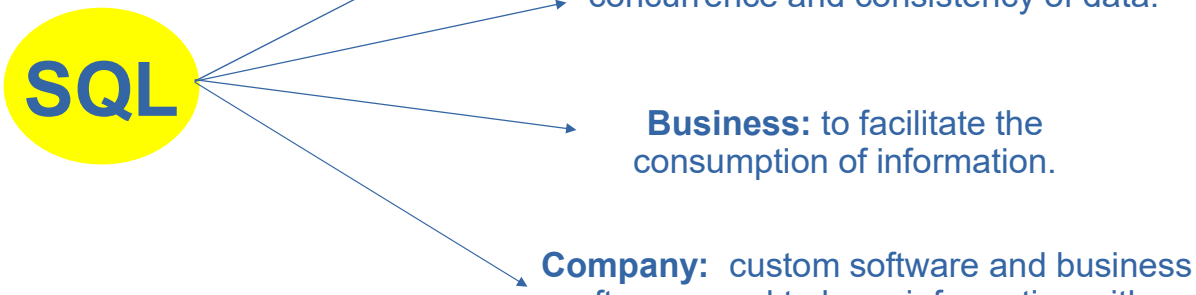
DBMS



Database market share

6

SQL vs NoSQL
When to use each one?



Education: structured data.

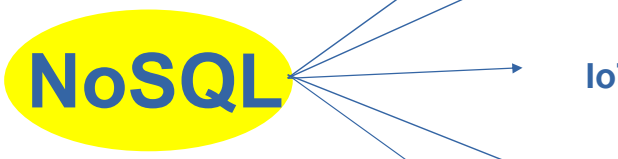
Web development: hierarchy, concurrence and consistency of data.

Business: to facilitate the consumption of information.

Company: custom software and business software need to keep information with a consistent structure.

7

SQL vs NoSQL
When to use each one?



Social Network: unstructured data.

Web development: the lack of universal information on the internet is increasing

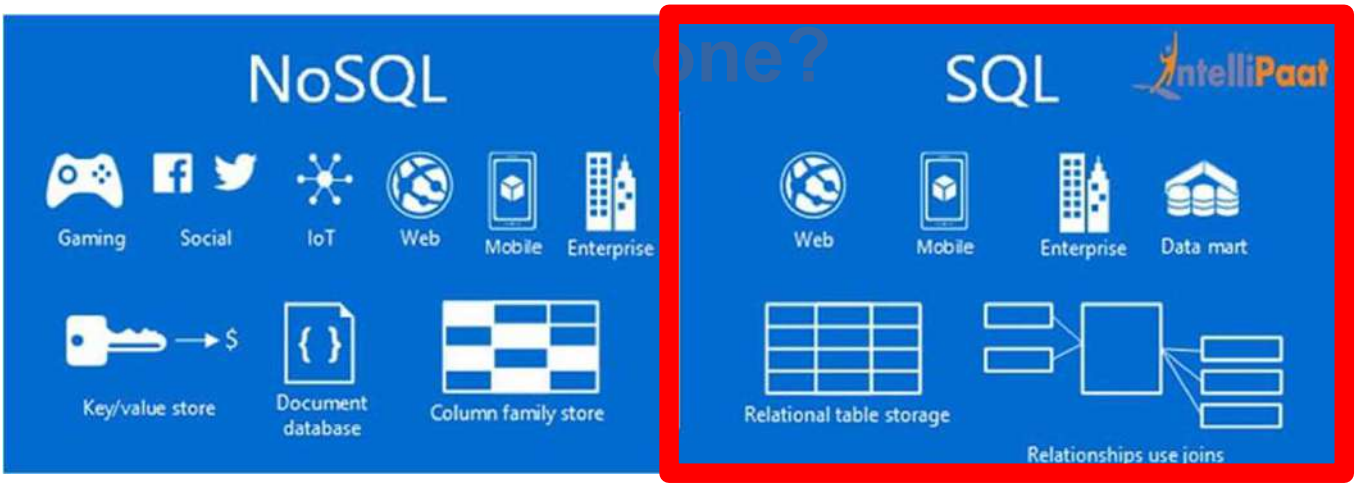
IoT: tendency of wearables and sensors.

Big Data: heterogeneity of data.

Cloud Services: adaptability to any type of client

8

SQL vs NoSQL
When to use each



9

Relation Database (SQL Database)

A **Relation Database** or **SQL Database Education** are databases based on the **relational model**. This model is an intuitive and direct way of representing data in relations (table of values).

So, this model organizes the data in tables (all data is stored in tables) which can be linked or related based on data common to each, providing access to data through previously established relationships between them.

10

Relation Database Management System (RDMS)

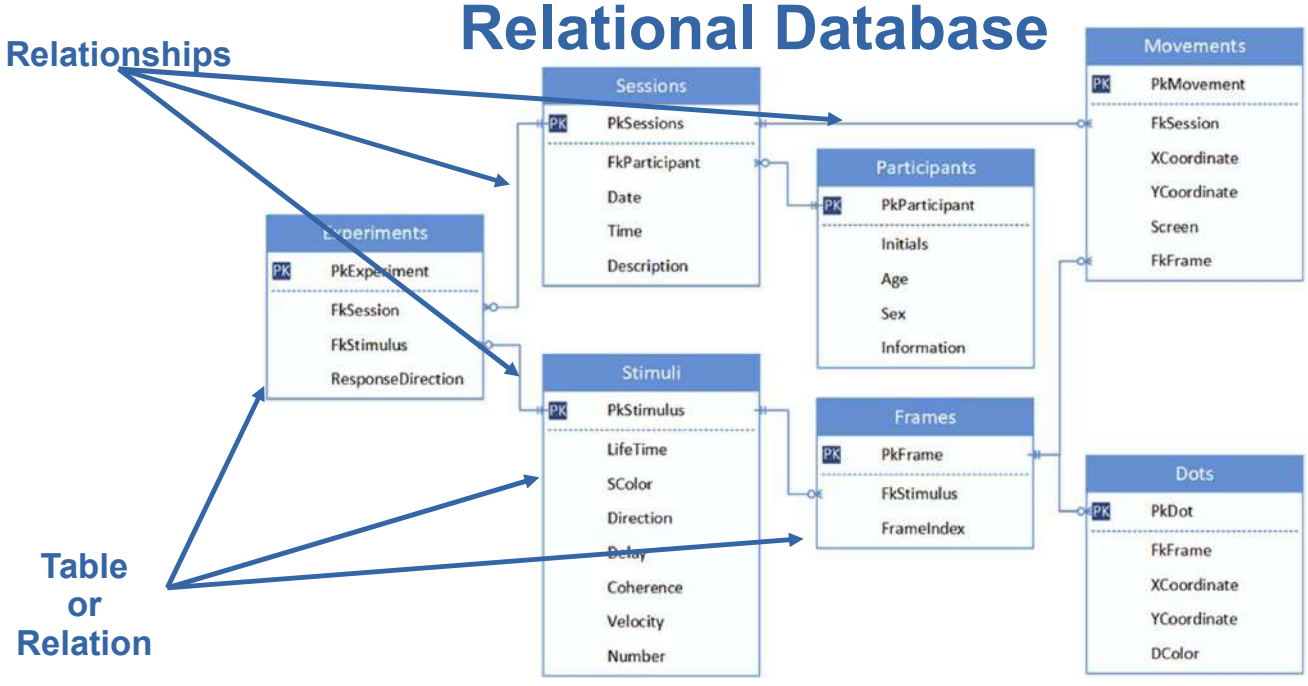
A **relational database management system** (**RDBMS**) is a program (software) that allows you to create, update, and administer a relational database. In a RDBMS the data is organized and accessible according to the relationships between data. As it was said above, relational databases use **Structured Query Language** (**SQL**) to access the database.

What is SQL?

SQL (**Structured Query Language**) is a programming language used to communicate with data stored in a RDBMS. SQL syntax is similar to the English language, which makes it relatively easy to write, read, and interpret. Many RDBMSs use SQL (and variations of SQL) to access the data in tables.

11

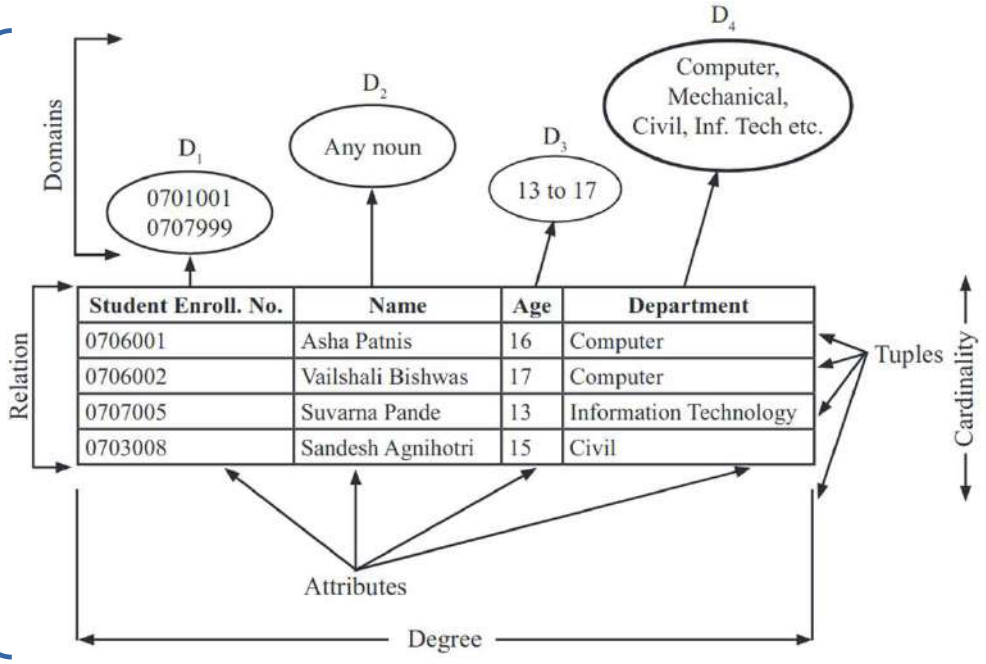
Structure of a Relational Database



12

Basic concepts SQL-database

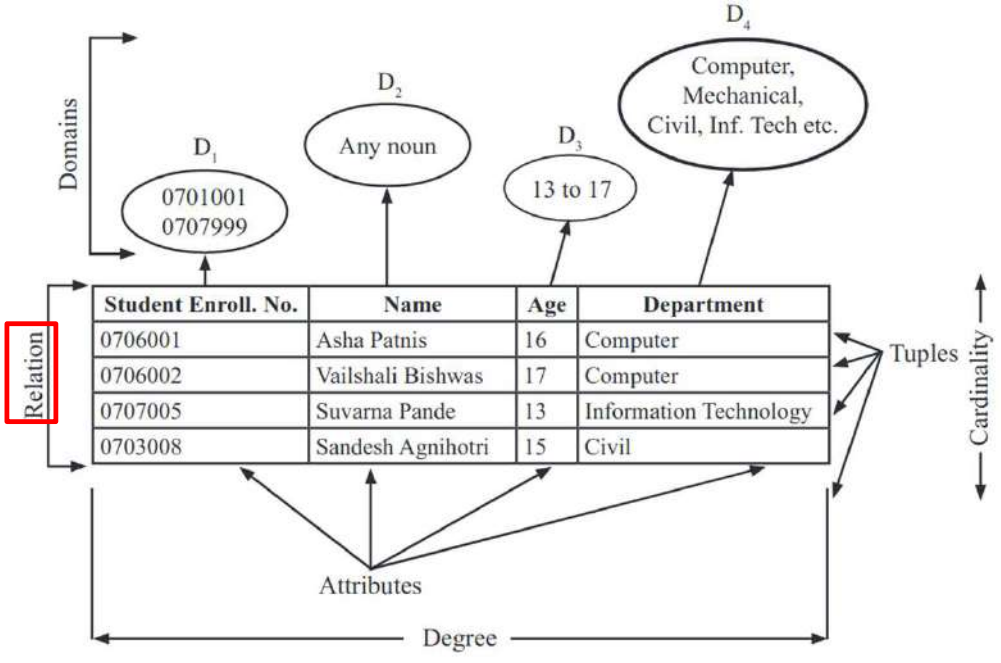
Elements of a Relation



13

Basic concepts SQL-database

Relation or Table:
it a table of values.

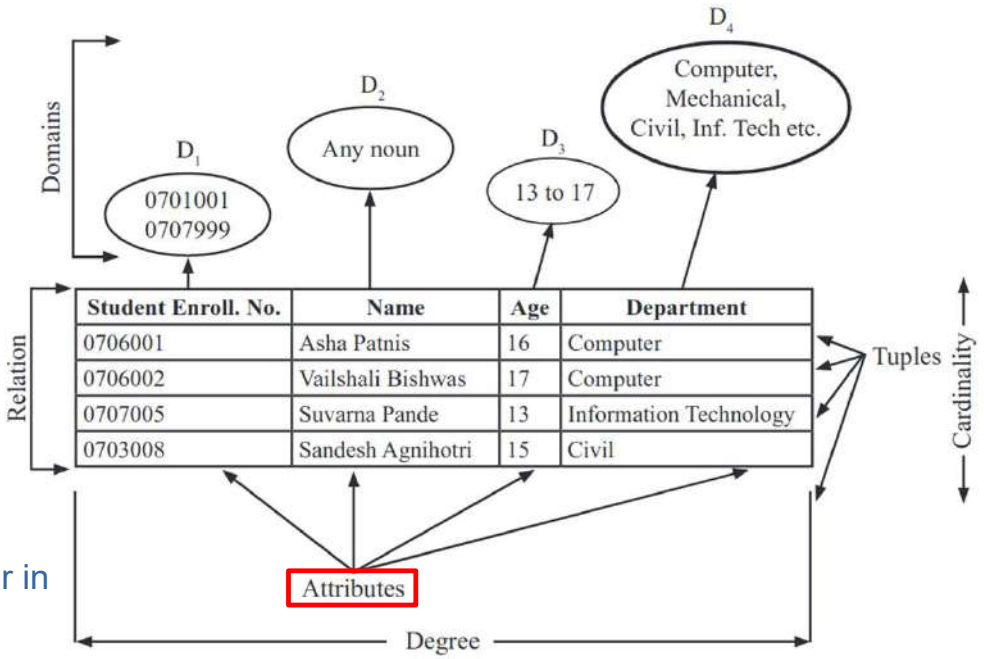


14

Basic concepts SQL-database

Attribute:
each column in a table.

IMPORTANT!!!
An attribute can appear in several tables in the same database.

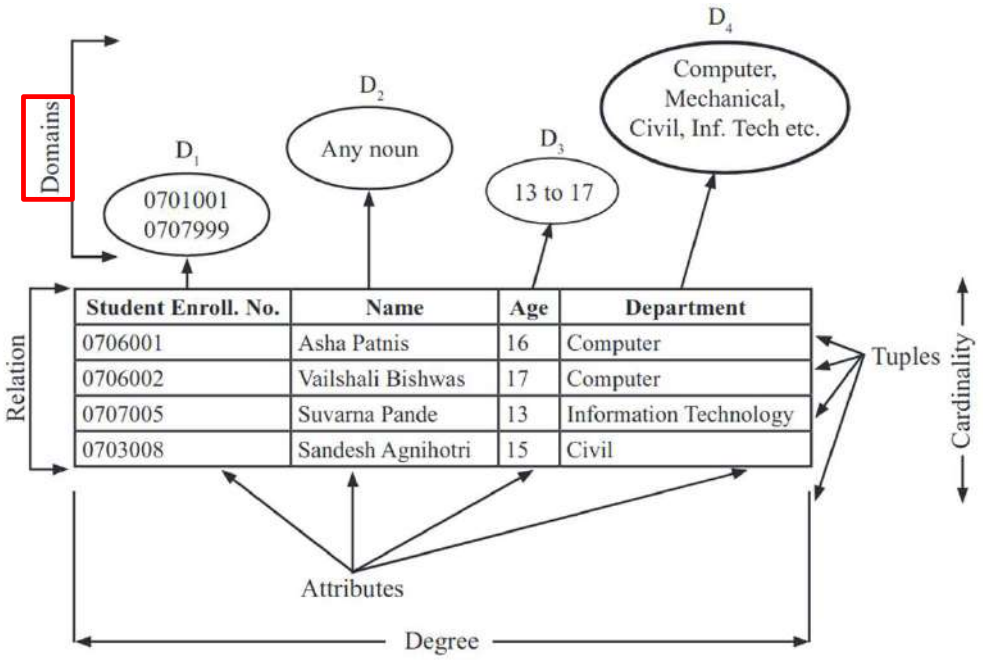


Student Enroll. No.	Name	Age	Department
0706001	Asha Patnis	16	Computer
0706002	Vailshali Bishwas	17	Computer
0707005	Suvarna Pande	13	Information Technology
0703008	Sandesh Agnihotri	15	Civil

15

Basic concepts SQL-database

Domain:
range of valid values that an attribute can take.

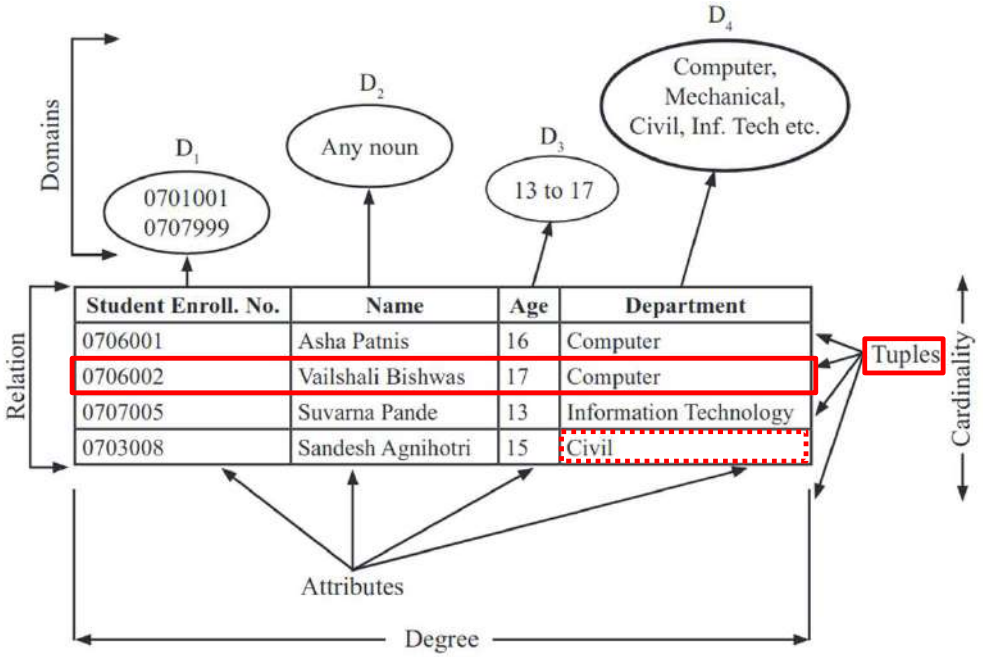


Student Enroll. No.	Name	Age	Department
0706001	Asha Patnis	16	Computer
0706002	Vailshali Bishwas	17	Computer
0707005	Suvarna Pande	13	Information Technology
0703008	Sandesh Agnihotri	15	Civil

16

Basic concepts SQL-database

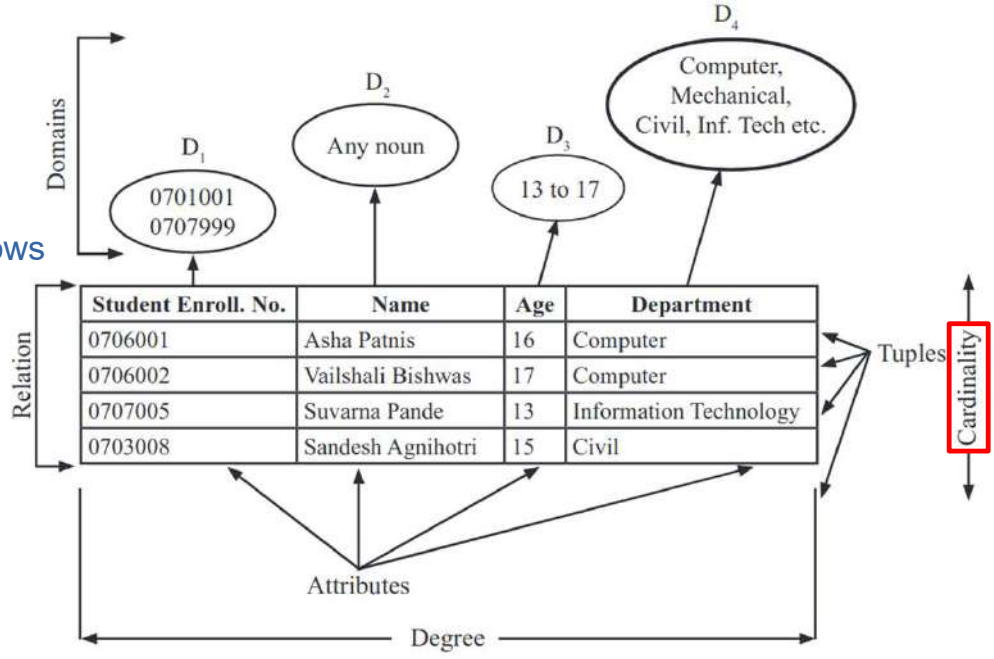
Tuple:
every single row of a table, which contains a set records.



17

Basic concepts SQL-database

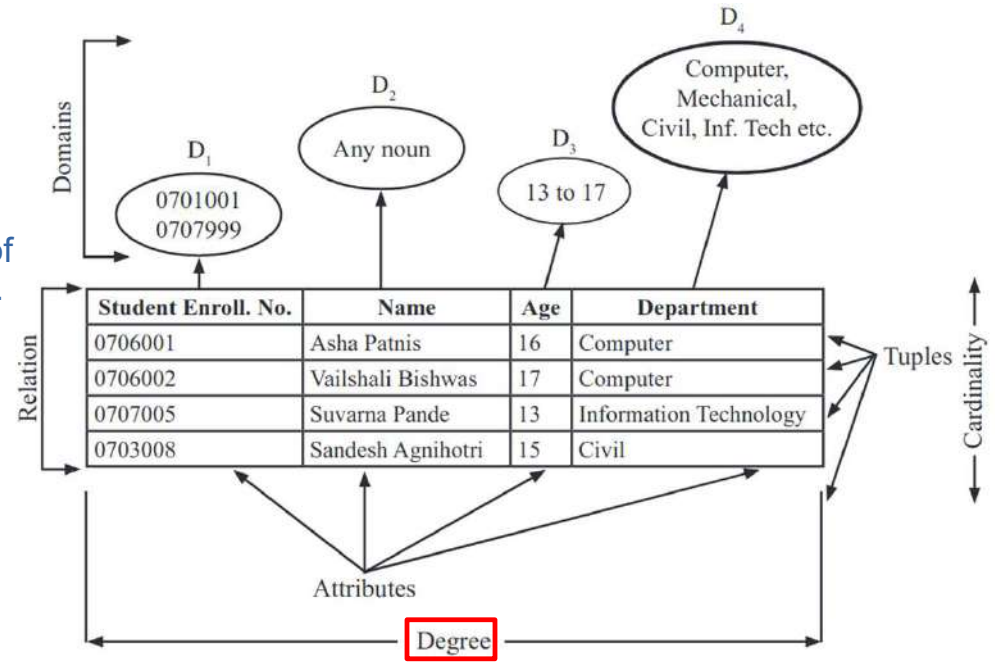
Cardinality:
the total number of rows present in the table.



18

Basic concepts SQL-database

Degree:
the total number of attributes in the relation.

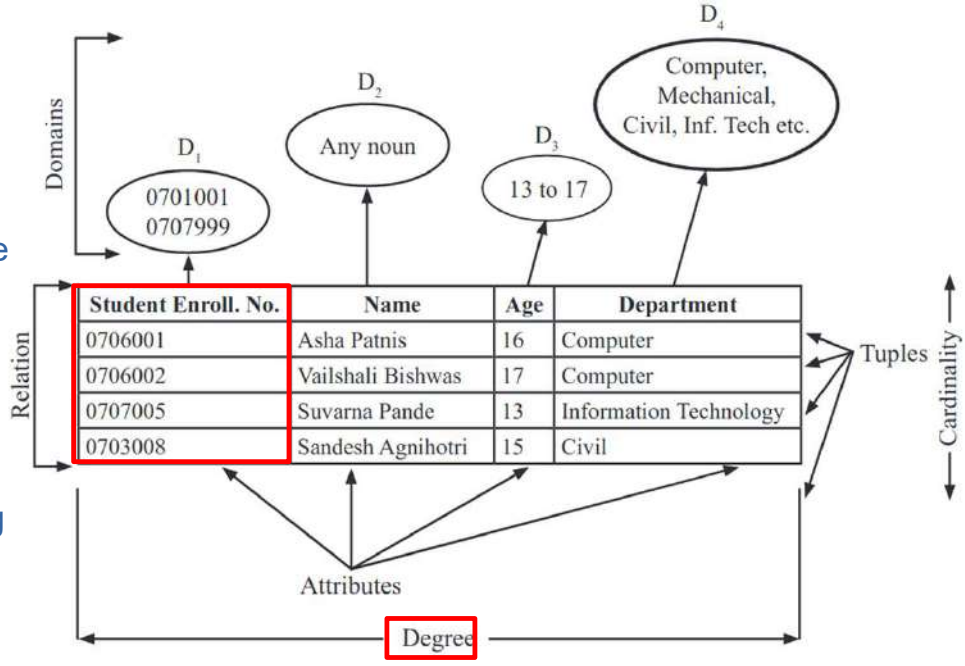


19

Basic concepts SQL-database

Key:
key in a DB is an attribute or set of attributes which helps us to identify a row (tuple) in a relation (table).

Moreover, they allow us to find the relation among tables.



20

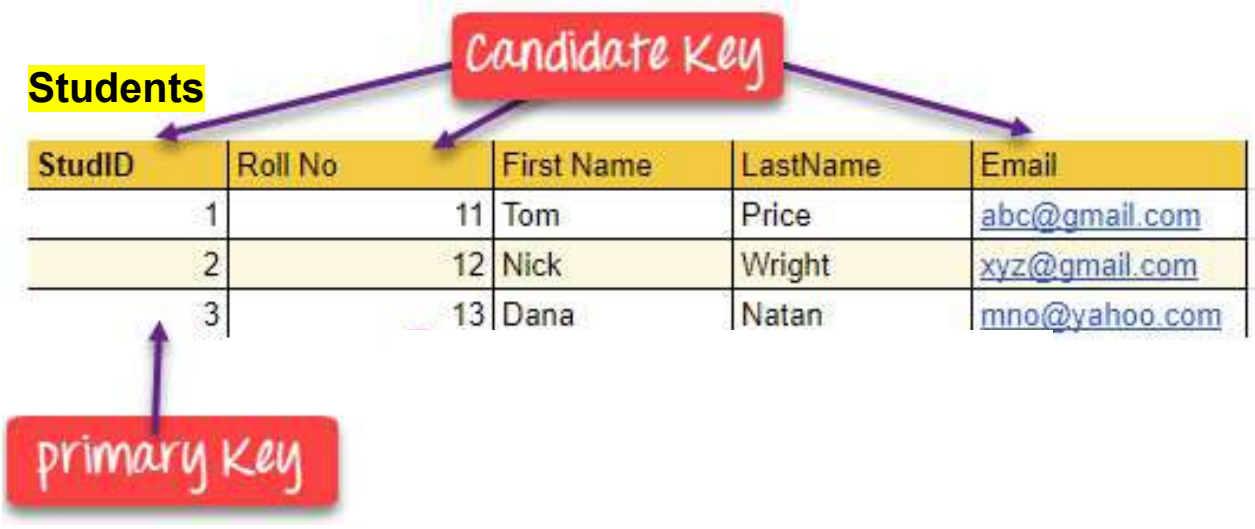
Basic concepts SQL-database

Key:
key in DB is an attribute or set of attributes which helps us to identify a row(tuple) in a relation(table), beside they allow us to find the relationships among tables.

- **Candidate key**: an attribute which could help us to uniquely identify rows and records in a table and to establish relationships among tables.
  A potential attribute to be used as a primary key.
- **Primary key**: attribute or set of attributes chosen from a set of candidate keys that allow us to uniquely identify each row in a table.
- **Foreign key**: it is an attribute which is the **Primary Key** in another table.

21

Basic concepts SQL-database



22

Basic concepts SQL-database

Student

Student_ID	Student_name	Age	Course_id
1	Andrew	18	78
2	Angel	19	78
3	Priya	20	56
4	Analisa	21	56

Foreign Key

Primary Key

Course

Primary Key

Course_id	Course_name	Duration (months)
78	Big Data	4
56	Algorithm	2

23

Basic concepts SQL-database

Primary Key

Candidate Key

Candidate Key

Candidate Key

Foreign Key

Students

Student_ID	Enroll Number	Roll Number	Name	Address	City	Dept_ID
1	AX001	1-BSCS-2018	John	Street 13 House 14	Lahore	3
2	AX002	2-BSBT-2018	Faiz	house 18 Defence Club	Karachi	4
3	AX003	3-BSEN-2018	Nouman	Street 19 House 20	Faislabad	5

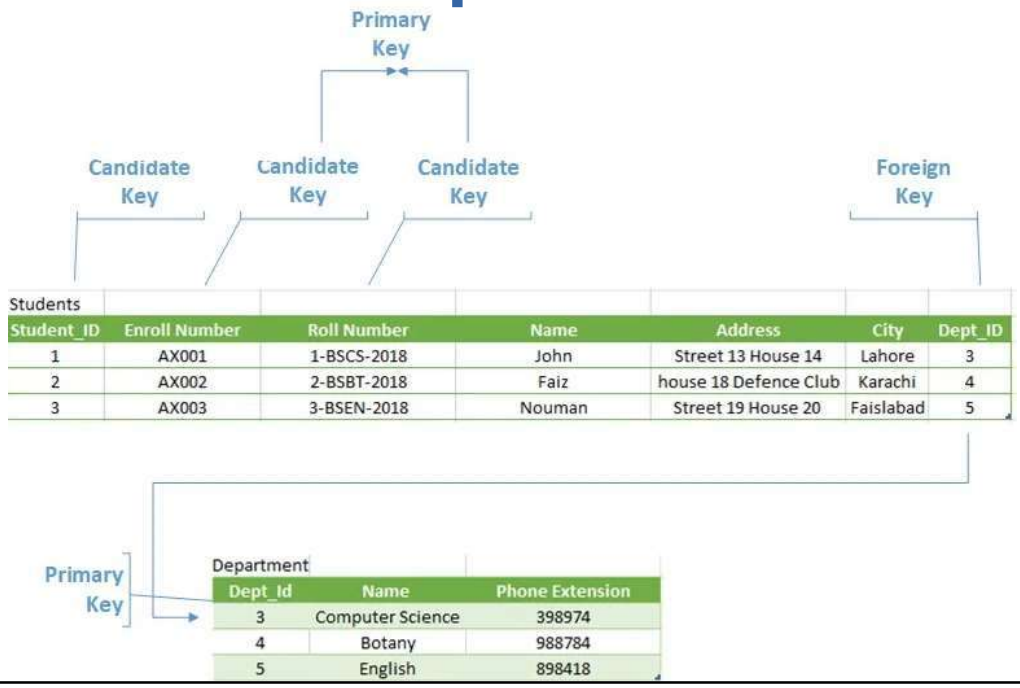
Primary Key

Department

Dept_Id	Name	Phone Extension
3	Computer Science	398974
4	Botany	988784
5	English	898418

24

Basic concepts SQL-database



25

Primary Key concept

IMPORTANT!!!

1. A primary key can be made up of an attribute or set of attributes.
2. There cannot be two rows in the same table that have the same primary key value.
3. Each row and record must be uniquely identified.

26

Basic concepts SQL-database

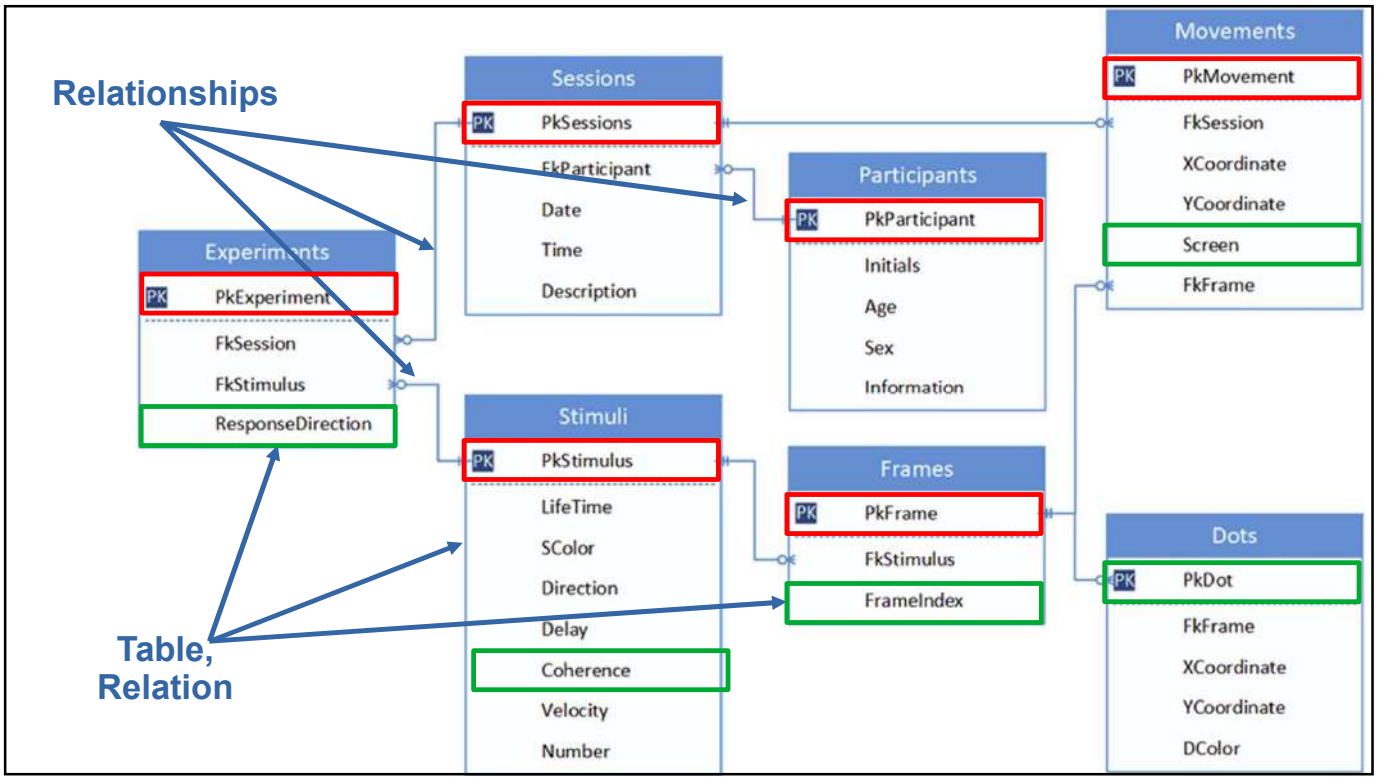
Each record must be uniquely identified.

Student

Student_ID	Student_name	Age	Course_id
1	Andrew	18	78
2	Angel	19	78
3	Priya	20	56
4	Analisa	21	56

Primary Key

27



28

Data model in a SQL-database

Relational model has **three stages** of design to help its implementation in a RELATIONAL DBMS (RDBMS). These phases are:

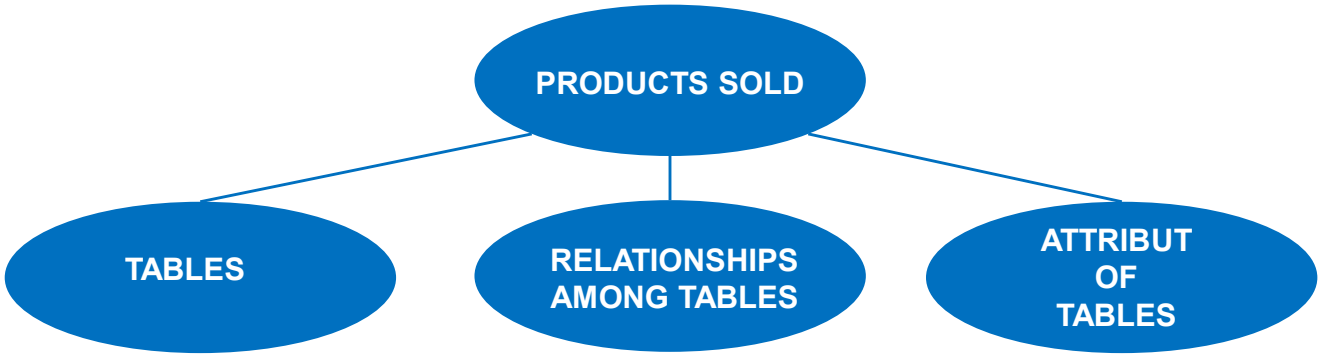
- **CONCEPTUAL**: this stage defines what tables the system contains, and it identifies the highest-level relationships between the different tables.
- **LOGICAL**: this stage describes the data in as much detail as possible, without regard to how they will be physical implemented in the RDBMS.
- **PHYSICAL**: this phase describes how the model will be implemented in the RDBMS.

29

Data model in a SQL-database

EXAMPLE

We want to build a relational database for a company like **MERCADONA**, this database will focus on storing the data about the products sold.



30

CONCEPTUAL stage

This stage defines what tables the system contains, and it identifies the highest-level relationships between the different tables.

TABLES:

- **Time**: information about whenever a product is sold
- **Product**: information about what is the product sold
- **Sales**: information about a sale of a product
- **Store**: information about wherever the sale is made

RELATIONSHIPS:

- **Time - Sales**: a sale is made at a given moment
- **Product - Sales**: what product is sold
- **Store - Sales:** where is the sale has been made

31

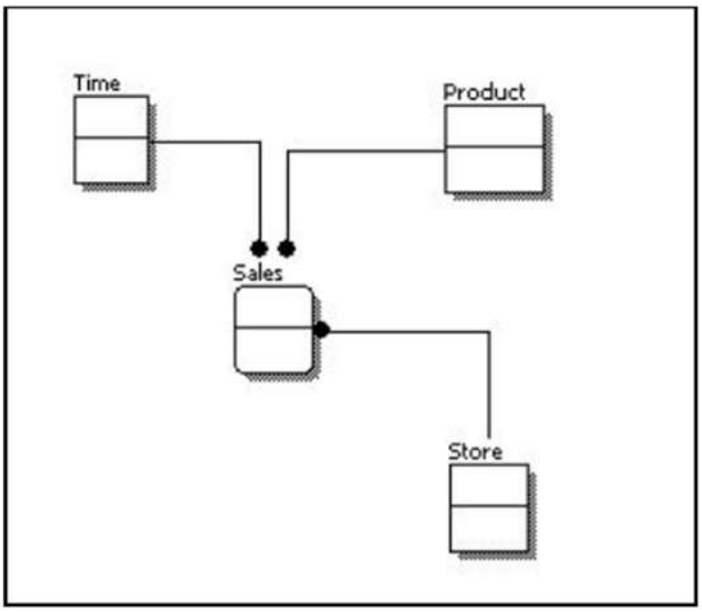
CONCEPTUAL stage

TABLES:

- **Time**:
- **Product**:
- **Sales**:
- **Store**:

RELATIONSHIPS:

- **Time - Sales**:
- **Product - Sales**:
- **Store - Sales:**



32

LOGICAL stage

TIME:

- **Date (dd/mm/yyyy hh:mm)**
- **Date Description:**
- **Month**
- **Month Description**
- **Year**
- **Week**
- **Week Description**

PRODUCT

- **Product ID**
- **Product Description**
- **Category**
- **Category Description**
- **Unit Price**
- **Created**

STORE:

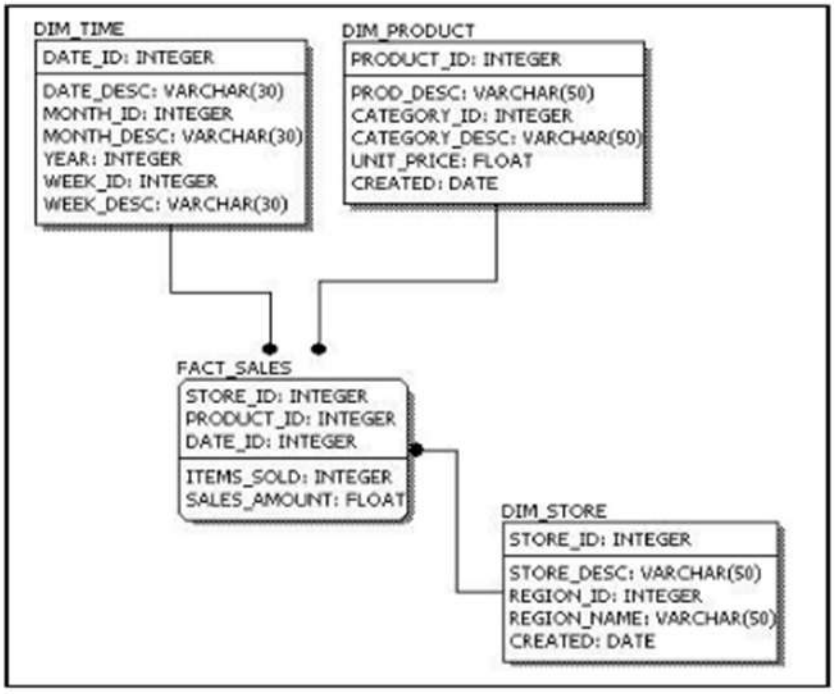
- **Store ID**
- **Store Description**
- **Region**
- **Region Description**
- **Created**

SALES:

- **Product ID**
- **Store ID**
- **Date**
- **Item Sold:**
- **Sales Amount**

33

PHYSICAL stage



34

1º Examples

Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

35

2º Examples- solution A

Candidate key	Candidate key	Candidate key	Candidate key		
Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

36

1º Examples- solution A

Primary key

Candidate key | Candidate key | Candidate key | Candidate key

Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

37

2º Examples- solution A

Primary key

Candidate key | Candidate key | Candidate key | Candidate key

Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

38

2º Examples- solution C

Primary key

Candidate key | Candidate key | Candidate key | Candidate key

Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

39

1º Example

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

40

2º Examples-solution

Candidate key | Candidate key

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

41

2º Examples-solution

Primary key

Candidate key | Candidate key

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

42

3º Examples

employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep

43

3º Examples-solution

CK CK CK CK

employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep

44

3º Examples-solution

Primary key

CK **CK** **CK** **CK**

employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep

45

4º Examples

Month	Name	Gender	Diagnosis	Treatment
May	Jessica	F	Allergy	Eye Drops
May	Sam	M	Allergy	Eye Drops
May	Wes	M	Cataract	Cataract Surgery
May	Rachel	F	Ptergyium	Eye Drops
May	Lily	F	Allergy	Eye Drops
May	Hannah	F	Cataract	Cataract Surgery
May	Denise	F	Allergy	Eye Drops
May	Sharon	F	Allergy	Eye Drops
May	Robin	F	Allergy	Eye Drops
May	Lianna	F	Ptergyium	Eye Drops
May	Thomas	M	Presbyopia	Reading Glasses
May	Kimberly	F	Refractive Error	Distance Glasses
May	Michael	M	Refractive Error	Distance Glasses
May	Jacob	M	Conjunctivitis	Eye Drops
June	John	M	Presbyopia	Reading Glasses
June	Tim	M	Refractive Error	Distance Glasses
June	Allison	F	Cataract	Cataract Surgery
June	Laura	F	Ptergyium	Eye Drops
June	Scott	M	Cataract	Cataract Surgery
June	Sarah	F	Ptergyium	Eye Drops
June	Alex	M	Ptergyium	Eye Drops
June	Robert	M	Cataract	Cataract Surgery

46

ID REGISTER TREATMENT

In this case, due to none of attributes can be use as primary key since they do not meet the condition of uniquely identifying each row of the table.

Month	Name	Gender	Diagnosis	Treatment
May	Jessica	F	Allergy	Eye Drops
May	Sam	M	Allergy	Eye Drops
May	Wes	M	Cataract	Cataract Surgery
May	Rachel	F	Ptergyium	Eye Drops
May	Lily	F	Allergy	Eye Drops
May	Hannah	F	Cataract	Cataract Surgery
May	Denise	F	Allergy	Eye Drops
May	Sharon	F	Allergy	Eye Drops
May	Robin	F	Allergy	Eye Drops
May	Lianna	F	Ptergyium	Eye Drops
May	Thomas	M	Presbyopia	Reading Glasses
May	Kimberly	F	Refractive Error	Distance Glasses
May	Michael	M	Refractive Error	Distance Glasses
May	Jacob	M	Conjunctivitis	Eye Drops
June	John	M	Presbyopia	Reading Glasses
June	Tim	M	Refractive Error	Distance Glasses
June	Allison	F	Cataract	Cataract Surgery
June	Laura	F	Ptergyium	Eye Drops
June	Scott	M	Cataract	Cataract Surgery
June	Sarah	F	Ptergyium	Eye Drops
June	Alex	M	Ptergyium	Eye Drops
June	Robert	M	Cataract	Cataract Surgery

4º Examples - solution

47

Conceptual and Logical design

Contents

1 Entity-Relationship model (ERm) **1**

2 Conceptual stage **5**
2.1 Entity identification 5
2.2 Relationship identification 6

3 Logical stage **6**
3.1 Field identification 6
3.2 Cardinality identification 7

4 Functional dependency **8**
4.1 Functional Dependency (FD) 8
4.2 Partial Functional Dependency (PDF) 9
4.3 Full Functional Dependency (FFD) 10

5 Standardization: a normalization process in a database **11**
5.1 First Normal From (1NF) 12
5.2 Second Normal From (2NF) 14
5.3 Third Normal From (3NF) 15

1 Entity-Relationship model (ERm)

Before starting with the three different phase to design a Relation Database, it is necessary to explain the tool called **Entity-Relationship model (ERm)**. This tool allows us to represent, in a simplified and schematic way, the interrelationship among different elements in a specific domain of knowledge. The domains of knowledge can range from **business processes** to **organizational structure of a company**, as well as **describing hardware infrastructure** or, as in our case, **database structure**.

One of the challenges faced when designing a database is the fact that designers, developers, and end-users tend to view data and its usage differently. If this situation is left unaddressed, we may end up producing a database system that does not meet user requirements.

Thus, this tool allows us to identify, describe and related the different components in a relational database (tables and relationships) that are imposed by business needs. At the same time, this tool works like a communication tool understood by all stakeholders (technical as well as non-technical users), allowing to design a database systems that meet the requirements of all users.

ERm is a tool used for describing a **Relational Model** to develop and implement a **Relational Database** in a **RDBMS** (see Fig.1).

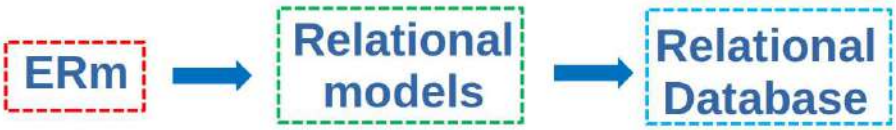


Figure 1: The different steps to obtain a relational database starting with ERm.

ERm has three principal elements:

- **Entities**: in our case the entities are the relations (tables) of a Relational database. Entities are the basic object of ERm.
- **Fields**: in our case the fields are the attributes of each relation in a Relational database.
- **Relationships**: they are the different associations (links) among entities. Relationships are described by their main characteristic, **cardinality**.

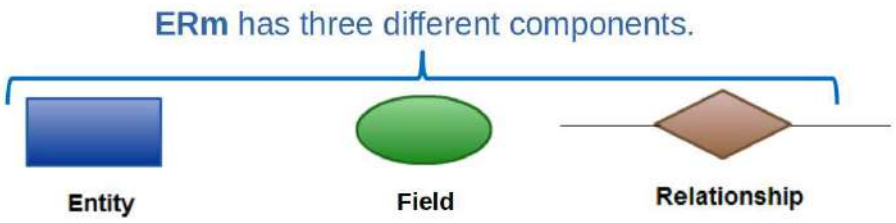


Figure 2: Way of representing the different elements in the ERm.

Remember that in **unit 3** we saw three types of cardinality, now it will be seen in more detail with different examples using ERm:

– **One to One**: a row of a tables is related to a row of another tables.

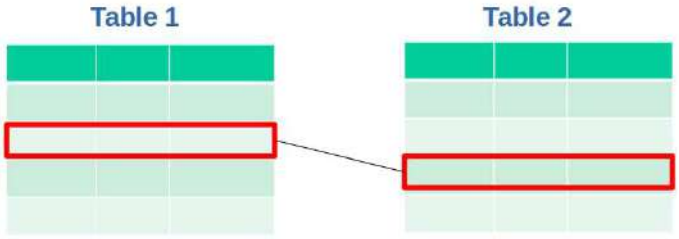


Figure 3: Cardinality **One to One**.

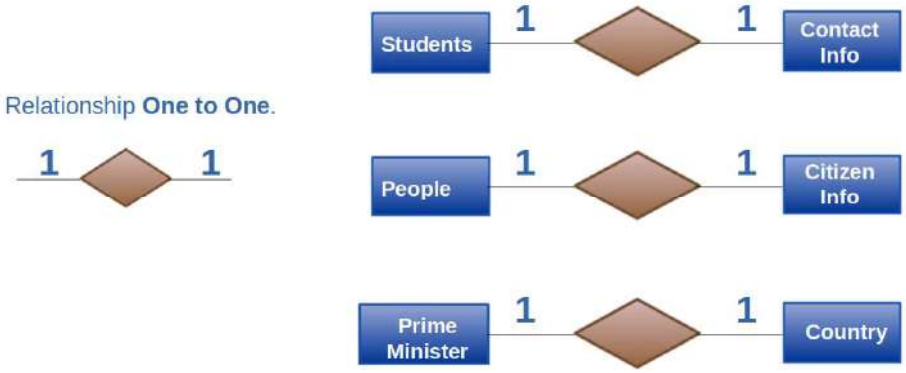


Figure 4: Cardinality **One to One** using ERm.

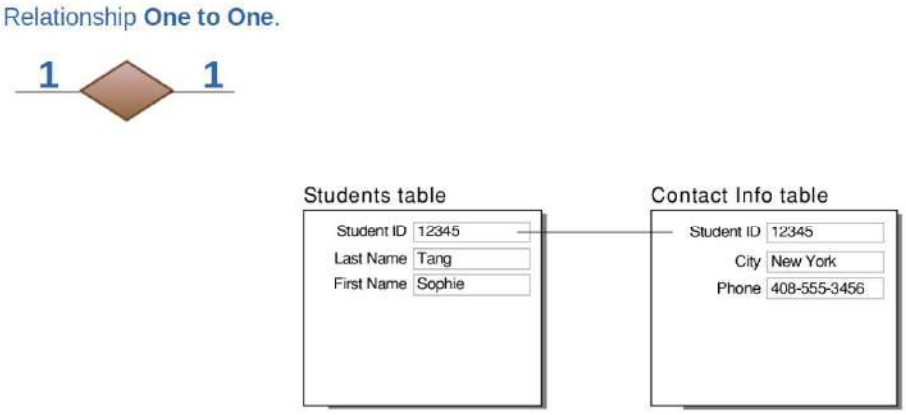


Figure 5: Cardinality **One to One** with a real example.

– **One to Many (or Many to One)**: one row in one table is related to many rows in another table and vice versa.

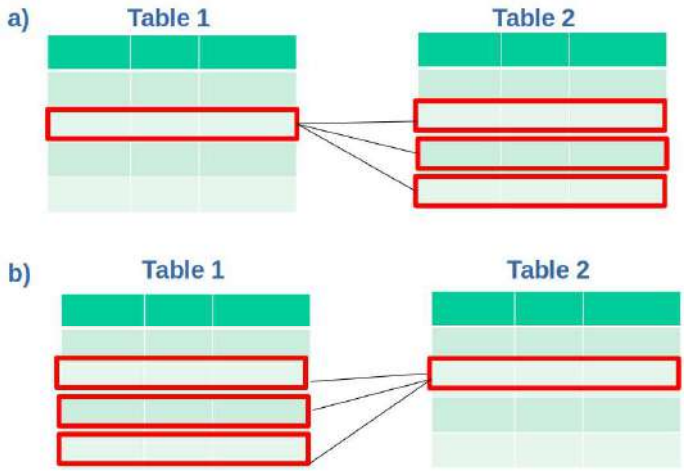


Figure 6: Cardinality: a) **One to Many** and b) **Many to One**.

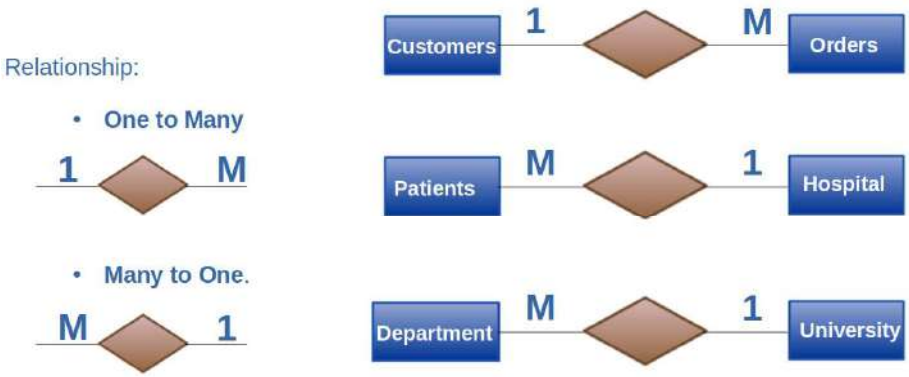


Figure 7: Cardinality **One to Many** using ERm.

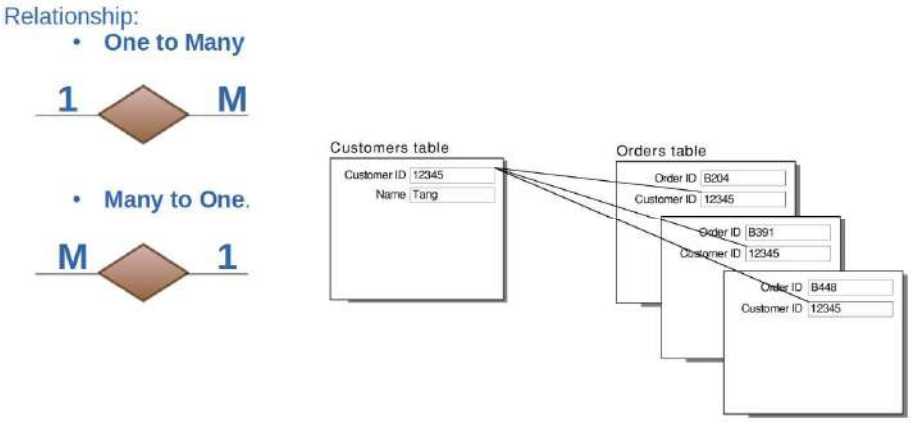


Figure 8: Cardinality **One to Many** with a real example.

– **Many to Many**:many rows in one table are related to many rows in another table.

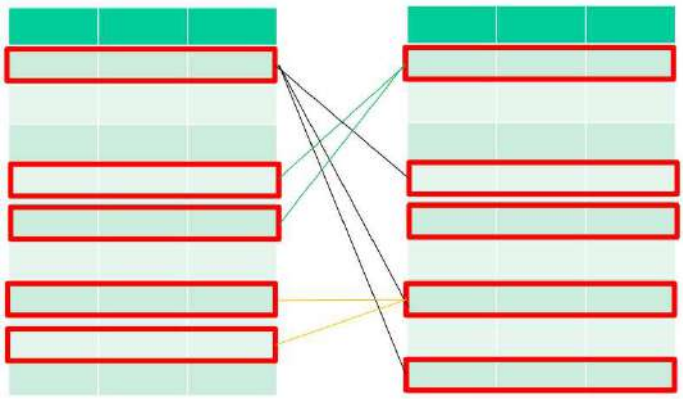


Figure 9: Cardinality **Many to Many (M to N)**.

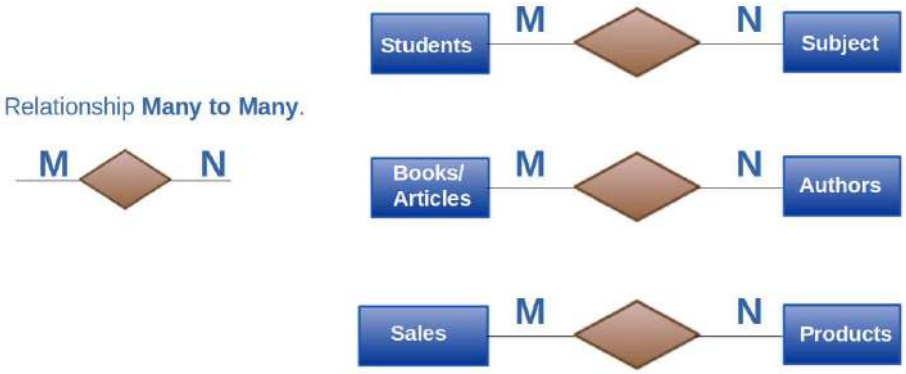


Figure 10: Cardinality **M to N** using ERm.

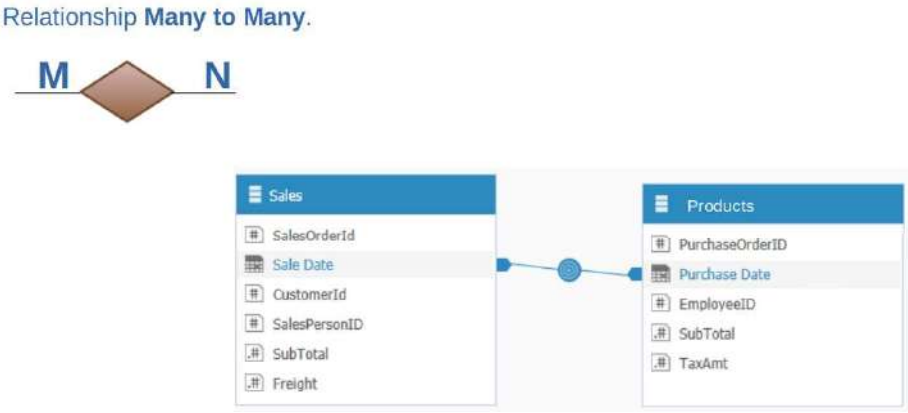


Figure 11: Cardinality **M to N** with a real example.

Figure 12 shows an entity-relationship diagram , where **EA** and **EB** are the entities (boxes), **R** is the relationship (diamond) and **Attr.1** and so on are the attributes (ovals).

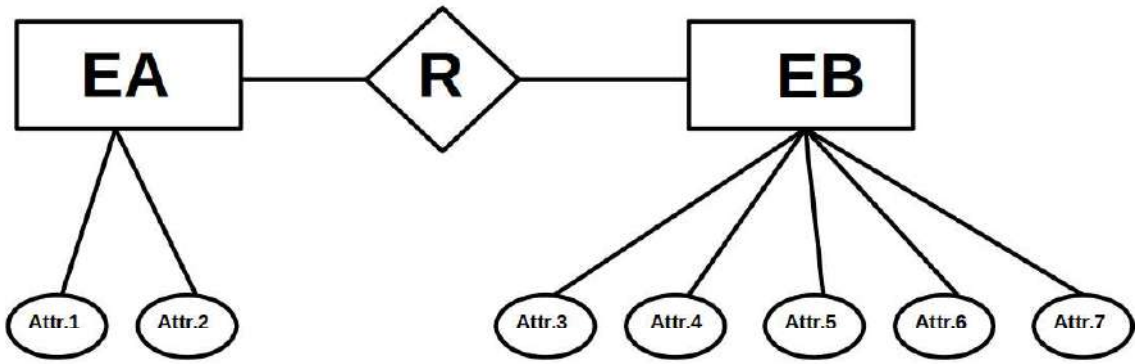


Figure 12: Example of ERm diagram.

2 Conceptual stage

As it was said previously, in the stage of **CONCEPTUAL design** it is defined what entities the system contains and it identifies the highest-level relationships between the different entities, always according to the business' needs. Thus, in this stage and using the tool ERm, we need to identify:

- **ENTITIES**
- **RELATIONSHIPS** (without considering the cardinality)

To consolidate and clarify the different concepts that will be used in the three design phases of a relational database, we will carry out the following example:

> **The design and implementation of a small database of a University, this database will focus on storing the data about the students and the courser where the student are enrolled.**

2.1 Entity identification

The first step is to identify the set of entities (relations or tables) in our database. An entity may be defined as a thing that can be uniquely identified. When we speak of an entity, we normally speak of some aspect of the real world that can be distinguished from other aspects of the real world.

Thus, an entity is a thing that exists either physically or logically. In our example, an entity may be a physical object such as:

- **Students**
- **Courses**

Figure 13: ERm diagram with the entities considered in our database.

2.2 Relationship identification

The second step is to identify the set of relationships more relevant among entities (tables) in our database, always according to business' needs. In our example and due to the simplicity of our database, we only have a logical relationship, which we will call **Enrolment**.

Figure 14: ERm diagram with the relationships between considered entities in our database.

3 Logical stage

As it was said previously in the stage of **LOGICAL design** the entities and relationships identified above are described in more detail, but still without regard to how they will be physical implemented in the RDBMS. Thus, in this stage and using the tool ERm, we need to identify:

- **Fields**
- **Primary key**
- **Cardinality**

3.1 Field identification

The fields in our case are the attributes of each relation. Thus, the attributes are the characteristics that define or identify a relation (or table). Each considered table needs as many attributes as are necessary to store the data correctly. It is also necessary to indicate the primary key for every relation identified.

- Fields of **Students** entity:
  - **Roll_no**
  - Student_name
  - Age

  - Course_id

- Fields of **Courses** entity:
  - **Course_id**
  - Course_name
  - Duration

The primary keys for each table are indicated in bold. Remember that the primary keys are chosen to uniquely identify the rows in a table. These keys can be chosen among the attributes or to add a specifically designed attribute for this purpose. In our case, the two keys have been designed creating a specific ID number: **Roll_no** and **Course_id**, since the other attributes do not meet the condition of uniquely identifying of the rows in a table.

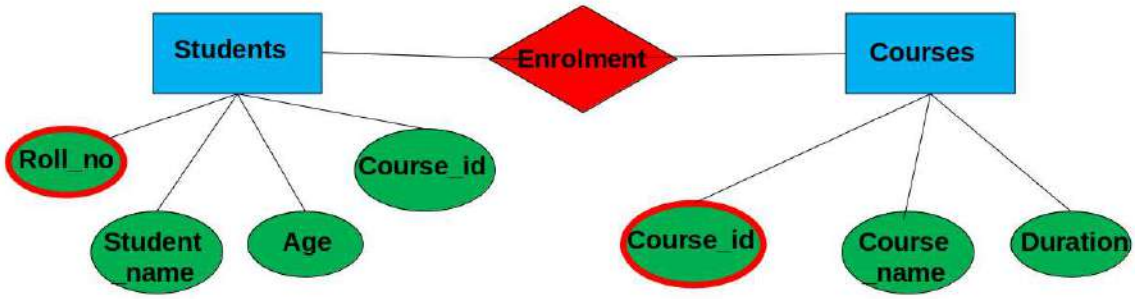


Figure 15: ERm diagram with the considered attributes in our database. Ovals with red lines indicate the primary keys for each entity.

3.2 Cardinality identification

The cardinality is a property of relationships which indicates how the rows of two tables are related to each other. Previously, we have see several kind of cardinalities: **One to One**, **One to Many** and **Many to Many**. Specifically in our case, the cardinality is of the type many to many **Many to Many (M to N)**, since a student can be enrolled in several courses and a course can have several student enrolled.

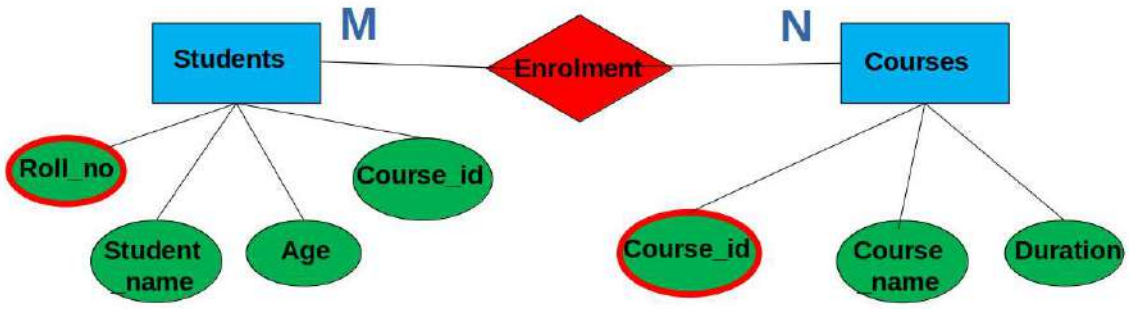


Figure 16: ERm diagram with the considered attributes in our database. Ovals with red lines indicate the primary keys for each entity.

4 Functional dependency

A **dependency**, in a broad sense, is a constrain between two sets of objects. In the case of Relational Database, these constrains are imposed limitations on the relationships between two sets of attributes. This will help us to structure and relate the attributes in a database. For the purpose of this course, we are going to see three different kind of dependencies:

- **Functional Dependency (FD)**
- **Partial Functional Dependency (PDF)**
- **Full Functional Dependency (FFD)**

4.1 Functional Dependency (FD)

It is a term derived from the mathematical theory that underpins relational database theory. It concerns the dependency of the values of one set of attributes on the values of another set of attributes. A formal definition of **functional dependency (FD)** is as follows:

*A set of attributes **Y** is functionally dependent on a set of attributes **X** if a given set of values for each attribute in **X** determine unique values for the set of attributes in **Y**. Mathematically, a dependence is written like:*

$$X \to Y$$

For instance, consider the following relation:

STUDENT

Student ID	First Name	Last Name
000348282	Tirilee	Lytler
234556677	Koons	Smith
777349993	John	Doe
800005555	Patrick	Saint
999001111	James	Kirkland
999009800	Mary	Smith
999110000	Sally	Jones
666110001	Bill	Mack
555119800	Greg	Johnson
444339837	Jacob	Schwartz
333448887	Jenny	Donald
222551288	Nancy	Ogur

Figure 17

To simplify the table can be expressed in a shorter way like:

STUDENTS = {Student ID, First Name, Last Name}

We may state that the set of attributes **{First Name, Last Name}** is functionally dependent on the attribute **{Student ID}**. This means that, given a value for **Student ID**, we can always uniquely determine the value of **First Name** and the value of **Last Name**. Note that, for this relation, the opposite would not be true. For example, if there are three students with the same **First Name** and **Last Name**, we will get a list of three student IDs, so we cannot uniquely determine a value for **Student ID** given values of the attributes **{First Name, Last Name}**.

According to the primary key definition, the attributes in a table are functionally dependent on the primary key, since given a primary key value, the values of the table's attributes are uniquely determined.

4.2 Partial Functional Dependency (PDF)

Let's consider the following examples where the table called **ASSING** (Table 1).

Table 1: ASSIGN

Person-ID	**Poject-Budget**	**Project**	**Time-spent-by-person-on-project**
S75	32	P1	7
S75	40	P2	3
S79	32	P1	4
S79	27	P3	1
S80	40	P2	5

To simplify the table can be expressed in a shorter way like:

ASSIGN = {Person-ID, Project-Budget, Project,
Time-spent-by-person-on-project }

We may state, as we have seen previously, that the set of attributes **{Project}** is functionally dependent on the attributes **{Person-ID, Project-Budget, Time-spent-by-person-on-project}**. If we take a closer look, in this case it is possible to find a subset of **{Person-ID, Project-Budget, Time-spent-by-person-on-project}** that can determine uniquely **{Project}**. Concretely this subset is **{Person-ID, Project-Budget}** and it is easy to check that **{Person-ID, Project-Budget}** holds the functional dependency of **{Project}**.

A formal definition of**partial functional dependency (PFD)** is as follows:

A functional dependency $\boldsymbol{X \rightarrow Y}$ *is called partial functional dependent if some attribute of* $\boldsymbol{X}$ *can be removed from* $\boldsymbol{X}$ *and the functional dependency is still hold.*

4.3 Full Functional Dependency (FFD)

A formal definition of **full functional dependency (FFD)** is as follows:

Y *is full functional dependent on a set of attributes* ***X*** *if* ***Y*** *is functional dependent on* ***X*** *and it is not possible to find another subset of* ***X*** *which holds that functional dependency.*

Considering the previous case saw, we have that **{Project}** is full functional dependent on **{Person-ID, Project-Budget}** because it is not possible to determine uniquely values of **{Project}** with just values of **{Project-Budget}** or **{Project-Budget}**. The term **full functional dependency** is used to describe the minimum set of attributes to hold a **FD**. Another example of **FFD** is shown in the following Fig.18 , where it is easy to check that:

{First Name, Last Name} is full functional dependent on **{Student ID}**.

STUDENT

Student ID	First Name	Last Name
000348282	Tirilee	Lytler
234556677	Koons	Smith
777349993	John	Doe
800005555	Patrick	Saint
999001111	James	Kirkland
999009800	Mary	Smith
999110000	Sally	Jones
666110001	Bill	Mack
555119800	Greg	Johnson
444339837	Jacob	Schwartz
333448887	Jenny	Donald
222551288	Nancy	Ogur

Figure 18

5 Standardization: a normalization process in a database

STANDARDIZATION (or also called **NORMALIZATION PROCESS**) is a part of **Logical stage** for the correct design of a database. This process of standardization or normalization is part of the successful database design. Without normalization, database systems can be inaccurate, slow, and inefficient and they might provide the performance you do not expect.

The standardization or normalization process has a set of rules called (**NORMALIZATION RULES**). These rules ensure that the way in which the data has been structured for the storage of data (entities, attributes and relationships chosen) does not cause problems such as inconsistency or lack of logic. In a relational database, a logical and efficient design is crucial.**A poorly design database may provide important problems of data management**.

Most of these problems are the result of two bad design features called:

- **redundant data**: unnecessarily repeated data.
- **anomalies**: any incident due to the irregularities and inconsistencies in the storage can be affect to the data integrity.

Basically, standardization or normalization is the process of efficiently organising data in a database, being the objectives of this process the followings:

- arrange data into logical groups (each group describes a small part of the whole).
- minimize the amount of duplicat data necessary.
- build a database that allows a quick access and manipulation of data.
- ensure data dependencies make sense.

Standardization or normalization process just adjusts or "tunes" the database for an optimize performance of this one, reducing the amount of space a database consumes, ensuring that data is logically stored and quickly accessible. Therefore, in the process of designing a relational database the finality of the design is to structure the data in a way that:

- eliminates unnecessary duplications
- provides a rapid search path to all necessary information
- specify and define tables, keys, columns and relationships in order to create an efficient database.

The normalization process involves several steps called **FORMS**. There are **BASIC FORMS**, which are:

- **1NF** (First Normal Form)
- **2NF** (Second Normal Form)
- **3NF** (Third Normal Form)
- **4NF** (Fourth Normal Form)
- **5NF** (Fifth normal Form)

- **6NF** (Sixth Normal Form)

, and on the other hand, there are other several forms called **HIGH-LEVEL FORMS**. Below we can see some of them:

- **EKNF** (Elementary Key Normal Form)
- **ETNF** (Essential Tuple Normal Form)
- **DKNF** (Domain Key Normal Form)
- ...

For the purpose of the course, we will see the first 3 forms, **1NF**, **2NF** and **3NF**, which are the most common.

5.1 First Normal From (1NF)

The basic idea of **First Normal Form** is:

All rows have to have UNIQUE and ATOMIC values.

A database meets **1NF** if:

- No duplicate columns
- Each cell must only contain a single values (atomic value, no a list of values)
- Each value should not be divisible

We will see several example to show what means to meet **1NF**.

EXAMPLE A

The table of Fig.19 does not meet 1NF because it has duplicate columns. We can solve this problem removing one of duplicate columns.

STUDENT

Student ID	First Name	Last Name	Last Name
000348282	Tirilee	Lytler	Lytler
234556677	Koons	Smith	Smith
777349993	John	Doe	Doe
800005555	Patrick	Saint	Saint
999001111	James	Kirkland	Kirkland
999009800	Mary	Smith	Smith
999110000	Sally	Jones	Jones
666110001	Bill	Mack	Mack
555119800	Greg	Johnson	Johnson
444339837	Jacob	Schwartz	Schwartz
333448887	Jenny	Donald	Donald
222551288	Nancy	Ogur	Ogur

Figure 19

EXAMPLE B

The table of Fig.20 does not meet 1NF because there is no a single value in each cell (no Atomic data). We can solve the problem splitting the information off and creating different tables with this information.

STUDENTS AND CLASSES (MODULES) DETAILS TABLE
Brian Smith, NBC Services, London, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Laura Grey, ABC Corporation, Nottingham, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000
John Brown, NBC Services, London, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Peter Black, ABC Corporation, Nottingham, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000

Figure 20

EXAMPLE C

The table of Fig.21 does not meet 1NF because there is information that can be splitted down further. We can solve this problem adding other column "Second Employee Contac No" or removing one of the numbers.

EMPLOYEE NAME	EMPLOYEE ID	EMPLOYEE LOCATION	EMPLOYEE CONTACT NO
AJAY	12455	DELHI	987565,356212
AMIT	89752	AGRA	455145,988965
AKSHAR	23654	HARYANA	987454
ALOKE	98765	KOCHI	124787

Figure 21

5.2 Second Normal From (2NF)

The basic idea of **Second Normal Form** is:

No partial dependencies.

A database meets **2NF** if:

- Database must be in 1NF.
- The attribute should not be fully functional dependent on the candidate keys, that is, eliminate any functional dependence on part of any candidate key.

EXAMPLE

We will see an example to show what means to meet **2NF**.

The tables of Fig.22 does not meet 2NF because the primary key of this tables is **Model full name** (red box), but we can see that the attribute **Manufacture country** dependent on a proper subset of attributes **Manufacture** and **Model**.

Figure 22

The solution is two create another table with that another dependency (the primary keys are indicated in red box) (see Fig.23).

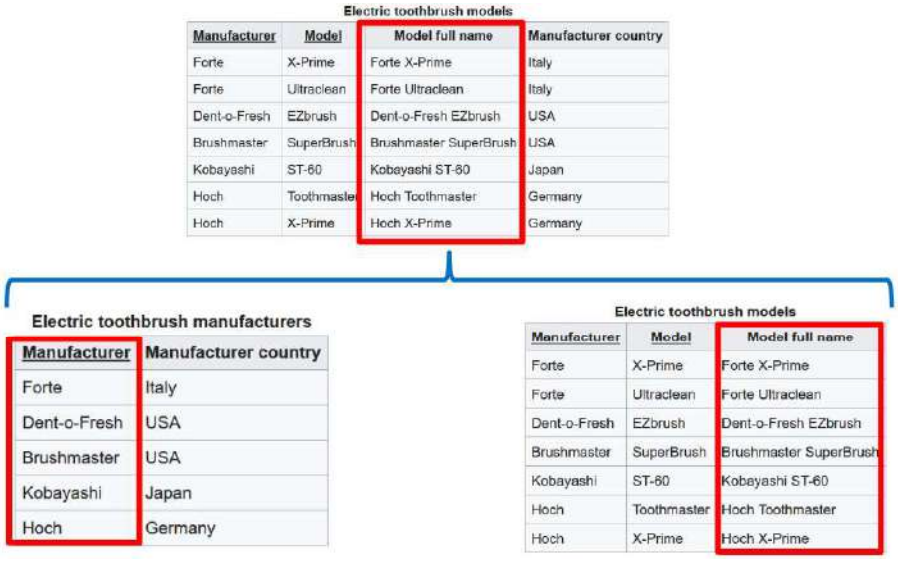


Figure 23

5.3 Third Normal From (3NF)

The basic idea of **Third Normal Form** is:

No transitive dependencies.

A database meets **3NF** if:

- Database must be in 1NF and 2NF.
- The attributes must only be determinable by the primary key and not by other attributes.

We will see an example to show what means to meet **3NF**.

The tables of Fig.24 does not meet 3NF because we can see a transitive dependency among **ID**, **Bank Code No** and **Bank**. This is not desirable since someone who is updating the database may change the name of **Bank** but may forget updating **Banck Code No**.

ID	Name	Balance	Bank Code No	Bank
5486546547	Jose Clapton	300000	Kjhasdcnlkn	Lenber Bank
6356865153	Erik Entrena	500000	LÑMkjhskldhl	Leamberber Bank
5651635856	Andrew Jackson	1000000	Lknalcnklñkna	Savana Bank
54656616354	Martin Denver	150000	AlkjdJADÑLJ	Bank of New Jersey
63560000053	Erik Entrena	500000	Qañajslashjcc	Trigger Bank
54656618884	Martin Denver	300000	Lknalcnln	Bank of New York
111110000053	Erik Entrena	1000	Zzzpojñlmsadn	Bank of Poland

Figure 24

Again, the solution is to split the table off, but in a correct way, without losing of any kind of information (see Fig.25).

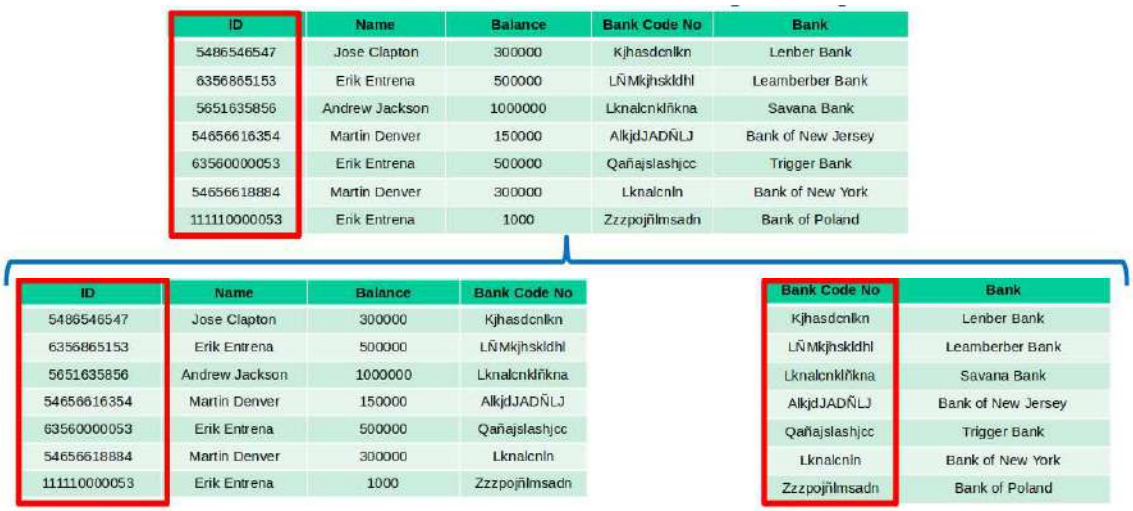


Figure 25

1

Unit 3 – Conceptual and Logical stage

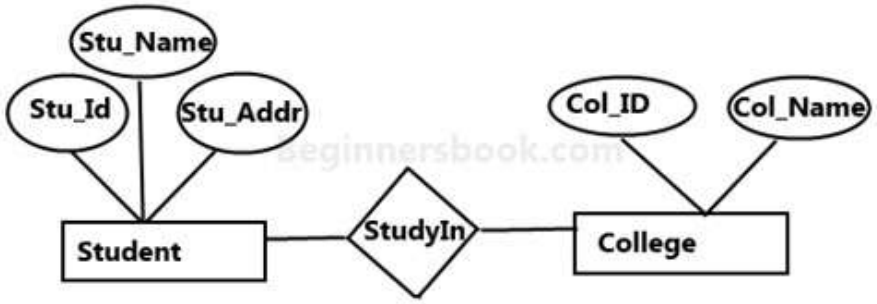
- Entity-Relationship model (ERm)
- Conceptual stage
  - Entity identification
  - Relationship identification
- Logical stage
  - Identification of fields (attributes)
  - Identification of primary keys
  - Identification of relationship properties (cardinality)
  - Standardization
  - Functional dependence

2

Entity-Relationship model (ERm)

ERm is a simple tool allows us to describes, in a simplified and schematic ways, the interrelationship among different elements in a specific domain of knowledge.

These domains of knowledge range from **business processes** to **organizational structure of a company,** as well as **describing hardware infrastructure** or **database structure**. The application domains of this tool are quite wide.



Example of **ERm scheme** of a **Relational Model** for a **Relational Database**.

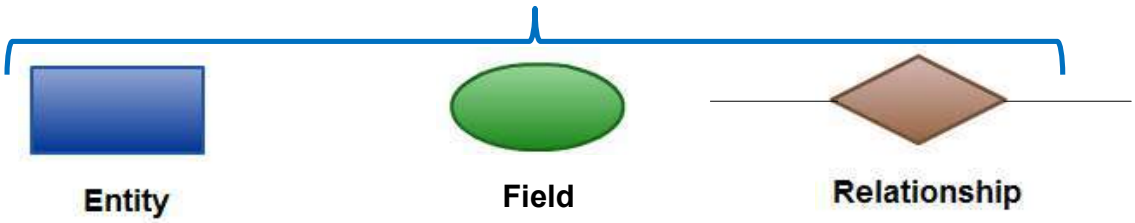
Entity-Relationship model (ERm)

ERm → **Relational models** → **Relational Database**

ERm is a tool used for describing a **Relational Model** to develop and implement a **Relational Database** in a **RDBM**.

Entity-Relationship model (ERm)

ERm have three different components.



Entity-Relationship model (ERm)

ENTITIES: They are the relations or tables.

FIELDS: they are the characteristic of an entity, they are the attributes of a table.

RELATIONSHIPS: they are the different associations (links) among entities (tables).

This tool, ERm, is only used in CONCEPTUAL and LOGICLA stages.

Different kinds of Relationships

Relationships have a characteristic called **CARDINALITY**. The cardinality indicates how the rows of different tables are related among them when a relationship exists between tables.

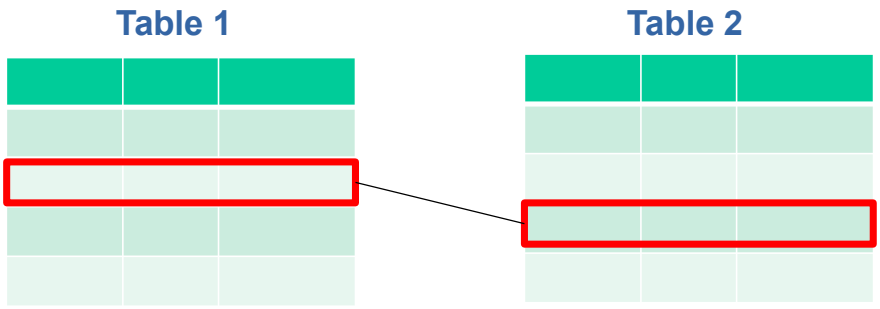
We are going to see 3 different types of cardinalities:

- **One to One**
- **Many to One** (or **One to Many**)
- **Many to Many**

IMPORTANT!!!!
Do not confuse this CARDINALITY with the CARDINALITY of a table, they are different.

Different kinds of Relationship

Relationship **One to One**.



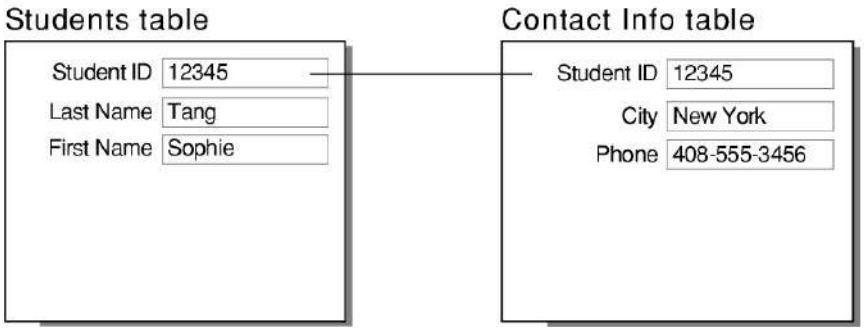
Different kinds of Relationship

Relationship **One to One**.

9

Different kinds of Relationship

Relationship **One to One**.



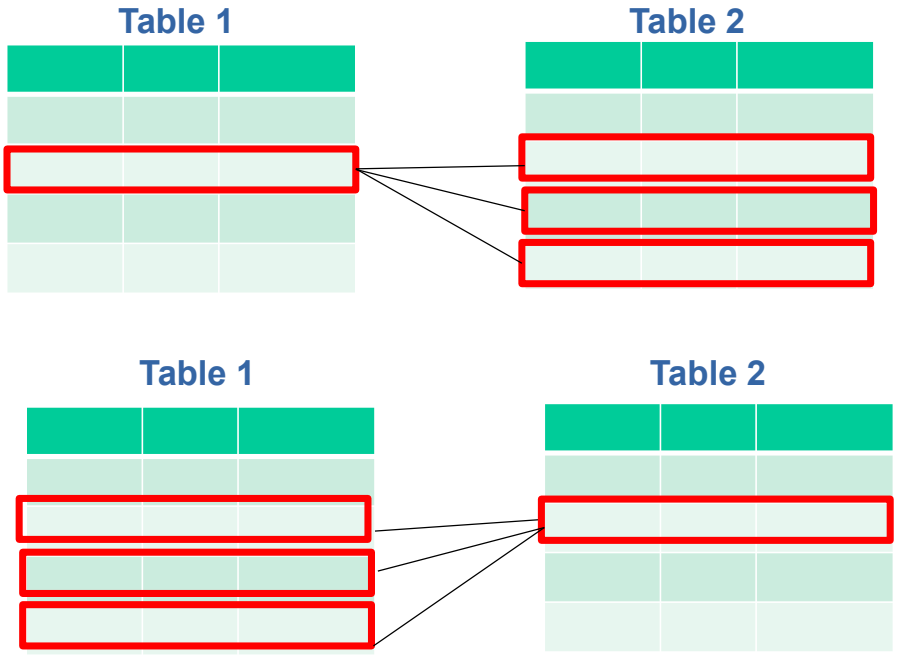
10

Different kinds of Relationship

Relationship:

- One to Many

- Many to One.



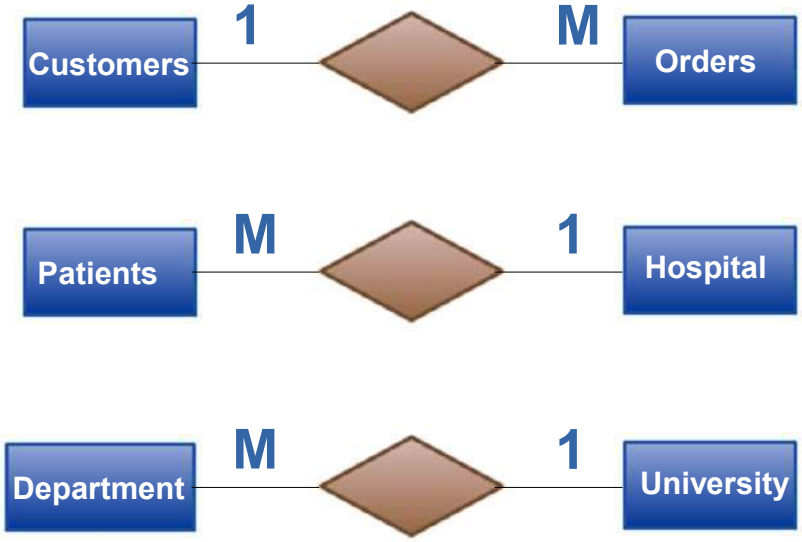
11

Different kinds of Relationship

Relationship:

- One to Many

- Many to One.



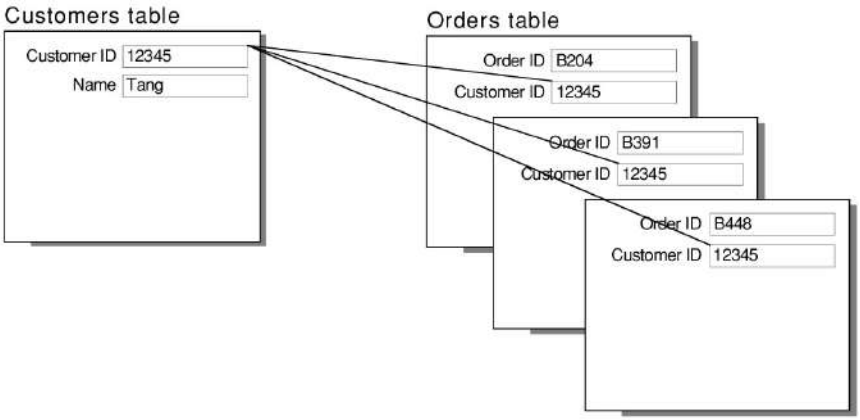
12

Different kinds of Relationship

Relationship:

- **One to Many**

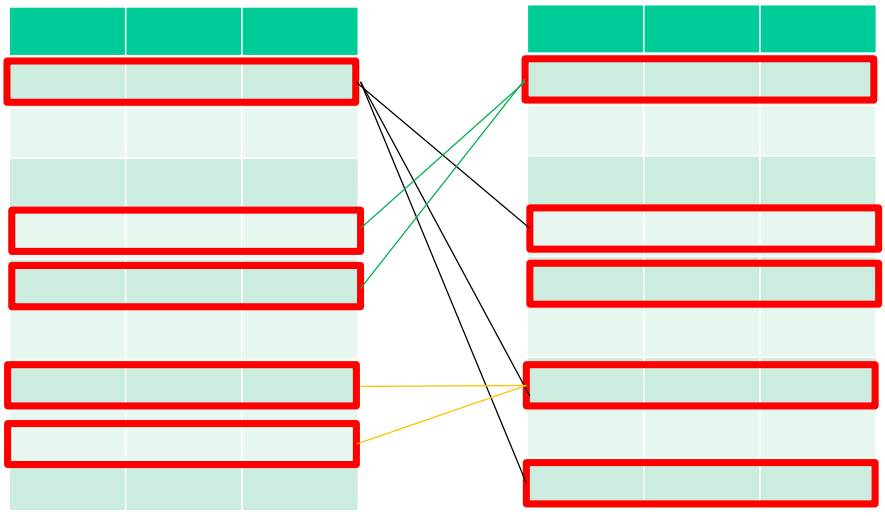
- **Many to One**.



13

Different kinds of Relationship

Relationship **Many to Many**.



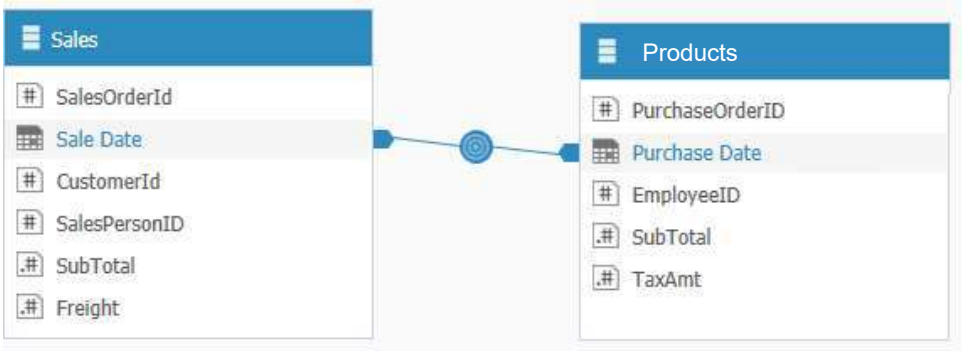
14

Different kinds of Relationship

Relationship **Many to Many**.

Different kinds of Relationship

Relationship **Many to Many**.



CONCEPTUAL stage

In this stage we identify what tables the database contains, and it identifies the highest-level relationships between the different tables.

Therefore, using the tool **ERm** we need to identify:

- **ENTITIES**
- **RELATIONSHIPS** (without consider the cardinality)

EXAMPLE - 1

The design and implementation of a small database of a University, this database will focus on storing the data about the students and the courser where the student are enrolled.

EXAMPLE - 1

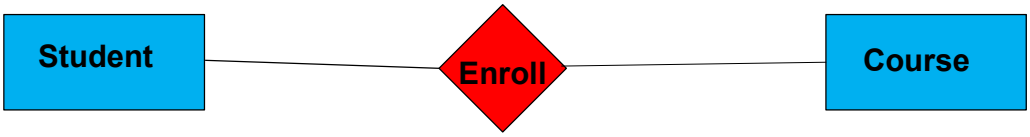
ENTITIES identification:

- **Students**
- **Course**

EXAMPLE - 1

RELATIONSHIPS identification:

- **Enroll**



LOGICAL stage

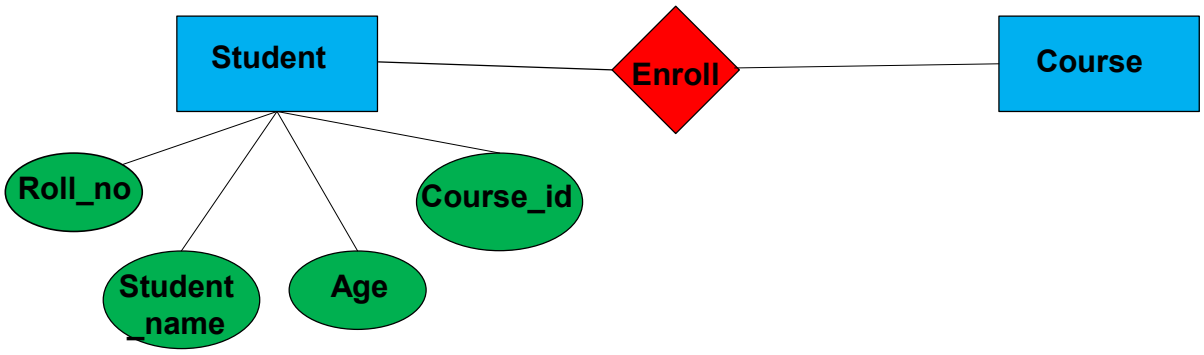
This stage describes the data in as much detail as possible: attributes, primary key and cardinality.

Therefore, using the tool **ERm** we need to identify:

- **FIELDS** (attributes, indicating the primary key)
- **RELATIONSHIPS** (indicating the cardinality of the relationships)

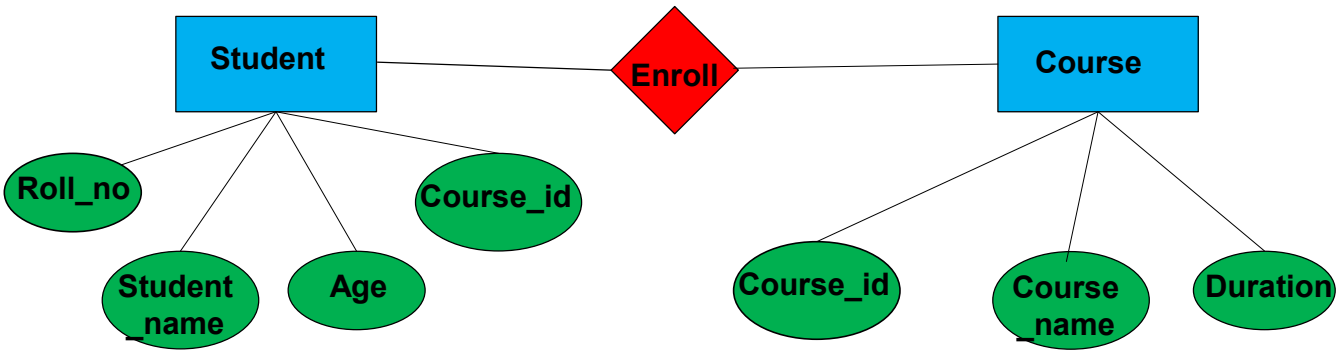
EXAMPLE - 1

Field identification.



EXAMPLE - 1

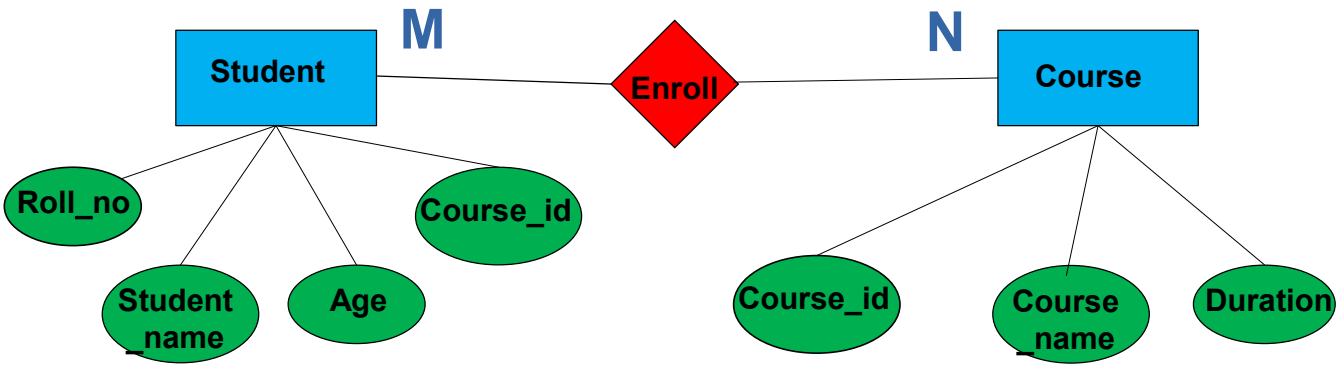
Field identification.



23

EXAMPLE - 1

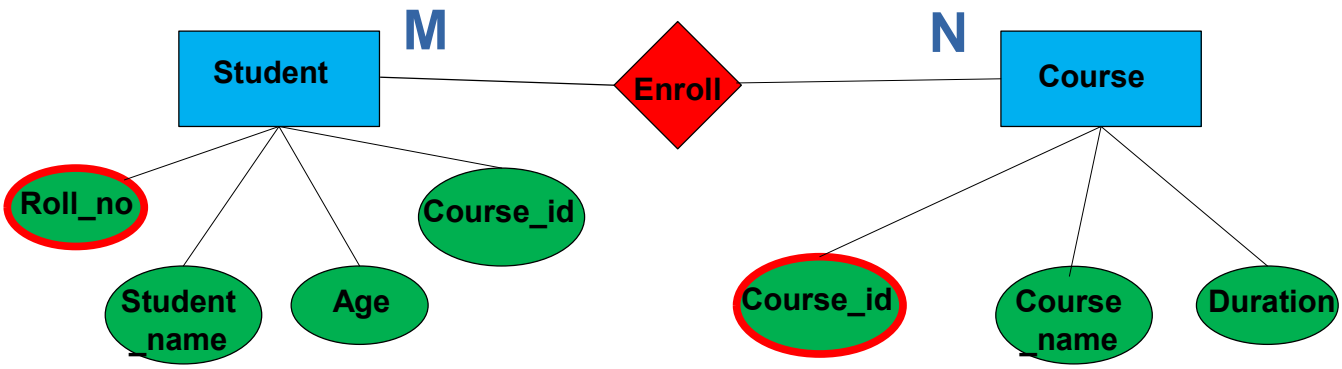
Indicate the cardinality of different relationships.



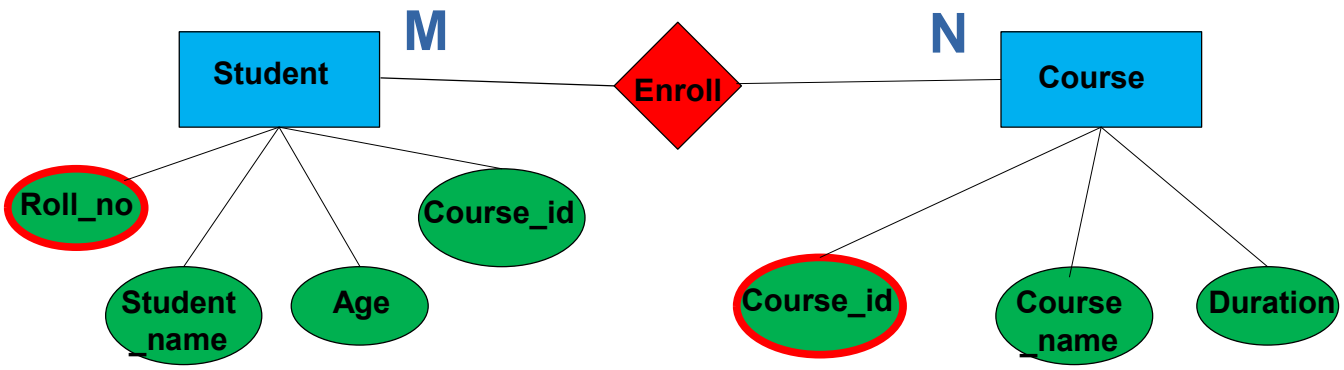
24

EXAMPLE - 1

Indicate the primary key in different entities.



EXAMPLE - 1



Enroll

Student

Roll_no	Student_name	Age	Course_id
1	Andrew	18	78
2	Angel	19	78
3	Priya	20	56
4	Analisa	21	56

Primary Key

Foreign Key

Course

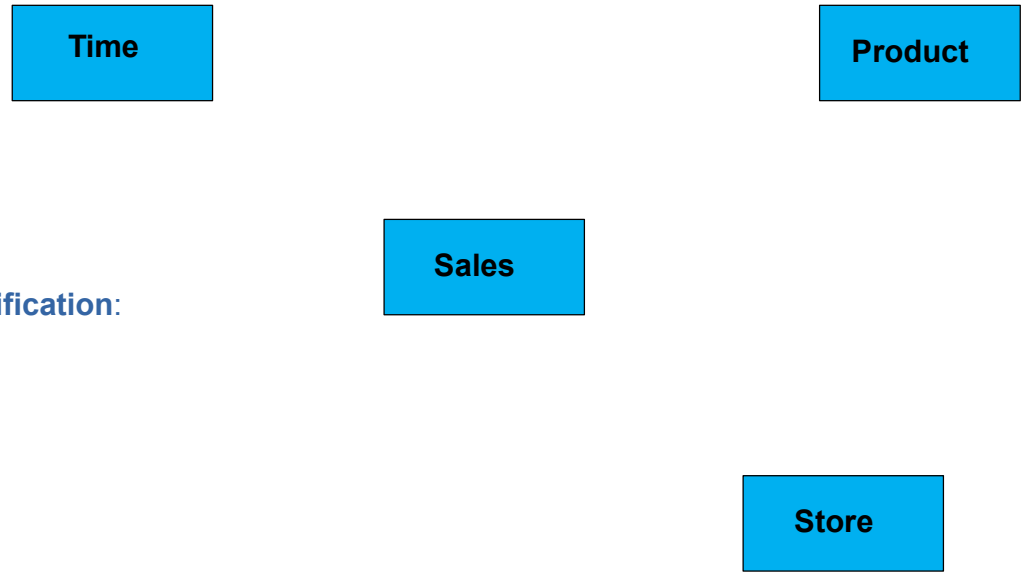
Course_id	Course_name	Duration (months)
78	Big Data	4
56	Algorithm	2

Primary Key

EXAMPLE - 2

The design and implementation of a small database of a company like MERCADONA, this database will focus on storing the data about the sold product.

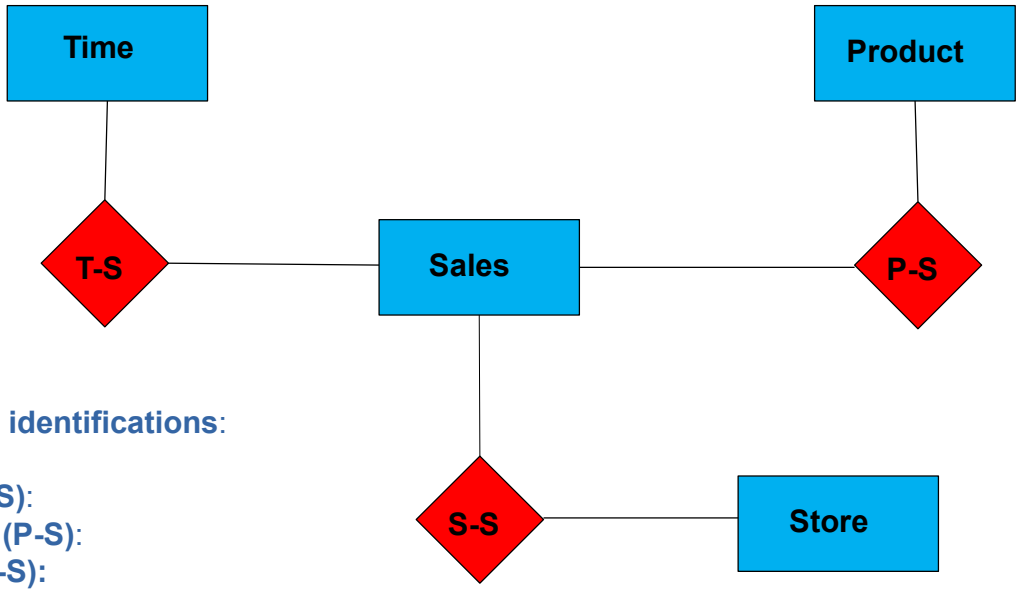
EXAMPLE – 2 (conceptual stage)



ENTITIES identification:

- **Time**:
- **Product**:
- **Sales**:
- **Store**:

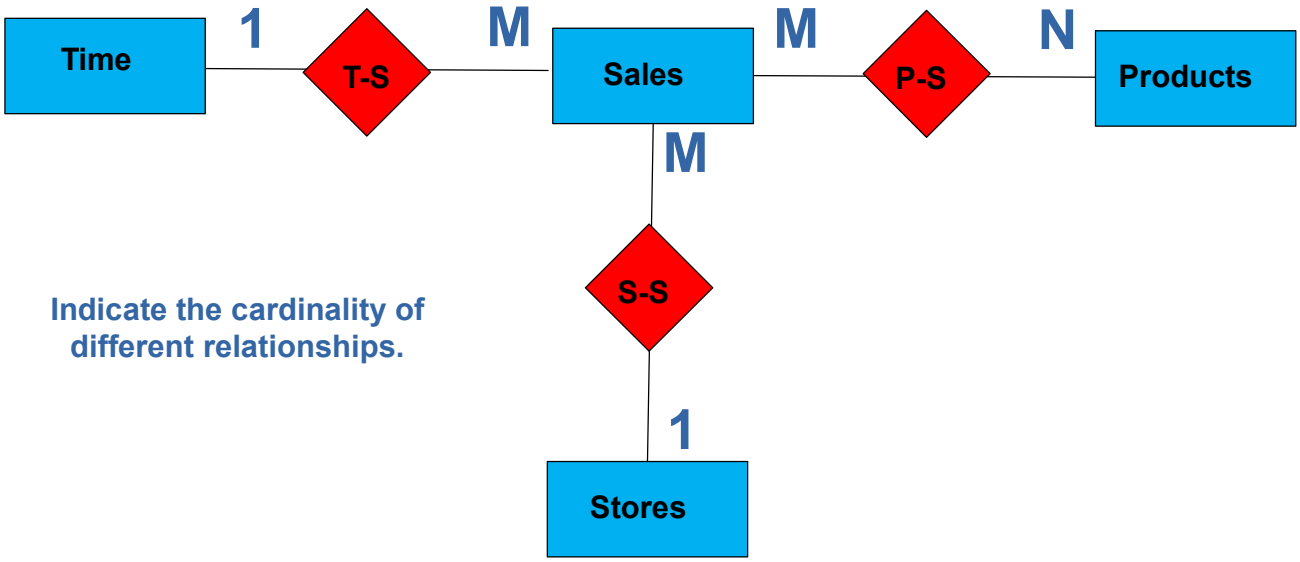
EXAMPLE – 1 (conceptual stage)



RELATIONSHIPS identifications:

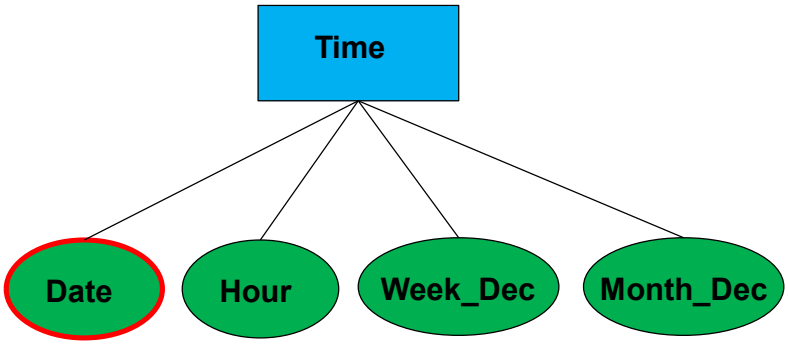
- **Time – Sales (T-S)**:
- **Product – Sales (P-S)**:
- **Store – Sales (S-S):**

EXAMPLE – 2 (logical stage)



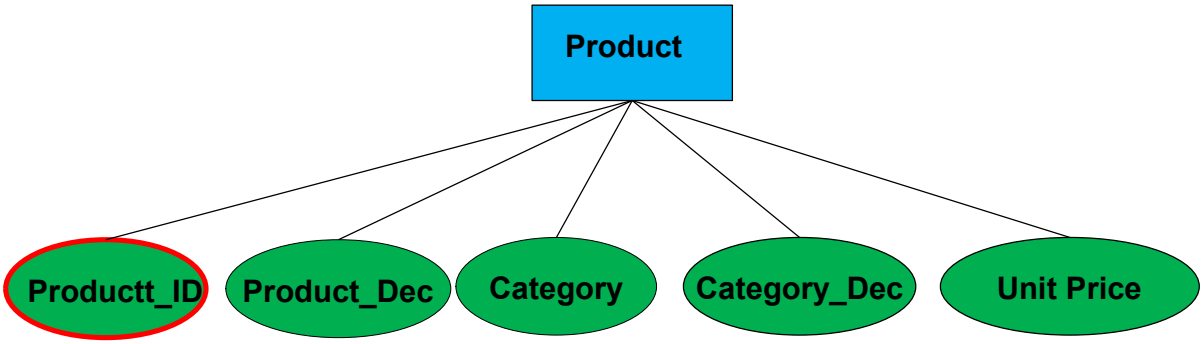
Indicate the cardinality of different relationships.

EXAMPLE – 2 (logical stage)



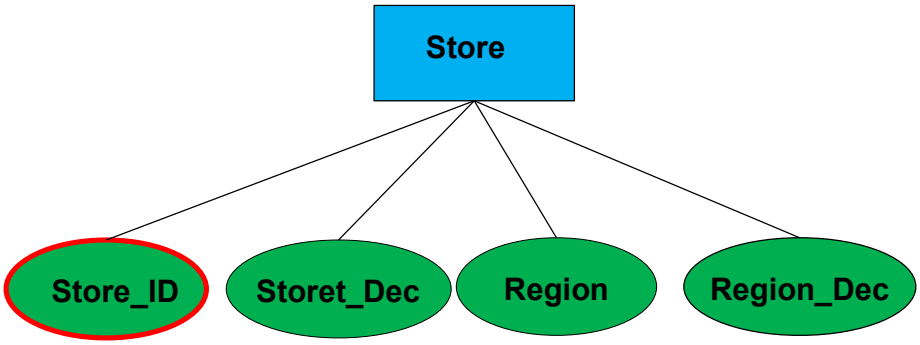
31

EXAMPLE – 2 (logical stage)



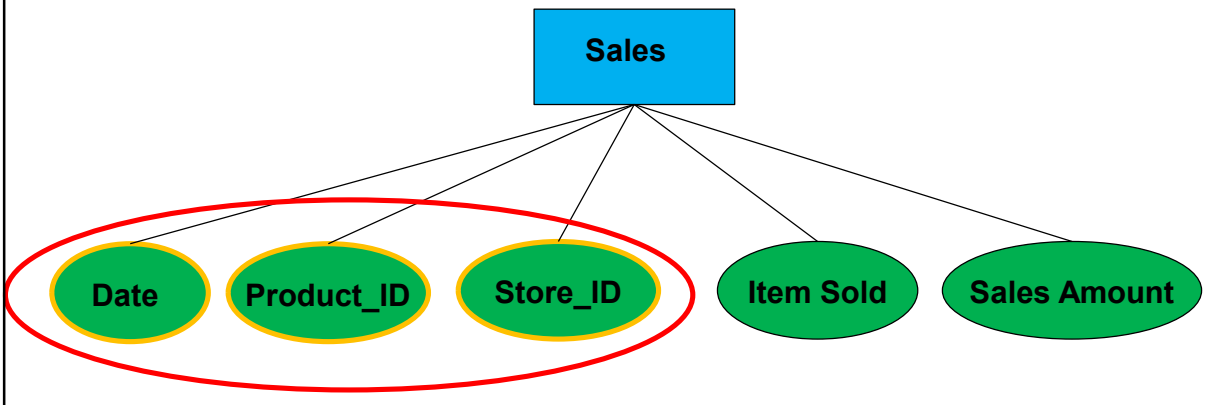
32

EXAMPLE – 2 (logical stage)



33

EXAMPLE – 2 (logical stage)



34

EXAMPLE - 3

The design and implementation of a small database that stores information on patients admitted to a hospital and attended by doctors.

EXAMPLE – 3 (conceptual stage)

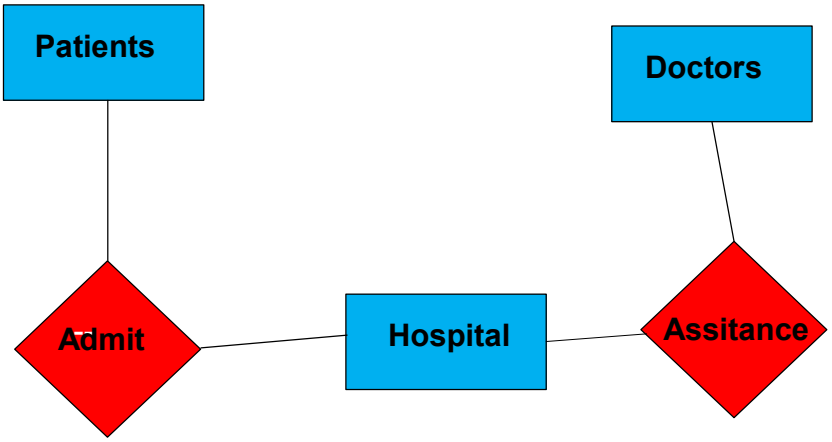
ENTITIES identification:

- **Patients**
- **Hospital**
- **Doctors**

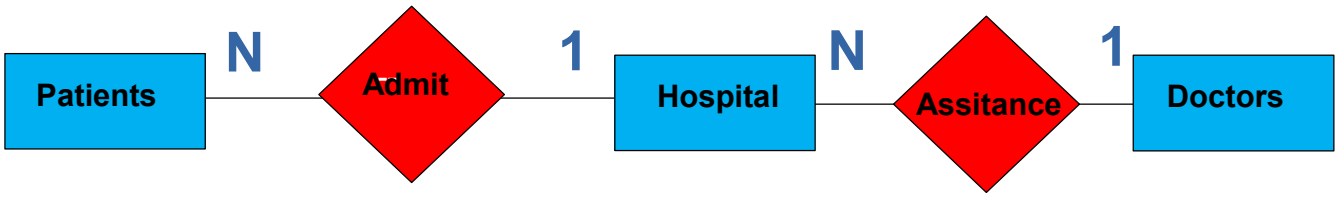
EXAMPLE – 3 (conceptual stage)

RELATIONSHIPS identifications:

- **Admit**
- **Assistance**

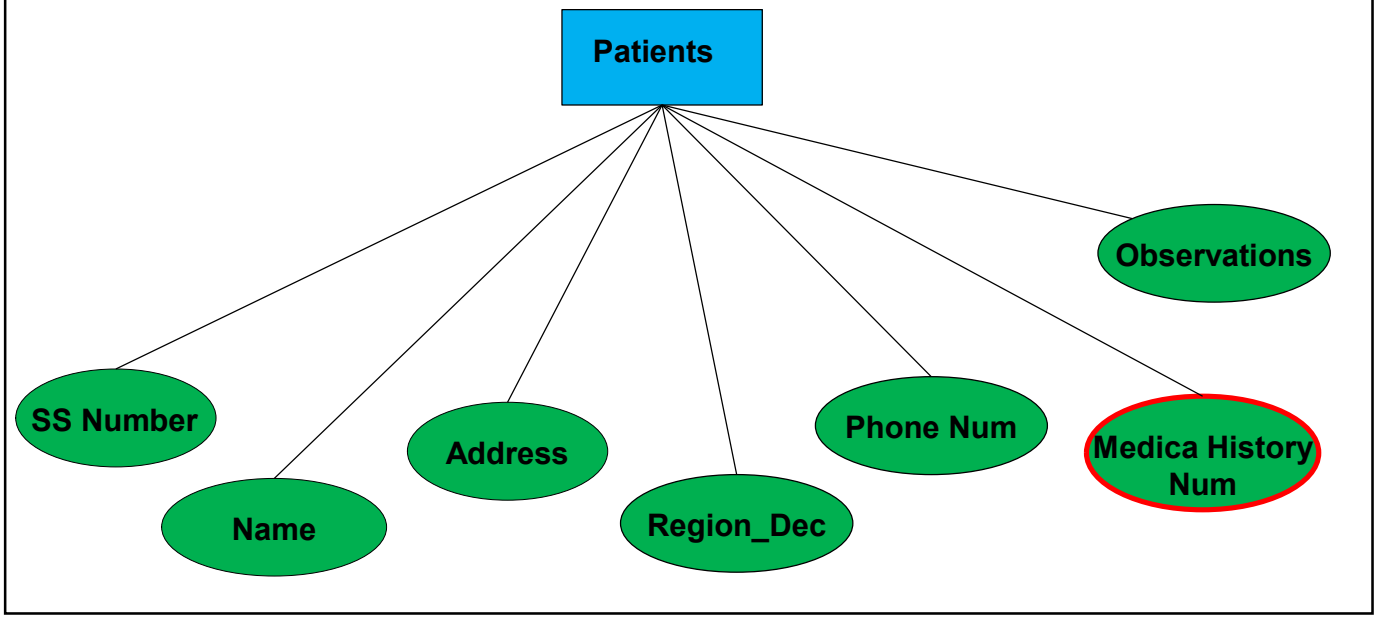


EXAMPLE – 3 (logical stage)



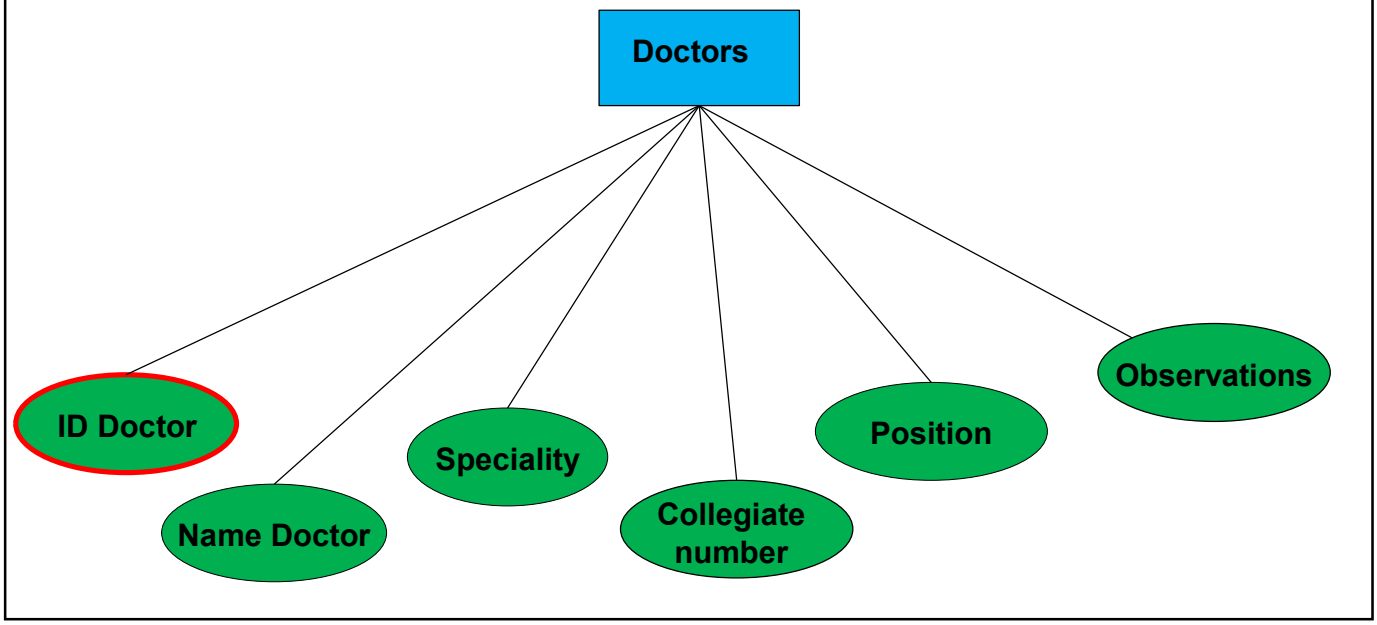
Indicate the cardinality of different relationships.

EXAMPLE – 3 (logical stage)



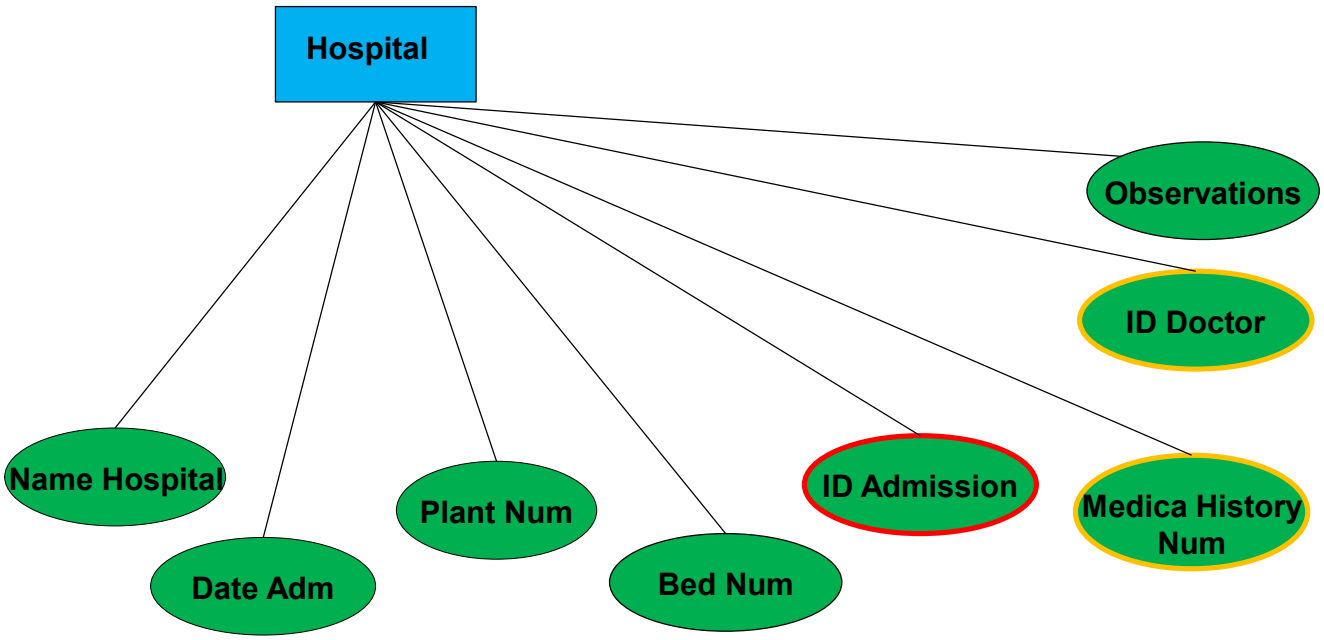
39

EXAMPLE – 3 (logical stage)



40

EXAMPLE – 3 (logical stage)



Types of Dependence

A **dependence**, in a broad sense, is a constrain between two sets of objects. In the case of Relational Database, these constrains are limitations imposed on the relationships between two sets of attributes.

For the purpose of this course, we are going to three different kind of concepts:

- **Functional Dependency** (**FD**)
- **Full Functional Dependence** (**FFD**)
- **Partial Functional Dependency** (**PFD**)

Functional Dependence (FD)

A set of attributes ***Y*** is functionally dependent on a set of attributes ***X*** if a given set of values for each attribute in ***X*** determine unique values for the set of attributes in **Y**.

STUDENT

Student ID	First Name	Last Name
000348282	Tirilee	Lytler
234556677	Koons	Smith
777349993	John	Doe
800005555	Patrick	Saint
999001111	James	Kirkland
999009800	Mary	Smith
999110000	Sally	Jones
666110001	Bill	Mack
555119800	Greg	Johnson
444339837	Jacob	Schwartz
333448887	Jenny	Donald
222551288	Nancy	Ogur

$X \rightarrow Y$

Given a set of values of a set attributes ***X***, the value of ***Y*** is determined uniquely.

STUDENTS = { Student ID, First Name, Last Name }

{First Name, Last Name } are functionally dependent on **{Student ID}**

Partial Functional Dependence (PFD)

A functional dependency ***X→Y*** is called partial functional dependent if some attribute ***A∈X*** (*A* belong *X*) can be remove from ***X***. and the dependency will still hold.

Table 13: ASSIGN

Person-ID	Poject-Budget	Project	Time-spent-by-person-on-project
S75	32	P1	7
S75	40	P2	3
S79	32	P1	4
S79	27	P3	1
S80	40	P2	5

ASSING = { Person-ID, Project-Budget, Project, Time-spend-by…}

{Project} is functionally dependent on **{Person-ID, Project-Budget, Project, Time-spend-by…}**.

Y ***X***

Partial Functional Dependence (PFD)

Table 13: ASSIGN

Person-ID	Poject-Budget	Project	Time-spent-by-person-on-project
S75	32	P1	7
S75	40	P2	3
S79	32	P1	4
S79	27	P3	1
S80	40	P2	5

ASSING = { Person-ID, Project-Budget, Project, Time-spend-by…}

But in this case is possible to find a subset of **{Person-ID, Project-Budget, Project, Time-spend-by…}** that can determine uniquely **{Project},** this subset is **{ Person-ID, Project-Budget}.**

{ Person-ID, Project-Budget} holds the functional dependence of **{Project}.**

45

Full Functional Dependence (FFD)

Y is functionally dependent on a set of attributes ***X*** and **Y** is not functionally dependent of any subset of ***X***. So, ***X*** is the minimum set of attributes that determine uniquely ***Y***.

Table 13: ASSIGN

Person-ID	Poject-Budget	Project	Time-spent-by-person-on-project
S75	32	P1	7
S75	40	P2	3
S79	32	P1	4
S79	27	P3	1
S80	40	P2	5

Considering the previous case, we have that:

{Project} is fully functional dependent on **{ Person-ID, Project-Budget}.**

It is not possible to determine unique values of **{Project}** with just values of **{Project-Budget}** or **{Project-Budget}** .

46

Full Functional Dependence (FFD)

Considering the first case:

{First Name, Last Name} is fully functional dependent on {Student ID}.

STUDENT

Student ID	First Name	Last Name
000348282	Tirilee	Lytler
234556677	Koons	Smith
777349993	John	Doe
800005555	Patrick	Saint
999001111	James	Kirkland
999009800	Mary	Smith
999110000	Sally	Jones
666110001	Bill	Mack
555119800	Greg	Johnson
444339837	Jacob	Schwartz
333448887	Jenny	Donald
222551288	Nancy	Ogur

Standardization (Normalization Process)

STANDARDIZATION (or also called **NORMALIZATION PROCESS**) is a part of logical design for the correct design of a database.

Without this process, the database system could be inefficient, slow, inconsistency or the data access or its update to be inaccurate.

A poorly design database may provide important **problems of data management.**

Standardization

Most of the problems with data management are the result of a bad design. This bad design is related to two features called:

- **Redundant data**: unnecessarily repeated data
- **Anomalies**: any incident due to the irregularities and inconsistencies in the storage can be affect to the data integrity.

Standardization (objectives)

The objectives of a standardization or normalization process are:

- arrange data into logical groups
- minimize the amount of duplicated data necessary
- a quick access and manipulation of data
- ensure data dependencies make sense

Standardization just adjusts or “tunes” the database for an optimize performance of this one.

Standardization (basic forms)

The standardization o normalization process involves several steps called **FORMS**:

- **BASIC FORMS**
  - **1NF**
  - **2NF**
  - **3NF**
  - 4NF
  - 5NF
  - 6NF
- **HIGH-LEVEL FORMS**
  - EKNF
  - ETNF
  - DKNF
  - …

Standardization (1NF)

1NF (First normal Form): All rows have unique and atomic values.

A database meets **1NF** if:

- No duplicate columns
- Each cell must only contain a single values (atomic value, no a list of values)
- Each value should not be divisible

Standardization (1NF) - Examples

Does table meet 1NF?

STUDENT

Student ID	First Name	Last Name	Last Name
000348282	Tirilee	Lytler	Lytler
234556677	Koons	Smith	Smith
777349993	John	Doe	Doe
800005555	Patrick	Saint	Saint
999001111	James	Kirkland	Kirkland
999009800	Mary	Smith	Smith
999110000	Sally	Jones	Jones
666110001	Bill	Mack	Mack
555119800	Greg	Johnson	Johnson
444339837	Jacob	Schwartz	Schwartz
333448887	Jenny	Donald	Donald
222551288	Nancy	Ogur	Ogur

Standardization (1NF) - Examples

Does table meet 1NF?

STUDENT

Student ID	First Name	Last Name	Last Name
000348282	Tirilee	Lytler	Lytler
234556677	Koons	Smith	Smith
777349993	John	Doe	Doe
800005555	Patrick	Saint	Saint
999001111	James	Kirkland	Kirkland
999009800	Mary	Smith	Smith
999110000	Sally	Jones	Jones
666110001	Bill	Mack	Mack
555119800	Greg	Johnson	Johnson
444339837	Jacob	Schwartz	Schwartz
333448887	Jenny	Donald	Donald
222551288	Nancy	Ogur	Ogur

No, we have duplicate columns. We can solve this problem removing one of duplicate columns.

Standardization (1NF) - Examples

Does table meet 1NF?

STUDENTS AND CLASSES (MODULES) DETAILS TABLE
Brian Smith, NBC Services, London, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Laura Grey, ABC Corporation, Nottingham, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000
John Brown, NBC Services, London, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Peter Black, ABC Corporation, Nottingham, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000

Standardization (1NF) - Examples

Does table meet 1NF?

STUDENTS AND CLASSES (MODULES) DETAILS TABLE
Brian Smith, NBC Services, London, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Laura Grey, ABC Corporation, Nottingham, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000
John Brown, NBC Services, London, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Peter Black, ABC Corporation, Nottingham, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000

No, there is no a single value in each cell (no Atomic data). We can solve the problem splitting the information off and creating different tables with this information.

Standardization (1NF) - Examples

STUDENTS AND CLASSES (MODULES) DETAILS TABLE
Brian Smith, NBC Services, London, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Laura Grey, ABC Corporation, Nottingham, Introduction to PCs, Introduction to Windows, Introduction to the Internet, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000
John Brown, NBC Services, London, Microsoft Word Advanced, Microsoft Excel Advanced, Microsoft Access Advanced
Peter Black, ABC Corporation, Nottingham, VB.NET Windows-Based Applications, VB.NET Web Applications, VB.NET XML Web Services and Server Components, Solution Architecture, SQL Server 2000

Table 5: STUDENTS

Student ID	Student Name	Company Name	Student Location
1	Brian Smith	NBC Services	London
2	Laura Grey	ABC Corporation	Nottingham
3	John Brown	IBM Limited	Birmingham
4	Peter Black	MCI Software	Edinburgh

Table 6: CLASSES (MODULES)

CLASS ID	CLASSES(MODULE)
1	Introduction to PCs
2	Introduction to Windows
3	Introduction to the Internet
4	Microsoft Word Advanced
5	Microsoft Excel Advanced
6	Microsoft Access Advanced
7	VB.NET Windows-Based Applications
8	VB.NET Web Applications
9	VB.NET XML Web Services and Server Components
10	Solution Architecture
11	SQL Server 2000

57

Standardization (1NF) - Examples

Does table meet 1NF?

EMPLOYEE NAME	EMPLOYEE ID	EMPLOYEE LOCATION	EMPLOYEE CONTACT NO
AJAY	12455	DELHI	987565,356212
AMIT	89752	AGRA	455145,988965
AKSHAR	23654	HARYANA	987454
ALOKE	98765	KOCHI	124787

58

Standardization (1NF) - Examples

Does table meet 1NF?

EMPLOYEE NAME	EMPLOYEE ID	EMPLOYEE LOCATION	EMPLOYEE CONTACT NO
AJAY	12455	DELHI	987565,356212
AMIT	89752	AGRA	455145,988965
AKSHAR	23654	HARYANA	987454
ALOKE	98765	KOCHI	124787

No, we can split down further. We can solve this problem adding other column “Second Employee Contac No” or removing one of the numbers.

Standardization (2NF)

2NF (Second Normal Form): No partial dependencies.

A database meets **2NF** if:

- Database must be in 1NF.
- The attribute should not be fully functional dependent on the candidate keys.

Standardization (2NF)

Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Does table meet 2NF?

Candidate keys

61

Standardization (2NF)

2NF (NO PARTIAL DEPENDENCIES): This implies first to meet 1NF and besides to eliminate a functional dependency on part of any candidate key.

Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Does table meet 2NF?

Candidate keys

62

Standardization (2NF)

Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

The primary key of this tables is **Model full name**, but we can see that the attribute **Manufacture country** dependent on a proper subset of attributes **Manufacture** and **Model**.

63

Standardization (2NF)

The solution is two create another table with that another dependency.

Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaste	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Electric toothbrush manufacturers

Manufacturer	Manufacturer country
Forte	Italy
Dent-o-Fresh	USA
Brushmaster	USA
Kobayashi	Japan
Hoch	Germany

Electric toothbrush models

Manufacturer	Model	Model full name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Brushmaster	SuperBrush	Brushmaster SuperBrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

64

Standardization (3NF)

3NF (Third Normal Form): No transitive dependencies.

A database meets **3NF** if:

- Database must be in 1NF and 2NF.
- The attributes must only be determinable by the primary key and not by other attributes.

Standardization (3NF)

Does table meet 3NF?

ID	Name	Balance	Bank Code No	Bank
5486546547	Jose Clapton	300000	Kjhasdcnlkn	Lenber Bank
6356865153	Erik Entrena	500000	LÑMkjhskldhl	Leamberber Bank
5651635856	Andrew Jackson	1000000	Lknalcnklñkna	Savana Bank
54656616354	Martin Denver	150000	AlkjdJADÑLJ	Bank of New Jersey
63560000053	Erik Entrena	500000	Qañajslashjcc	Trigger Bank
54656618884	Martin Denver	300000	Lknalcnln	Bank of New York
111110000053	Erik Entrena	1000	Zzzpojñlmsadn	Bank of Poland

Standardization (3NF)

Does table meet 3NF?

ID	Name	Balance	Bank Code No	Bank
5486546547	Jose Clapton	300000	Kjhasdcnlkn	Lenber Bank
6356865153	Erik Entrena	500000	LÑMkjhskldhl	Leamberber Bank
5651635856	Andrew Jackson	1000000	Lknalcnklñkna	Savana Bank
54656616354	Martin Denver	150000	AlkjdJADÑLJ	Bank of New Jersey
63560000053	Erik Entrena	500000	Qañajslashjcc	Trigger Bank
54656618884	Martin Denver	300000	Lknalcnln	Bank of New York
111110000053	Erik Entrena	1000	Zzzpojñlmsadn	Bank of Poland

No, we can see a **TRANSITION DEPENDENCE** among **ID**, **Bank Code No** and **Bank.**

This is not desirable since someone who is updating the database may change the name of **Bank** but may forget updating **Banck Code No**.

67

Standardization (3NF)

ID	Name	Balance	Bank Code No	Bank
5486546547	Jose Clapton	300000	Kjhasdcnlkn	Lenber Bank
6356865153	Erik Entrena	500000	LÑMkjhskldhl	Leamberber Bank
5651635856	Andrew Jackson	1000000	Lknalcnklñkna	Savana Bank
54656616354	Martin Denver	150000	AlkjdJADÑLJ	Bank of New Jersey
63560000053	Erik Entrena	500000	Qañajslashjcc	Trigger Bank
54656618884	Martin Denver	300000	Lknalcnln	Bank of New York
111110000053	Erik Entrena	1000	Zzzpojñlmsadn	Bank of Poland

ID	Name	Balance	Bank Code No
5486546547	Jose Clapton	300000	Kjhasdcnlkn
6356865153	Erik Entrena	500000	LÑMkjhskldhl
5651635856	Andrew Jackson	1000000	Lknalcnklñkna
54656616354	Martin Denver	150000	AlkjdJADÑLJ
63560000053	Erik Entrena	500000	Qañajslashjcc
54656618884	Martin Denver	300000	Lknalcnln
111110000053	Erik Entrena	1000	Zzzpojñlmsadn

Bank Code No	Bank
Kjhasdcnlkn	Lenber Bank
LÑMkjhskldhl	Leamberber Bank
Lknalcnklñkna	Savana Bank
AlkjdJADÑLJ	Bank of New Jersey
Qañajslashjcc	Trigger Bank
Lknalcnln	Bank of New York
Zzzpojñlmsadn	Bank of Poland

Again, the solution is to split the table off, but in a correct way, without losing of any kind of information.

68

Physical design

Contents

1 Contextualization of the subject **1**

2 Selection of a DBMS **3**
2.1 How to select a DBMS? 4

3 Creation of tables **5**
3.1 Field types: definitions and properties 6
3.1.1 Field 6
3.1.2 Data Type 6

4 Primary key assignment **8**
4.1 Characteristics for Primary key assignment 9
4.2 Creating index 10

5 Creating Relationships **12**

6 Referential integrity **13**

1 Contextualization of the subject

At this point you could still ask yourself, why is this subject about databases necessary? In unit 1, we talked about the amount of data that is currently and continuously being generated, data from very diverse sources (IoT, ERP, CRM, social networks, internal transactions, cliens, sales, products, etc ...) and with a very varied structures. The storage of this data has great potential for business, since a correct analysis of this data can improve this one or create new lines of business.

The creation of a database (in our case, a Relational database) implies to ensure that the database meets the business' needs, being necessary a person in charge (the manager) of transmitting those needs to the team of database designers who will develop the database, since generally these are unrelated to the business and its needs.

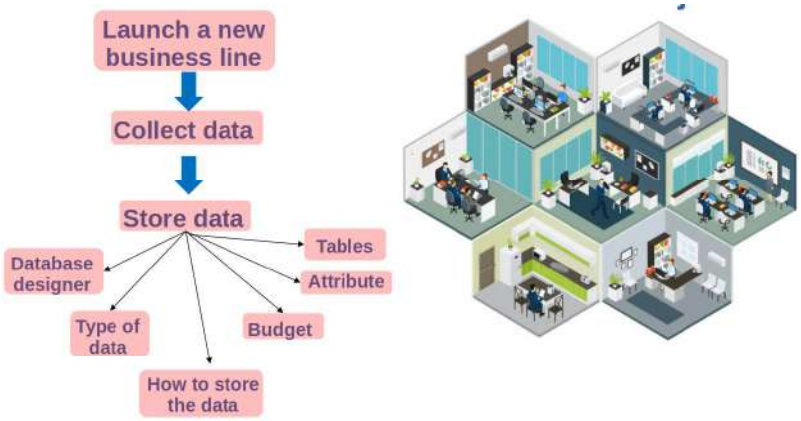


Figure 1

This manager (for instance you) has a idea of the database, a structure of the database (type of data stored, needed tables, attributes interesting, database final users, budget, etc ..., see Fig. 1), and also knows the purpose of it. Therefore all this needs to be explained and agreed with the team of database designers. Figures 2 and 3 show that interaction between them and how the tasks could be distributed throughout the design process (the three different stages in a Relational model) between the manager and the database designer.

Figure 4, 5 and 6 explain in more detail that distribution of tasks between the manager and database designer according to the different stages of Relational model, as well as the main goals sought in this division of tasks.

Figure 4, 5 and 6 also show a shaded region that indicates the region where there is an interaction between the manager and database designer. This region is just an estimate, with this region we only want to show that there will be an exchange of ideas between the manager and the database designer, there will be feedback and a flow of ideas to carry out the correct design of the database and meeting business needs.

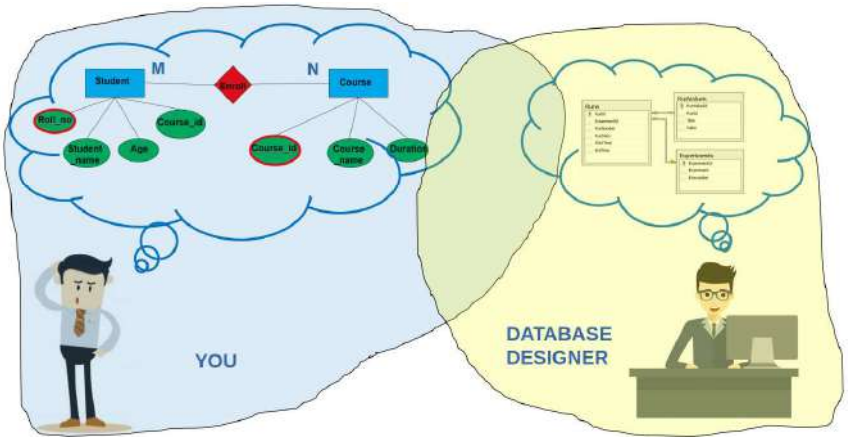


Figure 2: Interchange of ideas between the manager and database designer.

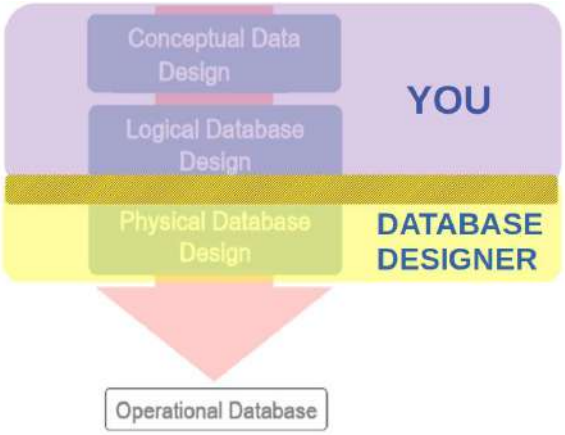


Figure 3: Possible distribution of relational model stages.

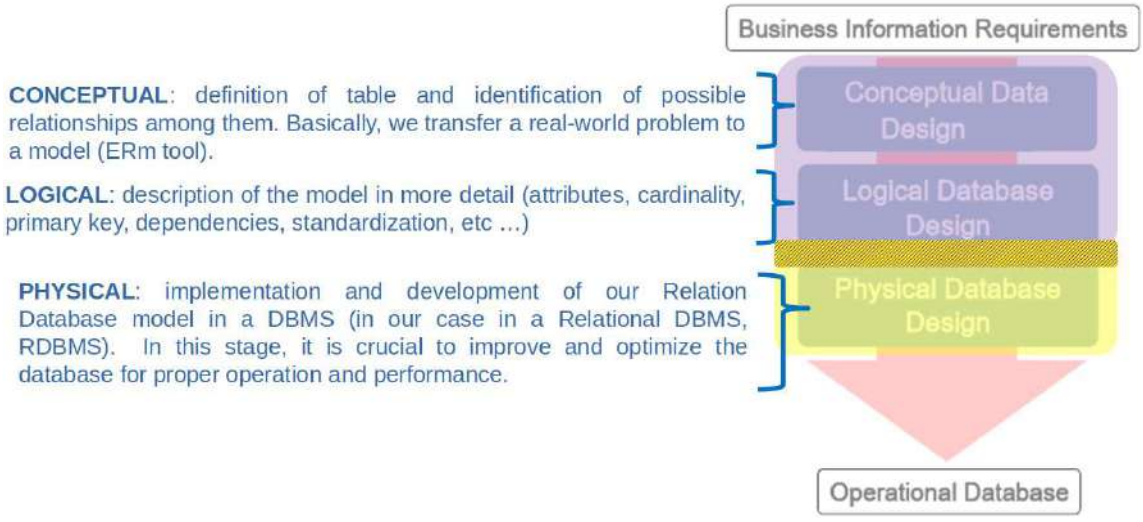


Figure 4: A brief explanation of tasks in relational model stages.

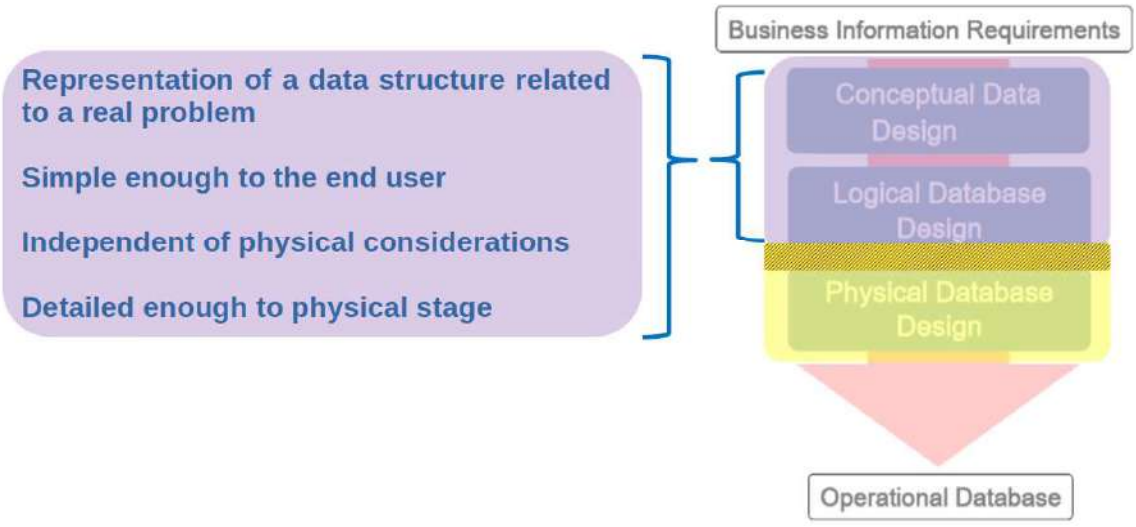


Figure 5: Main goals in the conceptual and logical stages of a relational model.

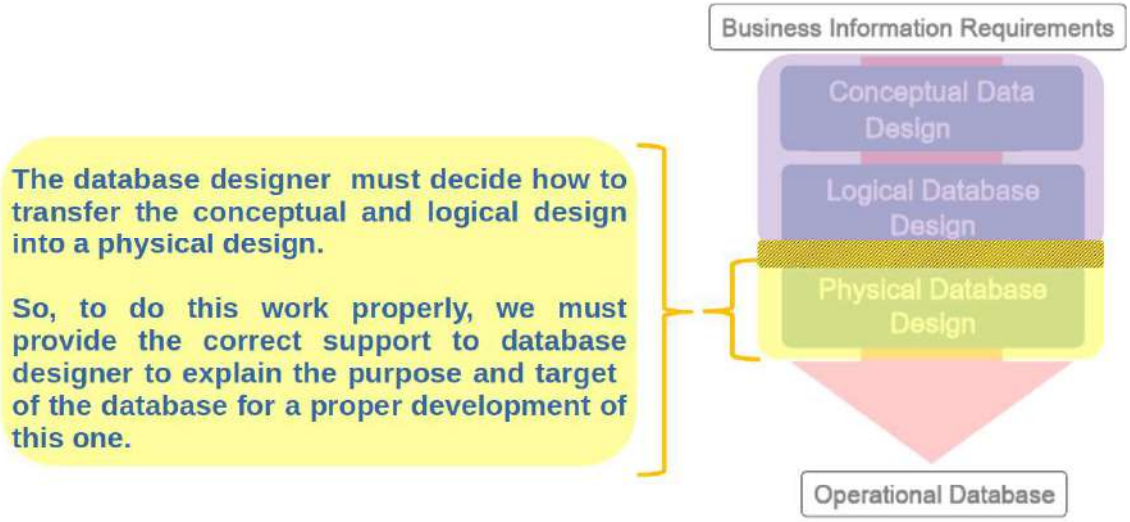


Figure 6: Main goals in the physical stages of a relational model, related to the manager and database designer.

2 Selection of a DBMS

The choice of a DBMS, in our case RDBMS (Relational DBMS), is a quite critical aspect because its choice partly constrains our PHYSICAL design.

Remember that the physical design implies how the data should be stored, structured and related in a specific DBMS according to conceptual and logical design. Therefore the physical design of a database depends on the chosen DBMS, since there may be more than one way of implementing any given part of the database. Consequently, to do this work properly, the manager and database designer must be fully aware of the functionality of the target DBMS, and must understand the advantages and disadvantages of different DBMS and alternative approaches for a particular implementation. So, the question is **How to select a DBMS?**.

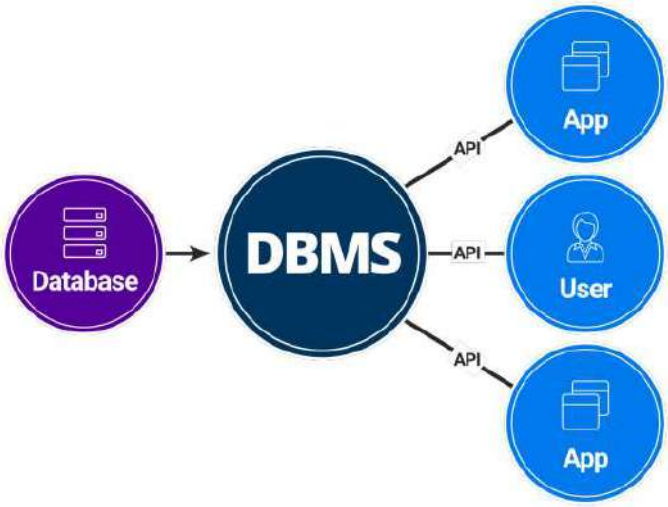


Figure 7: Database infrastructure.

2.1 How to select a DBMS?

Considering what has been mentioned above, what should we take into account when a DBMS is chosen? The choice of a DBMS depends on many factors, these ones could be divided in two big groups:

- from a company point of view
  - Number of users supported by DBMS.
  - Interfaces used (an easy-to-use interface).
  - Operative system of the company (Linux, Mac or Windows).
  - Cost of implementation (there are some nice, free, open-source databases available).
  - Trade-offs between:
    - "ease-to-use" and "features"
    - “open-source” and “support or documentation”
  - Organizational constraints since (a specific database or not to go off budget).
  - Types of data stored in the DBMS.
- from a technical point of view
  - Creation of table.
  - Definition and deal with keys (primary, candidate, . . .).
  - Whether the system support “required” constrains (for example, if the system accepts “Null values”).
  - whether the system supports the definition of domains (restrictions on allowed values).

– whether the system does not support some operations between data

There are a lot of different consideration to take into account, different business's needs for different types of business. Above, we have indicated the most relevant to be considered.

3 Creation of tables

Creation or design of tables is one of the most important actions done to develop and implement a Relational database. The creation of tables decides how to represent the data in tables according to our DBMS chose. From Logical design, we had a set of table schemas that describe:

- the name of the relation.
- a list of simple attributes.
- the primary key and, where appropriate, alternate keys (AK) and foreign keys (FK).
- standardization

Now we have to add additional information, since we have to specify every attribute in a relation:

- domain (type, length, pattern and other constrains)
- whether the attribute can hold nulls
- default value for the attribute (optional)
- derived attributes ((attributes calculated from other attributes)

All this information can be given in a **DATA DICTIONARY**, a type of scheme in table form where is described the type of attributes in a relation.

Continuing with our example of the **database of enrolled student in courses**, we have the following **DATA DICTIONARY** for each relation. Attributes in blue letters are **Primary Keys** and green letters indicate **Foreign Keys**.

- **STUDENTS**

Table 1: STUDENTS Data Dictionary

Attribute	Description	Data Type	Domain	Null?
Roll_no	Enrolled ID		nn nnnn	No
Student_name	name		String	No
Age	Age		Positive integer	No
Course_id	Course ID		ABC-nnnnnn	No

- **COURSES**

Table 2: COURSES Data Dictionary

Attribute	Description	Data Type	Domain	Null?
Course_id	Course ID		ABC-nnnnnn	No
Course_name	Course name		String	No
Duration	Cours hours		Positive integer	No

In order to the domain of **Roll_no**, it has been chosen a patter of numbers, while for the domain of **Course_id** the chosen pattern is a mix of letters and number with a dash. In these cases, there is a certain freedom choosing the pattern by manager and database designer. It all depends on how you want to design a specific attribute.

For the rest of attributes there is less freedom, since for the attributes **Student_name** and **Courset_name** their domains are a simple chain of characters (string), while for **Age** and **Duration** their domains are just numbers (positive integers), respectively.

IMPORTANT: A domain can be descriptive a specific pattern (for instance, dd/mm/yyyy-hh:mm), integer number positive or not (for instance, 99999) or a list of specific values for instance, a string), etc... all this depends on the characteristic of the attribute.

IMPORTANT: Primary and Foreign keys can never be NULL because they are fields that are always required. Remember that the primary key help us to uniquely identify each row in a table, a NULL value for a primary key in a row means that the records of this row are not identify for the database.

3.1 Field types: definitions and properties

In this section two concepts are presented: **Field** and **Data Type**, these concepts are related with the idea of attribute.

3.1.1 Field

The attributes of tables are represented as fields in ERm. For the purpose of this course, we can identify three type of fields:

- Primary key - field used to uniquely identify a row in a table.
- Foreign key - field used to link several tables.
- Descriptive field - any field that store data except the above indicate.

After a field is created it is necessary to set what kind of field, to set its **Data Type**.

3.1.2 Data Type

A **Data Types** is coding scheme recognized by system software in order to represent the data. The selection of a **Data Type** for the different attributes of tables seek to find the trade off among four objectives:

- Minimize storage space.

- Represent all possible values of the field.
- Improve data integrity of the field.
- Support all data manipulations desired on the field.

There are a lot of data types different for databases, below some basic ones are presented:

- **VARCHAR2(n)** → variable-length character with a maximum length of 4000 characters.

  VARCHAR2(30): the number in parentheses indicates the maximum length of characters, so the length of data can be 30 character or less.

- **CHAR(n)** → variable-length character with a maximum length of 255 characters; default length is 1.

  CHAR(5): the number in parentheses indicates the maximum length of characters, so the length of data can be 5 character or less.

- **LONG** → capable of storing up to two gigabytes of one variable-length characters data.

  LONG: to hold medical instructions or customer comments.

- **NUMBER** → capable of storing positive or negative values in a very wide range of values

  NUMBER(5): specified an integer number of a maximum of 5 digits. In some databases you can find INT(5), where INT specifies that it is an integer of 5 digits as a maximum.

  NUMBER(5,2): specified a number of a maximum of 5 digits, with two digits to the right of the decimal point. In any case the length id 5 digits in total as a maximum.

- **DATE**→ capable of storing dates in a lot of formats, generally this one must be specified in the **domain column** in the data dictionary.
- **BLOB** → capable of storing binary large object, up to four gigabytes. Tt is used for any type of digitized document (photograph, sound, video, pdf, etc...).

Taking into account the **DATA TYPE** the data dictionaries created before (Tables 1 and 2 will be modified as follows:

- **STUDENTS**

Table 3: STUDENTS Data Dictionary

Attribute	Description	Data Type	Domain	Null?
Roll_no	Enrolled ID	CHAR(7)	nn nnnn	No
Student_name	name	VARCHAR2(100)	String	No
Age	Age	VARCHAR2(3)	Positive integer	No
Course_id	Course ID	CHAR(10)	ABC-nnnnnn	No

- **COURSES**

Table 4: COURSES Data Dictionary

Attribute	Description	Data Type	Domain	Null?
Course_id	Course ID	CHAR(10)	ABC-nnnnnn	No
Course_name	Course name	VARCHAR2(25)	String	No
Duration	Cours hours	CHAR(3)	Positive integer	No

Why in the above tables is sometimes CHAR(n) and other times VARCHAR2(n) chosen? Well, in this case, concretely, we could have used CHAR(n) or VARCHAR2(n) indistinctly, because the length of characters is always lower than 255 ($n \leq 255$). If this length had been longer than 255 ($n \geq 255$), then we would have have used **VARCHAR2(n)** instead **CHAR(n)**.

4 Primary key assignment

Choosing a primary key is one of the most important steps in good database design. A primary key is a table column that serves several special purpose:

- it uniquely identifies each row in a table, ensuring row-level accessibility.
- it identifies each table in a database.
- it relates tables to each other (through foreign key).

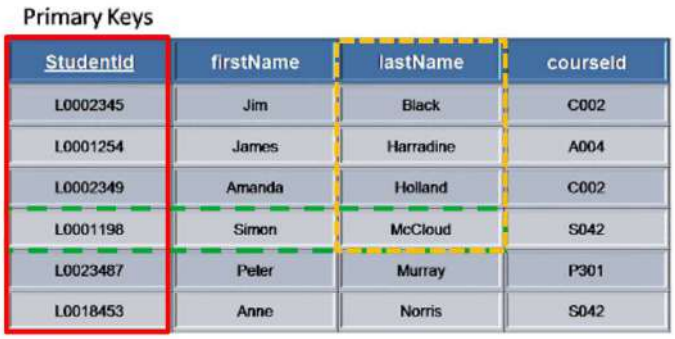


Figure 8: Row-level accessibility by primary key.

The values that compose a primary key column are unique; no two values are the same. Each table has one and only one primary key, which can be made up using one or several attributes (chosen attributes from the set of candidate keys, see Fig.9).

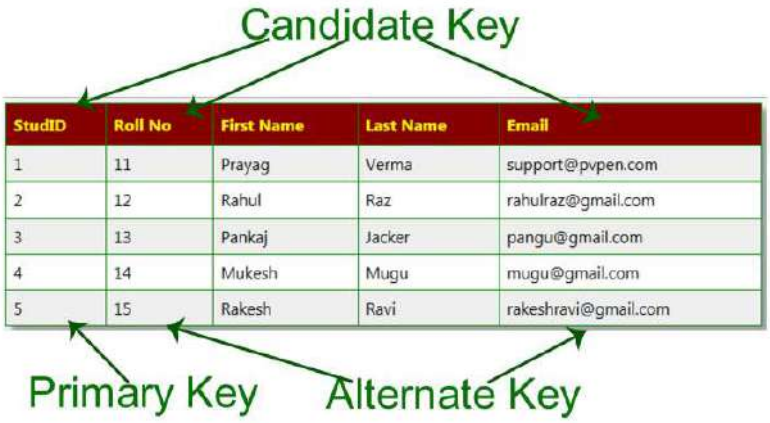


Figure 9: Primary and candidate keys.

4.1 Characteristics for Primary key assignment

There are several recommended characteristics of a candidate key must meet to be a good primary key are:

- **NEVER NULL**

  No primary key value can be null. This is an inviolate rule of relational database design and relational database management management system (RDBMS). A primary key whose values is NULL means that the records of the row in that table are not identified in the database.

- **BREVITY**

  The use of brief primary key (one attribute from the set of candidate keys) speeds up the data search and therefore improves the speed with which queries can be made in a database. Then, it is better to use, to improve the search speed, primary keys made up of a single attribute which is chosen from the candidate keys.

- **SIMPLICITY**

  Primary key without special characters (such as @, #, %, &, etc ...), blanks (embedded spaces) or or uppercase and lowercase. This facilitates the realization of queries, avoiding errors or misunderstandings .

- **PREFERRED DATA TYPE**

  NUMBER data types are the best choice for primary key, followed by fixed-length character data types. Generally RDB process *NUMBER* data type faster than others.

  On the other hand, Ffixed-length character data types are better than variable-length character data types because RDBMS must decompress variable-length character data before processing the data. This extra step consumes valuable processing power.

- **NONIDENTIFYING VALUE**

  A common mistake among database designers is trying to build intelligence, or meaning, in to the primary key. The most compelling reason not to create a primary key with meaning is that the primary key might become obsolete.

- **NEVER CHANGE**

  After you assign a primary key value, never change it. Imagine that the primary key of that relation (table) is a **foreing key** in others relations (tables), this would imply that you have to change the associated **foreing key** values in many tables.

4.2 Creating index

We must know that every time we execute a query, the DBMS analyses all the rows in the table to find the ones that match the request. As the records in the database grow, it is necessary to process more and more rows which at one time, decreases the overall performance of the DBMS. To avoid this, the indexes are created in the databases.

A index of a database is very similar to an index in a book: it occupies its own space, references information stored elsewhere, and thus makes it easy to search when querying those tables.

The **creation of indexes** is related to the concept of **indexing**. Indexing is a technique of databases to efficiently retrieve record from the database based on some attributed on which the indexing has been done.

An index has a structure of table with two columns (see Fig.10):

- **Search Key** - it is a copy of the attribute (primary key or another attribute) of a table.

  Data Reference or Pointer - it contains values that hold the location where the records associated to chosen attribute are stored.

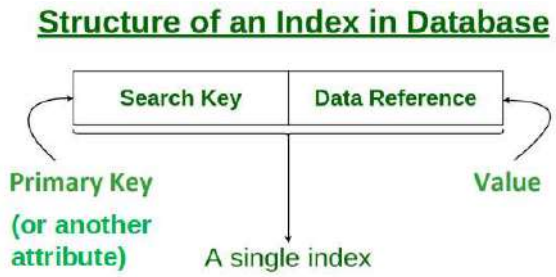


Figure 10: Index table.

The creation of INDEXES has pros and cons:

- **PROS**
  - INDEXES reduce the data search time.
- **CONS**

– INDEXES make it slower to add or update to existing rows for tables and occupy storage.

– It is necessary a storage space to store the index tables.

Below it is shown, by means of a comparison, how indexing works.

- **Without index**

  If I want to know **the value of "magB" with "haloid=6626"** (yellow box) the database would analyses all the rows in the tables till to find those data that match our request.

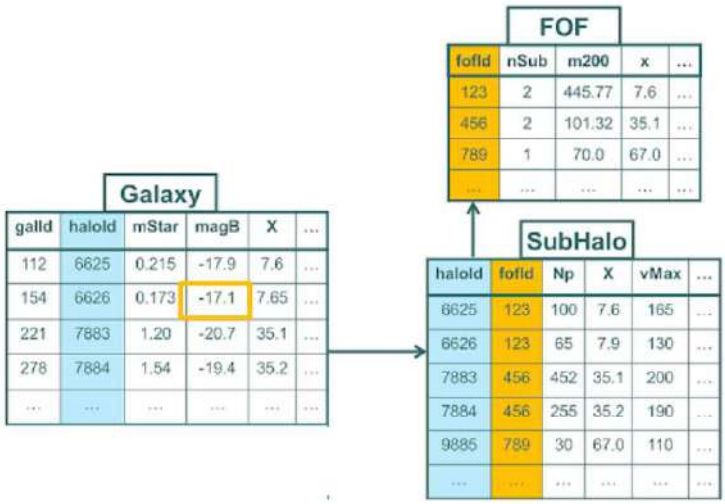


Figure 11: Index table.

- **With index**

  If I want to know **the value of "magB" with "haloid=6626"** the database would firstly read the index and go directly to row where the record is (red ovals), this just needs to find in the row the specific attribute ("magB" in our case).

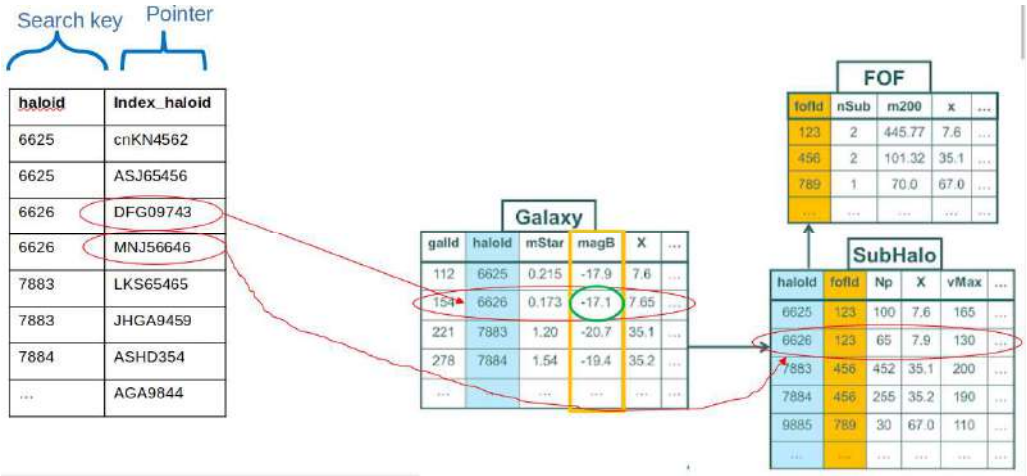


Figure 12: Index table.

5 Creating Relationships

A relationships between two tables is created through matching an attribute that appears in both tables. The main kind of relationship consider in this course is through a primary key (see Fig.13), so the attribute shared is the primary key in one table (**parent table**) and is the foreign in another tables (**associated tables**).

Figure 13: Relationship between table by sharing the primary key.

Figure 14: Index table.

It is possible to create other kind relationships sharing other type of attributes or even to have two table related to each other in several different ways. This kind or secondary relationships (see Fig. 14, yellow boxes) can be accidental or done knowingly, it all depends on the business needs it all depends on the business' needs. The simpler the relationships are, the better the database works, but in some cases a secondary relationship could be needed.

6 Referential integrity

The concept of referential integrity refers to the accuracy and consistency of data within a relationship. In a relationships, the data is linked between two (or more) tables. This is mainly achieved by the foreign key values in the associated tables reference a primary key values in parent table.

Thus, referential integrity requires that, whenever a foreign key value is used in associated table this value must reference a valid and existing value of a primary key in the parent table, but the reverse is not necessary to be fulfilled. It means that whenever a primary key value is used in parent table this value does not need to reference a valid and existing value of a foreign key in the associated table (see Fig.15).

Figure 15 shows a simple database with two tables: **able of Artist** and **Table of Album**. In this case, it is logic to this

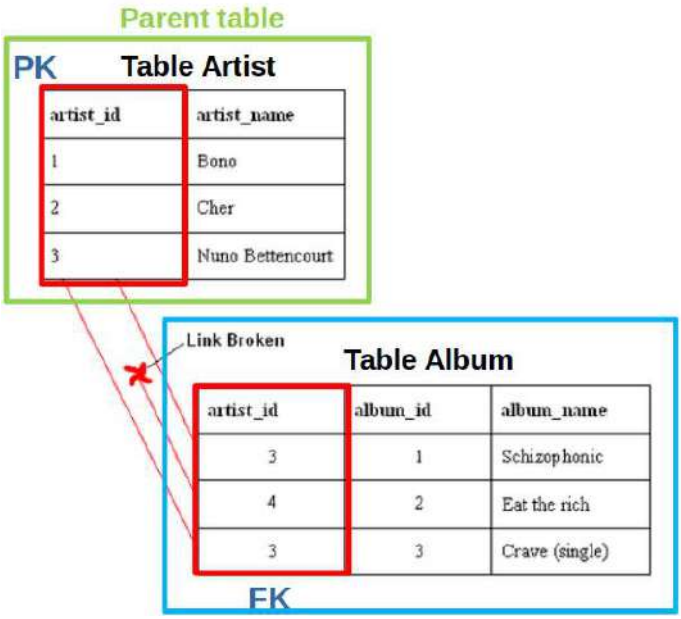


Figure 15: Example of referential integrity

Another way of saying it is that the domain range of the primary key in the parent table must be greater than or equal to the domain range of the foreign key in the associated table.

A lack of referential integrity can imply that some records in the parent table have been lost after some manipulation (duplication, updating, copying, adding new data, etc ...) in the table.

1

Unit 4 – Physical stage

- How to select a DBMS?
- Creation of tables
- Field types: definition and properties
  - Field
  - Data type
- Primary key assignment
- Creating index
- Creating relationships
- Referential integrity

2

NEWS

1º PARTIAL EXAM

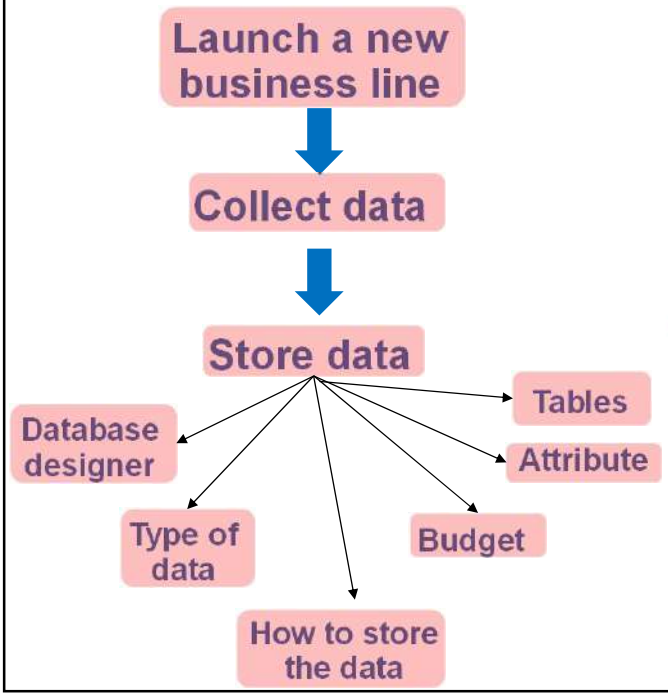
- 28/04/2021
- Units 1, 2, 3 and 4
- Exam score 10 point
- Two parts:
  - 20 multiple choice question
  - Make a Relational model (Conceptual, Logical and Physical design)

2º PARTIAL EXAM

- 19/05/2021
- Unit 5
- Practical exam (10 question approx.)

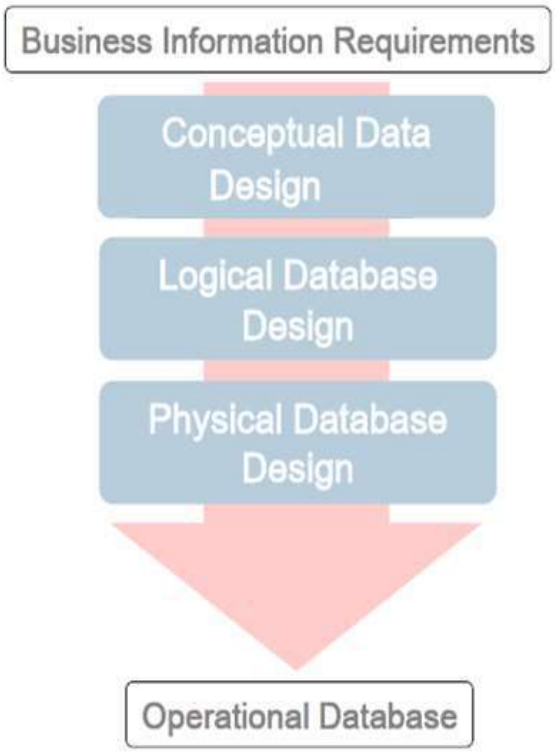
3

Contextualization of the subject



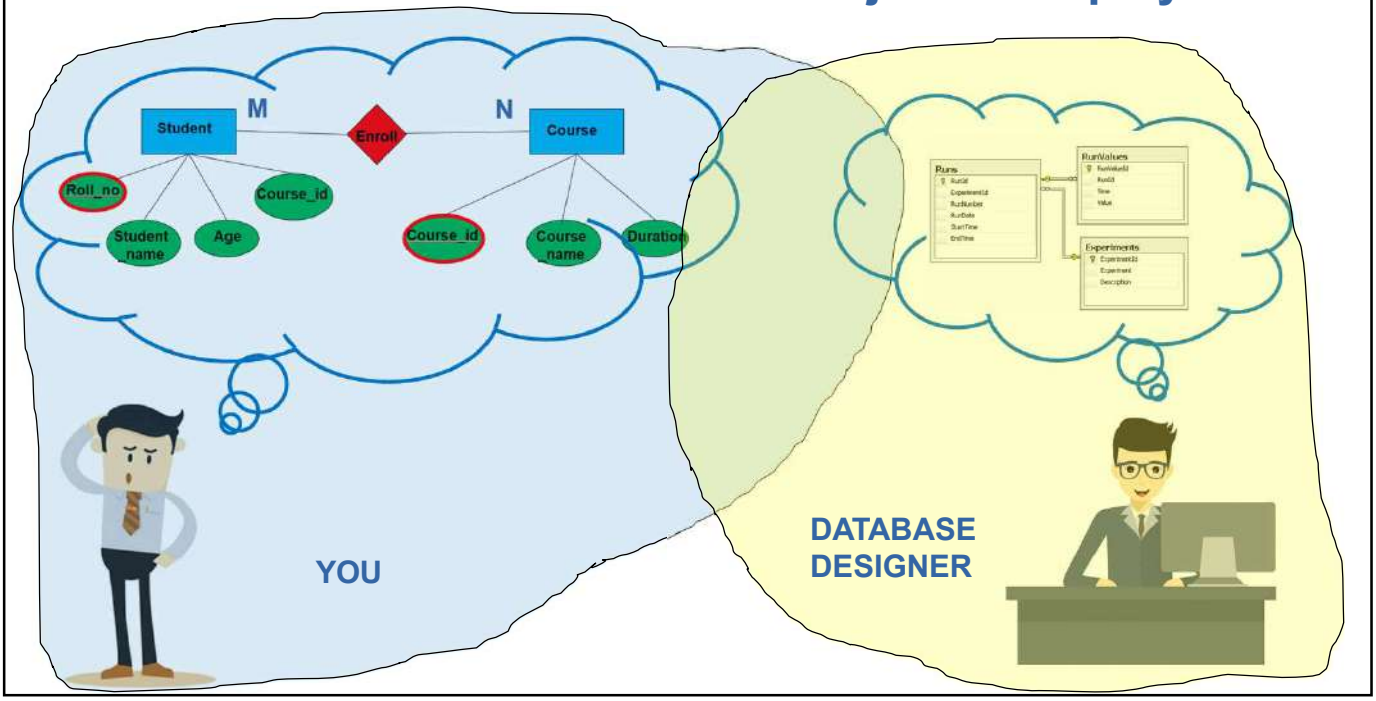
4

Contextualization of the subject



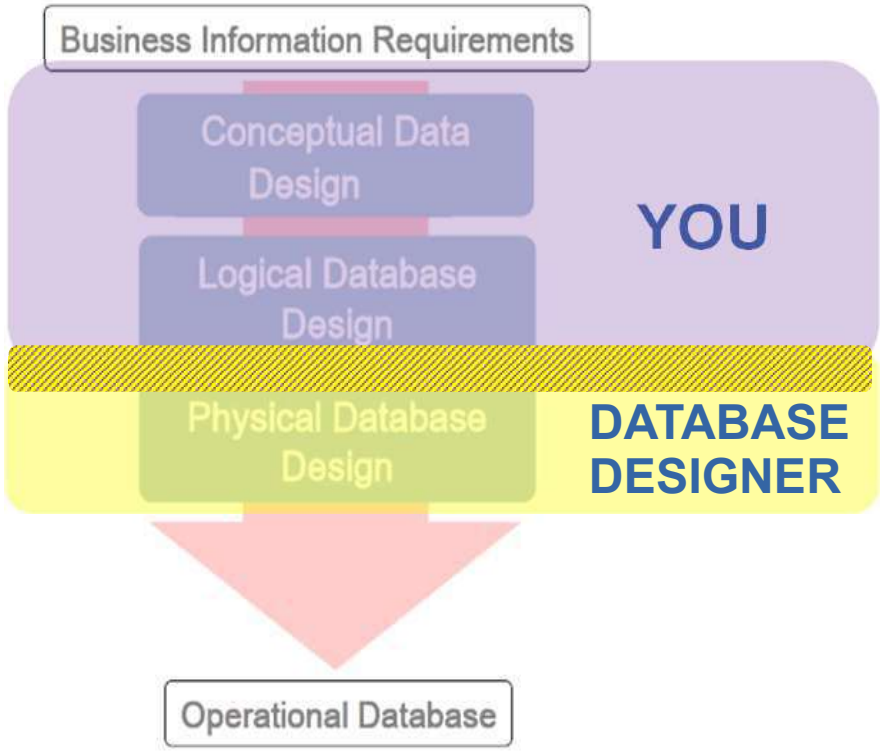
5

Contextualization of the subject: Two players



6

Contextualization of the subject: Two players

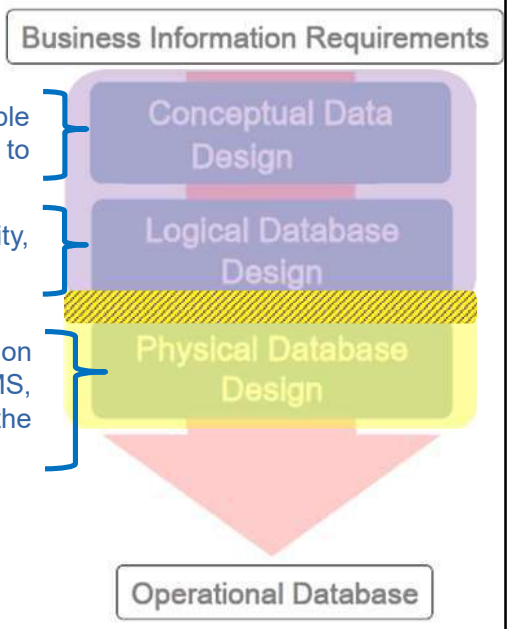


Review of DESIGN STAGES

CONCEPTUAL: definition of table and identification of possible relationships among them. Basically, we transfer a real-world problem to a model (ERm tool).

LOGICAL: description of the model in more detail (attributes, cardinality, primary key, , dependencies, standardization, etc …)

PHYSICAL: implementation and development of our Relation Database model in a DBMS (in our case in a Relational DBMS, RDBMS). In this stage, it is crucial to improve and optimize the database for proper operation and performance.



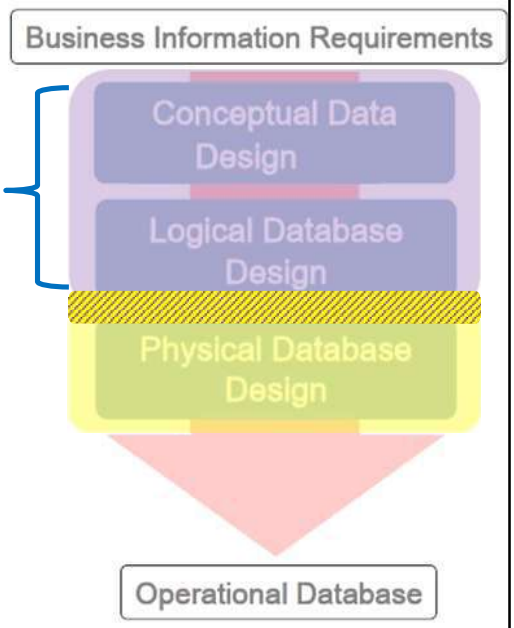
Review of DESIGN STAGES

Representation of a data structure related to a real problem

Simple enough to the end user

Independent of physical considerations

Detailed enough to physical stage

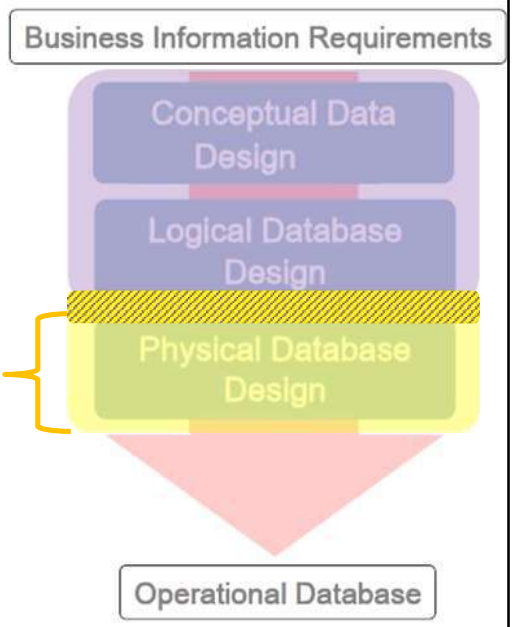


9

Review of DESIGN STAGES

The database designer must decide how to transfer the conceptual and logical design into a physical design.

So, to do this work properly, we must provide the correct support to database designer to be fully aware of purpose and target of the database for a proper development of this one.

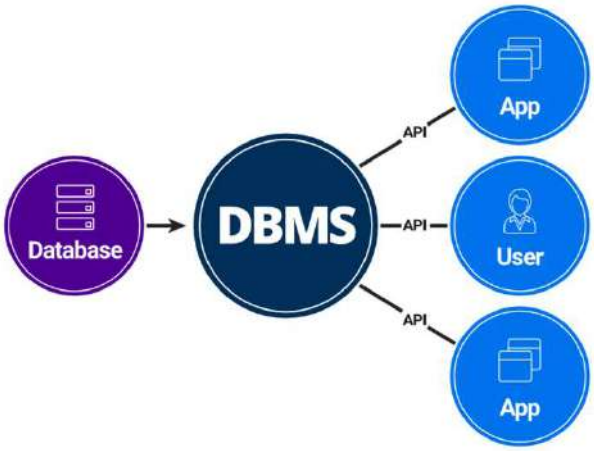


10

Selection of a DBMS

The choice of a RDBMS is critical, because this choice partly determines our physical design. How to select a DBMS? In our case RDBMS (Relation DBMS).

REMEMBER!!! The physical design implies how the data should be stored, structured and related in a specific DBMS according to conceptual and logical design.



So, it is very important to know the advantages and disadvantages that the DBMS provides us for a particular implementation.

Selection of a DBMS

The choice of a RDBMS depends on many factors, these ones could be divided in two big groups:

From a company point of view

From a technical point of view

Selection of a DBMS

From a company point of view:

- Number of user
- Interface (an easy-to-use interface)
- Operative system (Windows, Mac or Linux)
- Cost of implementation (open-source database)
- Trade off between:
  - "ease-to-use" and "features"
  - "open-source" and "support or documentation"
- Organizational constrains (a specific database or not to go off budget)
- Type of data allowed in the DBMS.

Selection of a DBMS

From a technical point of view:

- Creation of table
- Definition and deal with keys (primary, candidate, …)
- Whether the system support "required" constrains ("Null values")
- Whether the system support definition of domains ("allowed values")
- Whether the system does not support come operations between data

These ones are the most relevant but there are much more!!!!

Creation of tables

The creation or design of table is to decide how to represent the data in table according to our DBMS chosen.

From Conceptual and Logical stages, we get:

- The name of the different relations (tables)
- List of attributes for each table
- The primary key for each table
- The different relationships among tables (foreign key)
- The cardinality of relationships

Creation of tables

In physical stage we must add **additional information** for each attribute in a relation, we have to specify:

- Domain: data type, length and other constrains
- Whether the attribute can hold NULL value
- Default value for attributes
- Derived attribute (attributes calculated from other attributes)
- … etc

All this information can be given in a **DATA DICTIONARY**, a type of scheme in table form where is described the type of attributes in a relation.

Creation of tables - IMPORTANT

DOMAIN: it describe the range if valid values that an attribute can take, but the domain can describe a pattern data, for instance:

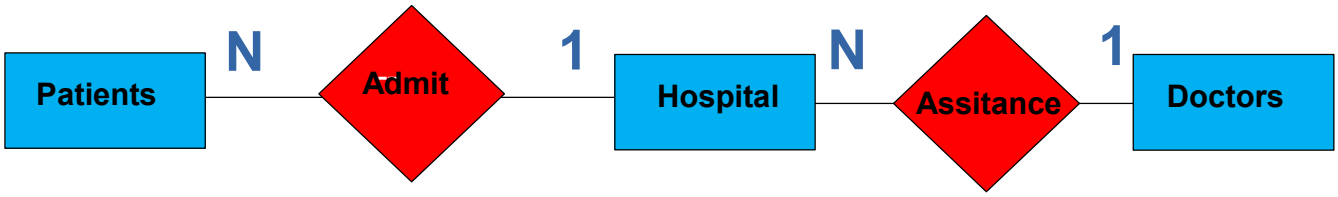
- a date have a specific patter: dd/mm/yyyy hh:mm
- a car license plate: nnnn abc
- ID card: nnnnnnnnA

Primary and **Foreign keys** can never be NULL value, because they are always required.

EXAMPLE - 3

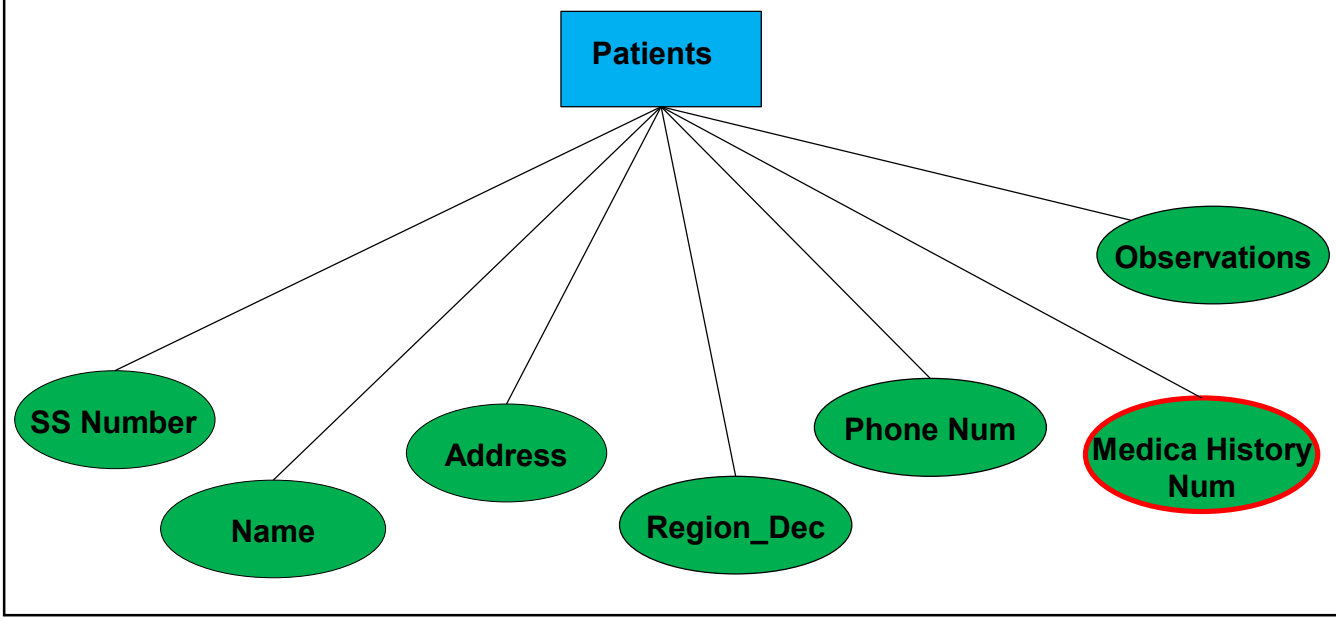
The design and implementation of a small database that stores information on patients admitted to a hospital and attended by doctors.

EXAMPLE – 3 (logical stage)



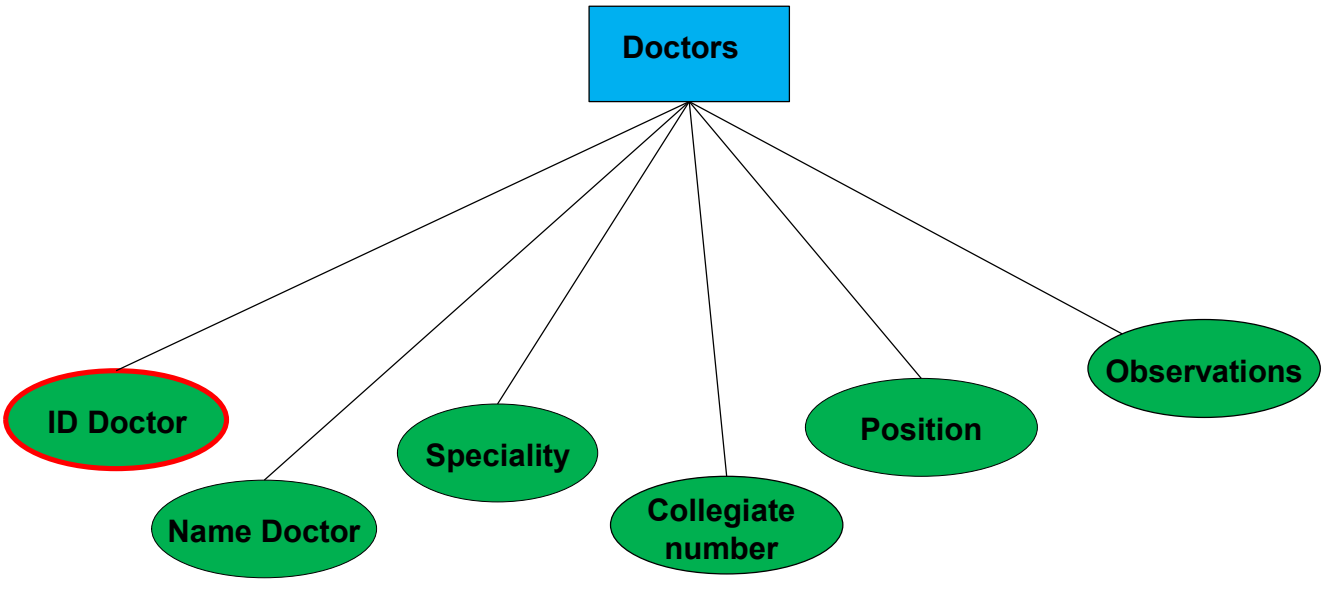
Indicate the cardinality of different relationships.

EXAMPLE – 3 (logical stage)



Attribute	Description	Data type	Domain	Null?	
SS Number	Id. patient		nnnn nnnn nnnn nnnn	No	
Name	Name patient		string	Yes	
Address	Address patient		string	Yes	
Region_Dec	Region patient		string	No	
Phone number	Phone patient		nnn nn nn nn	Yes	
M. H. Number	**Id. Code**		**String**	**No**	**PK**
Observation	Information		String	Yes	

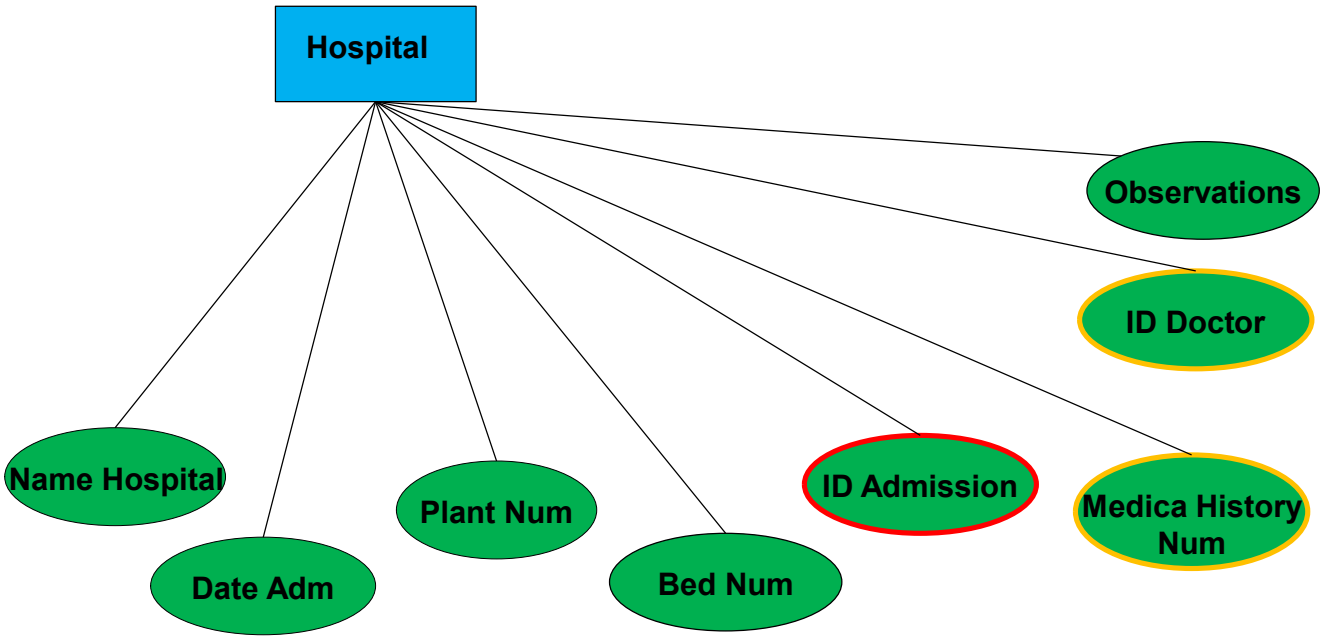
EXAMPLE – 3 (logical stage)



EXAMPLE – 3
Data Dictionary of Table DOCTORS

Attribute	Description	Data type	Domain	Null?	
ID Doctor	Id. doctor		AAAnnnnnnnnnn	No	PK
Name	Name doctor		string	No	
Speciality	Category		string	No	
Collegate num.	Code		Aannnn AAnn	No	
Position	Category		String	Yes	
Observation	Information		String	Yes	

EXAMPLE – 3 (logical stage)



EXAMPLE – 3
Data Dictionary of Table HOSPITAL

Attribute	Description	Data type	Domain	Null?	
Name Hospital	Name Hospital		string	No	
Date admission	Name doctor		string	No	
Plant num.	Number		string	Yes	
Bed num.	Number		string	Yes	
M. H. Number	**Id. Code**		**String**	**No**	**FK**
ID doctor	**Id. doctor**		**AAAnnnnnnnnnn**	**No**	**FK**
ID admission	**Id. Admission**		**Aannnn nnn nnn nnn**	**No**	**PK**
Observations	Information		String	Yes	

Field and Data Types: definition and properties

FIELD TYPE - the attributes of tables are represented as fields in **ERm**. For the purpose of this course, we can identify three type of fields:

- **Primary Key** - field used to uniquely identify a row in a table
- **Foreign Key** - field used to link several tables
- **Descriptive field** - any field that store data except the above indicate

After a field is created it is necessary to set what kind of field is it, to set its **DATA TYPE**.

Field and Data Types: definition and properties

DATA TYPE - it is a coding scheme recognized by DBMS for representing the data. This selection seeks to find the trade off among four objectives:

- **Minimize the storage space**
- **Represent all possible values of the field**
- **Improve data integrity of the field**
- **Support all data manipulation desired on the field**

There are a lot of different data types for databases, we are going to see some basic ones.

Data Types

- **VARCHAR2(n):** variable-length character with a maximum length of 4000 characters

  VARCHAR2(30)= the data can have a length of 30 character or less

- **CHAR:** variable-length character with a maximum length of 255 characters, by default is 1.

  CHAR(5)= the data can have a length of 5 character or less

- **LONG:** capable to store up to several gigabytes of one variable-length characters data.

  LONG= to hold medical instructions or customer comments.

Data Types

- **NUMBER:** store positive and negative values in a wide range
    - NUMBER(5)= to store an integer number of 5 digits, other databases use INT(5).
    - NUMBER(5,2)= to store a number of 5 digits with two decimal, other databases use REAL(5).
- **DATE:** capable to store date in a lot of formats, generally the format must be specified in the **domain column** of the data dictionary.
- **BLOB:** capable to store a binary large object, up to four gigabytes, this is used to digitalized documents (pictures, sound, video, pdf, etc …)

EXAMPLE – 3
Data Dictionary of Table PATIENS

Attribute	Description	Data type	Domain	Null?	
SS Number	Id. patient	CHAR(19)	nnnn nnnn nnnn nnnn	No	
Name	Name patient	CHAR(30)	string	No	
Address	Address patient	VARCHAR2(300)	string	Yes	
Region_Dec	Region patient	CHAR(100)	string	No	
Phone number	Phone patient	CHAR(12)	nnn nn nn nn	Yes	
M. H. Number	**Id. Code**	**CHAR(20)**	**String**	**No**	**PK**
Observation	Information	LONG	String	Yes	

EXAMPLE – 3
Data Dictionary of Table DOCTORS

Attribute	Description	Data type	Domain	Null?	
ID Doctor	**Id. doctor**	**CHAR(13)**	**AAAnnnnnnnnnn**	**No**	**PK**
Name	Name doctor	CHAR(30)	string	No	
Speciality	Category	CHAR(20)	string	No	
Collegate num.	Code	CHAR(11)	AAnnnn AAnn	No	
Position	Category	CHAR(20)	String	Yes	
Observation	Information	LONG	String	Yes	

EXAMPLE – 3
Data Dictionary of Table HOSPITAL

Attribute	Description	Data type	Domain	Null?	
Name Hospital	Id. doctor	CHAR(30)	string	No	
Date admission	Name doctor	CHAR(15)	dd/mm/yyyy hh:mm	No	
Plant num.	Number	CHAR(2)	string	Yes	
Bed num.	Number	CHAR(4)	string	Yes	
M. H. Number	**Id. Code**	**CHAR(20)**	**String**	**No**	**FK**
ID doctor	**Id. doctor**	**CHAR(13)**	**AAAnnnnnnnnnn**	**No**	**FK**
ID admission	**Id. Admission**	**CHAR(18)**	**Aannnn nnn nnn nnn**	**No**	**PK**
Observations	Information	LONG	String	Yes	

Primary Key assignment

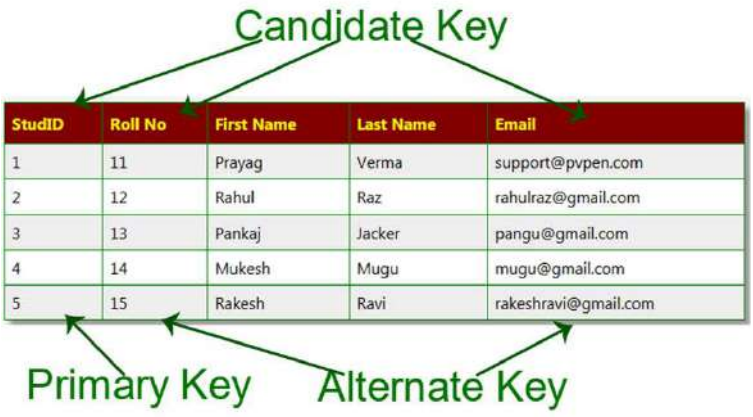
Choosing a correct primary key is one of the most important step in a good database design. Each table in a relational database needs a primary key. **The primary key ensures row-level accessibility**.

Primary Keys

StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Primary Key assignment

A table can have several attribute that can be considered as candidate keys, and the primary key is chosen from this set.



Remember, a primary can be made up using only a candidate key or it can be made up using several candidate keys. The first option is the best ones (**BREVITY**).

Primary Key assignment

The recommended characteristics of a candidate key to be a good primary key are:

1) NEVER NULL: no primary key value can be null.

2) SIMPLICITY: primary key without **special characters**, **blanks (embedded spaces)** or **uppercase and lowercase.**

.

Primary Key assignment

3) BREVITY: a primary key made up of just one candidate key (just one attribute, just one column). The use of this kind of primary key speeds up the data search.

Month	Name	Gender	Diagnosis	Treatment
May	Jessica	F	Allergy	Eye Drops
May	Sam	M	Allergy	Eye Drops
May	Wes	M	Cataract	Cataract Surgery
May	Rachel	F	Ptergyium	Eye Drops
May	Lily	F	Allergy	Eye Drops
May	Hannah	F	Cataract	Cataract Surgery
May	Denise	F	Allergy	Eye Drops
May	Sharon	F	Allergy	Eye Drops

In this case, due to none of attributes can be use as primary key since they do not meet the condition of uniquely identifying each row of the table.

Primary Key assignment

It is possible to made up a primary using the columns **Name**, **Gegender** and **Diagnosis**, but in this case that primary key does not meet the condition of brevity.

Month	Name	Gender	Diagnosis	Treatment
May	Jessica	F	Allergy	Eye Drops
May	Sam	M	Allergy	Eye Drops
May	Wes	M	Cataract	Cataract Surgery
May	Rachel	F	Ptergyium	Eye Drops
May	Lily	F	Allergy	Eye Drops
May	Hannah	F	Cataract	Cataract Surgery
May	Denise	F	Allergy	Eye Drops
May	Sharon	F	Allergy	Eye Drops

Primary Key assignment

But to make up a primary key using several candidate keys is not the best option for computational point of view.

The best option is created a new primary key.

ID TREATMENT	Month	Name	Gender	Diagnosis	Treatment
	May	Jessica	F	Allergy	Eye Drops
	May	Sam	M	Allergy	Eye Drops
	May	Wes	M	Cataract	Cataract Surgery
	May	Rachel	F	Ptergyium	Eye Drops
	May	Lily	F	Allergy	Eye Drops
	May	Hannah	F	Cataract	Cataract Surgery
	May	Denise	F	Allergy	Eye Drops
	May	Sharon	F	Allergy	Eye Drops

Primary Key assignment

4) PREFERRED DATA TYPE: a candidate key whose data type is numeric and also with fixed-length the best choice to be a primary key.

5) NONINDENTIFYING VALUE: no design or create a primary key with meaning, since it might become obsolete.

Songs in format mp3 → mp3 654a64a8

6) NEVER CHANGE: after assigning a primary key value, never change it. If your primary key is an attribute in other tables (foreign key). If you change the value of primary key, you have to change the associated foreign key value in many tables.

1º Examples- solution A

Candidate key	Candidate key	Candidate key	Candidate key		
Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

1º Examples- solution A

Primary key

Candidate key	Candidate key	Candidate key	Candidate key		
Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

41

1º Examples- solution A

Primary key

Candidate key	Candidate key	Candidate key	Candidate key		
Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

42

2º Examples- solution C

Primary key

Candidate key | Candidate key | Candidate key | Candidate key

Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

2º Example

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

2º Examples-solution A

Candidate key Candidate key

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

2º Examples-solution A

Primary key

Candidate key Candidate key

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

2º Examples-solution B

ID Personal	Name	FName	City	Age	Salary
	Smith	John	3	35	$280
	Doe	Jane	1	28	$325
	Brown	Scott	3	41	$265
	Howard	Shemp	4	48	$359
	Taylor	Tom	2	22	$250

3º Examples

employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep

3º Examples-solution

CK CK CK CK

	employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
▶	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
	1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
	1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
	1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
	1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
	1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
	1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
	1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
	1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep

3º Examples-solution

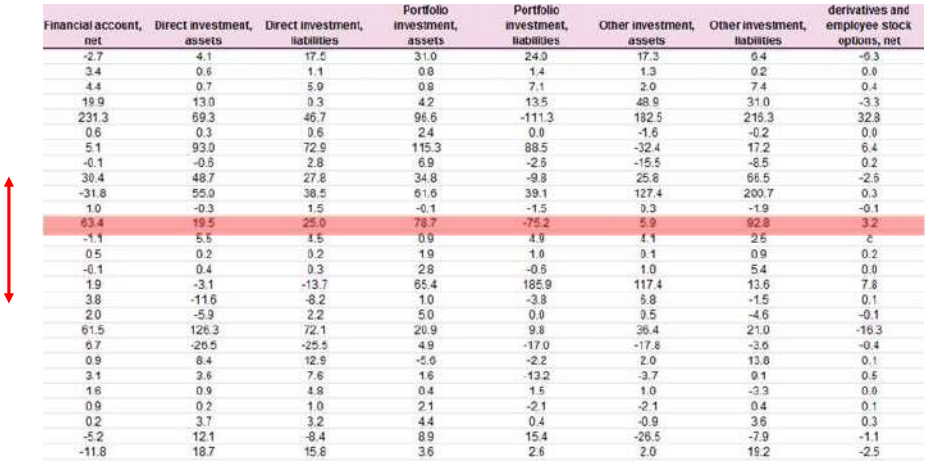
Primary key

CK CK CK CK

	employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
▶	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
	1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
	1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
	1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
	1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
	1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
	1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
	1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
	1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep

Creating Index

When we execute a query for searching data, our RDBMS analyses all the rows in the tables to find those data that match our request.



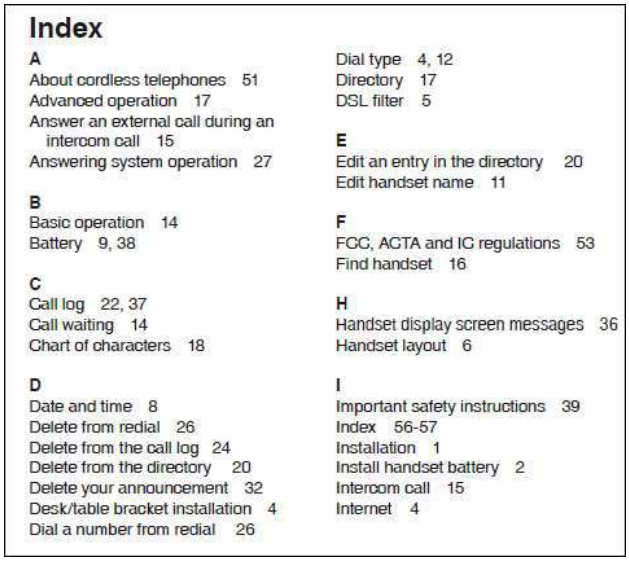
Financial account, net	Direct investment, assets	Direct investment, liabilities	Portfolio investment, assets	Portfolio investment, liabilities	Other investment, assets	Other investment, liabilities	derivatives and employee stock options, net
-2.7	4.1	17.5	31.0	24.0	17.3	6.4	-6.3
3.4	0.6	1.1	0.8	1.4	1.3	0.2	0.0
4.4	0.7	5.9	0.8	7.1	2.0	7.4	0.4
19.9	13.0	0.3	4.2	13.5	48.9	31.0	-3.3
231.3	69.3	46.7	96.6	-111.3	182.5	216.3	32.8
0.6	0.3	0.6	2.4	0.0	-1.6	-0.2	0.0
5.1	93.0	72.9	115.3	88.5	-32.4	17.2	6.4
-0.1	-0.6	2.8	6.9	-2.6	-15.5	-8.5	0.2
30.4	48.7	27.8	34.8	-9.8	25.6	66.5	-2.6
-31.8	55.0	38.5	61.6	39.1	127.4	200.7	0.3
1.0	-0.3	1.5	-0.1	-1.5	0.3	-1.9	-0.1
63.4	19.5	25.0	78.7	-75.2	5.9	92.8	3.2
-1.1	5.5	4.5	0.9	4.9	4.1	2.5	2
0.5	0.2	0.2	1.9	1.0	0.1	0.9	0.2
-0.1	0.4	0.3	2.8	-0.5	1.0	5.4	0.0
1.9	-3.1	-13.7	65.4	185.9	117.4	13.6	7.8
3.8	-11.6	-8.2	1.0	-3.8	5.8	-1.5	0.1
2.0	-5.9	2.2	5.0	0.0	0.5	-4.6	-0.1
61.5	126.3	72.1	20.9	9.8	36.4	21.0	-16.3
6.7	-26.5	-25.5	4.9	-17.0	-17.8	-3.6	-0.4
0.9	8.4	12.9	-5.6	-2.2	2.0	13.8	0.1
3.1	3.6	7.6	1.6	-13.2	-3.7	9.1	0.5
1.6	0.9	4.8	0.4	1.5	1.0	-3.3	0.6
0.9	0.2	1.0	2.1	-2.1	-2.1	0.4	0.1
0.2	3.7	3.2	4.4	0.4	-0.9	3.6	0.3
-5.2	12.1	-8.4	8.9	15.4	-26.5	-7.9	-1.1
-11.8	18.7	15.8	3.6	2.6	2.0	19.2	-2.5

As the records in the database grow, it is necessary to analyze more a more rows, which implies a decrease in the performance of our database searching data.

Creating Index

A solution for this problem is to create INDEXES, whose purpose is to make it faster to search through the tables and find the rows we want, the data we want.

So, an index of a database is like an index in a book.



Index

A
About cordless telephones 51
Advanced operation 17
Answer an external call during an intercom call 15
Answering system operation 27

B
Basic operation 14
Battery 9, 38

C
Call log 22, 37
Call waiting 14
Chart of characters 18

D
Date and time 8
Delete from redial 26
Delete from the call log 24
Delete from the directory 20
Delete your announcement 32
Desk/table bracket installation 4
Dial a number from redial 26
Dial type 4, 12
Directory 17
DSL filter 5

E
Edit an entry in the directory 20
Edit handset name 11

F
FCC, ACTA and IC regulations 53
Find handset 16

H
Handset display screen messages 36
Handset layout 6

I
Important safety instructions 39
Index 56-57
Installation 1
Install handset battery 2
Intercom call 15
Internet 4

Creating Index

The creation of INDEXES has pros and cons:

INDEXES reduce the data search time.

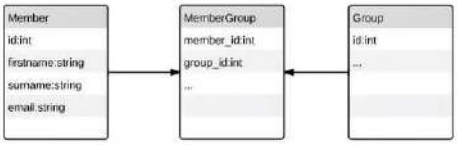
INDEXES have two drawback make it slower to add or update to existing rows for tables and occupy storage.

It is necessary a storage space to store the index tables.

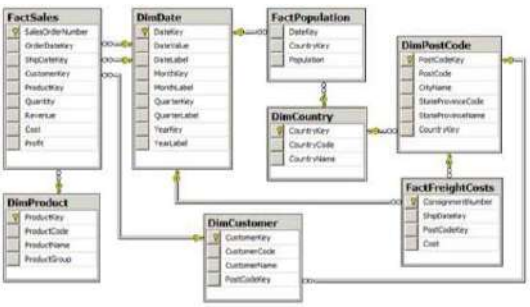
Creating Index

It is always necessary to add an index in our database?

This depends on how the complicate is the structure of our database and the amount of data stored in it.



From structural point of view does not need it.



From structural point of view needs it.

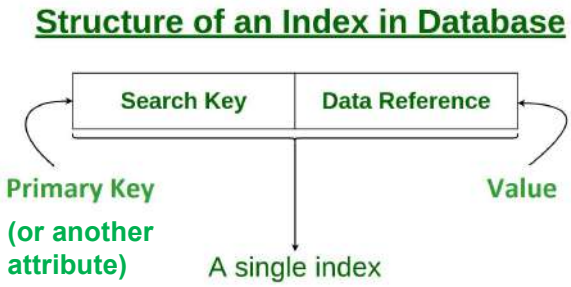
Creating Index

INDEXING is a technique of databases to efficiently retrieve record from the database based on some attributed on which the indexing has been done.

An index has a structure of table with two columns:

Search Key: it is a copy of the attribute (primary key or another attribute) of a table

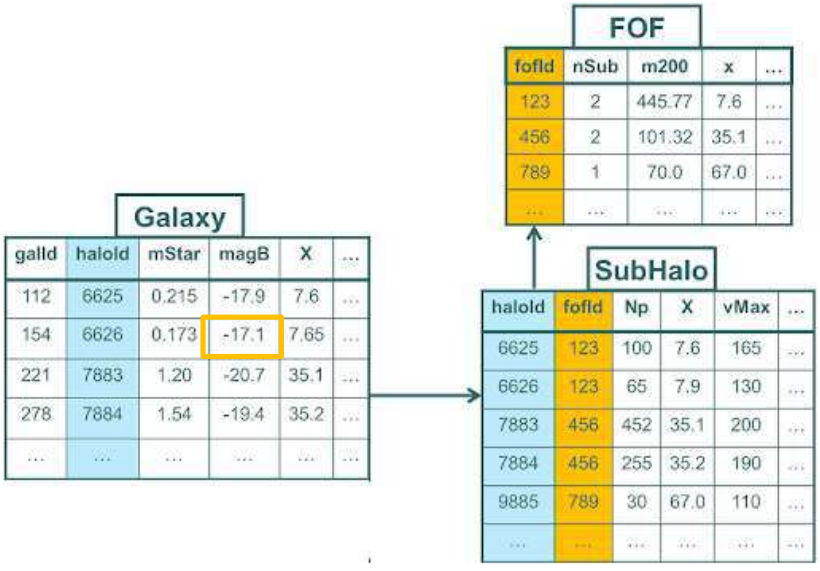
Data Reference or **Pointer**: it contains values that hold the location (of the disk block) where the records associated to that attribute are stored.



Creating Index

Without index

I want to know the value of **"magB"** with **"haloid=6626".**



Creating Index

With index

Search key | Pointer

haloid	Index_haloid
6625	cnKN4562
6625	ASJ65456
6626	DFG09743
6626	MNJ56646
7883	LKS65465
7883	JHGA9459
7884	ASHD354
…	AGA9844

I want to know the value of **"magB"** with **"haloid=6626".**

Galaxy

galId	haloId	mStar	magB	X	…
112	6625	0.215	-17.9	7.6	…
154	6626	0.173	-17.1	7.65	…
221	7883	1.20	-20.7	35.1	…
278	7884	1.54	-19.4	35.2	…
…	…	…	…	…	…

FOF

fofId	nSub	m200	x	…
123	2	445.77	7.6	…
456	2	101.32	35.1	…
789	1	70.0	67.0	…
…	…	…	…	…

SubHalo

haloId	fofId	Np	X	vMax	…
6625	123	100	7.6	165	…
6626	123	65	7.9	130	…
7883	456	452	35.1	200	…
7884	456	255	35.2	190	…
9885	789	30	67.0	110	…
…	…	…	…	…	…

The pointers locate the records of the attributes indexed in the tables of our database.

57

Creating Relationships

A relationships between two tables is created through matching an attribute that appears in both tables.
The main kind of relationship consider in this course is through a primary key, so the attribute shared is the primary key in one table (parent table) and is the foreign in another tables (associated tables).

PK

Table Employee

Employee_ID	Employee_Username	Employee_Name
1	7328	Steve
2	9843	Anna
3	2389	Ahmed
4	4739	Tina

Parent table

Table Salary

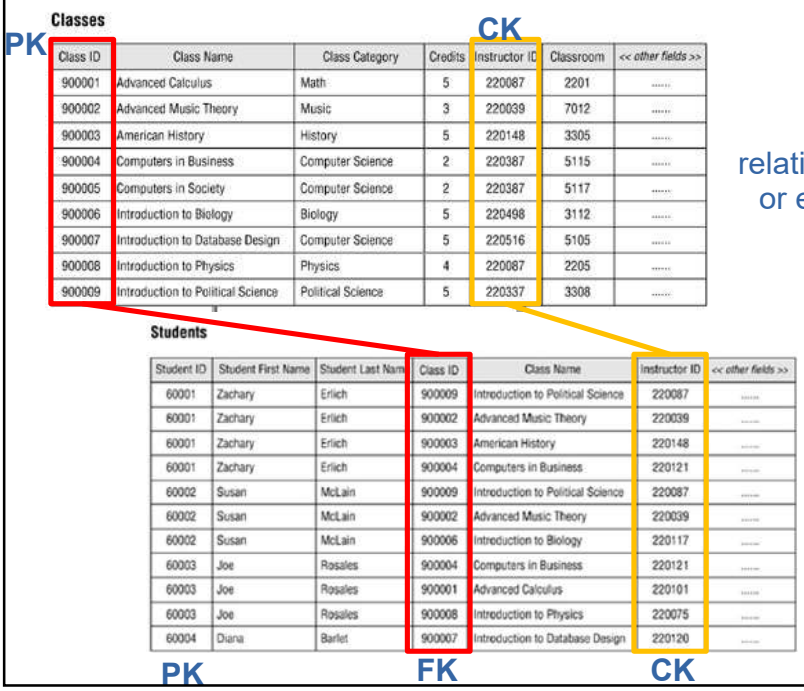
FK

Salary_ID	Date	Salary	Employee_ID
1	Dec-2017	£1500	1
2	Dec-2017	£1700	2
3	Dec-2017	£2000	3
4	Jan-2018	£1650	1

Associated table

58

Creating Relationships



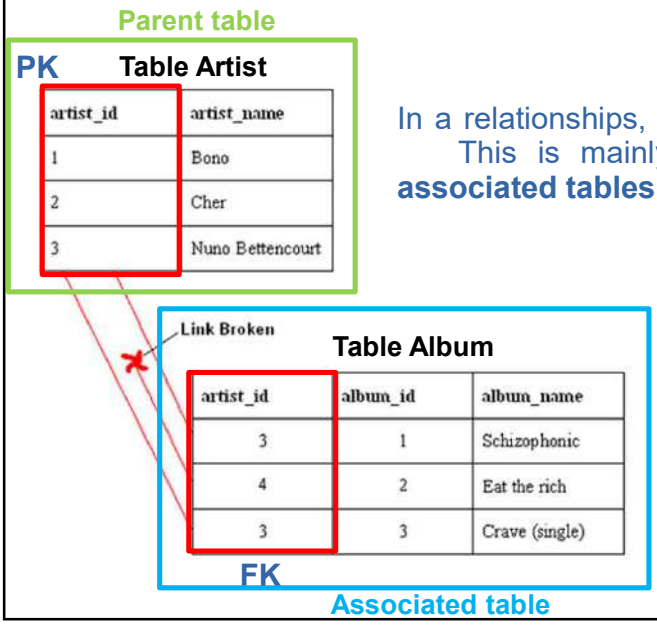
It is possible to create other kind relationships sharing other type of attributes or even to have two table related to each other in several different ways.

It is better just a relationship by each pair of tables.

The simpler is the database, the better is the performance of this one.

Referential integrity

Referential integrity refers to the accuracy and consistency of data within a relationship.



In a relationships, the data is linked between two (or more) tables. This is mainly achieved by the foreign key values in the **associated tables** reference a primary key values in **parent table**.

So, **referential integrity** requires that, whenever a foreign key value is used in **associated table** this value must reference a valid and existing value of a primary key in the **parent table**.

Practical Database

1

- from 13.04 al 16.04
- from 19.04 al 23.0a
- from 26.04 al 30.04 --- **1º Partial Exam (Units 1, 2, 3 and 4)**
- from 03.05 al 07.05
- from 10.05 al 14.05
- from 17.05 al 21.05 --- **2º Partial Exam (Unit 5-practical exam)**

2

Unit 5 – Data mining

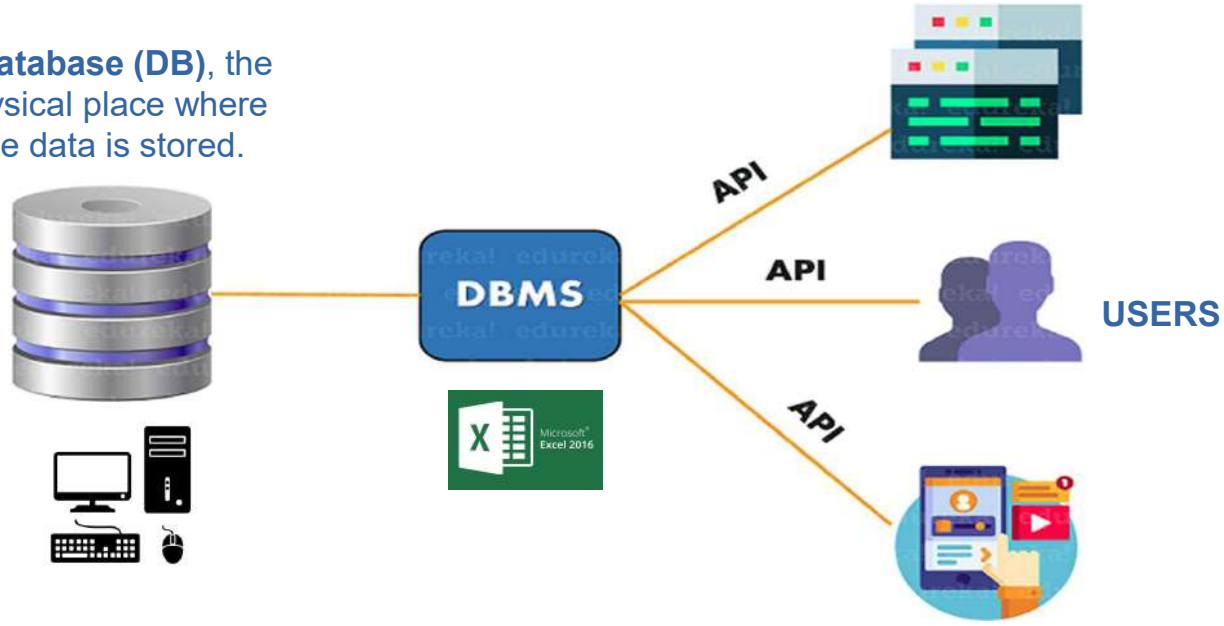
- Import and export of data.
- Creation of Relational Database in Excel.
- Concepts of relational algebra.
- Selective data extraction (filtering data).
- Statistical analysis of data.
- Multidimensional analysis (OLAP cubes).
- Dashboard design and creation.

3

4

¿Is EXCEL a DBMS?

A **database (DB)**, the physical place where the data is stored.



5

What is DBMS?-Characteristics

ABSTRACTION: details of physical storage are unknown for the normal user.

INDEPENDENCE: modify the database scheme without affecting the applications that depend on it

REDUNDANCY: the data is stored multiple time in different localization.

CONSISTENCY: the data repeated is updated simultaneously.

SECURITY: the data is safe against malicious activities.

INTEGRITY: the validity of the stored data is guaranteed

CONCURRENCY: a simultaneous access from different places is allowed without problem of consistency.

6

What is DBMS?-Functions

DATA DEFINITION: allow to define new kind of data by allowed users.

DATA MANIPULATION: allow the manipulation of data by allowed users.

MAINTENANCE Of PERMISSIONS AND ROLES: control and keep update the permissions and roles of the users.

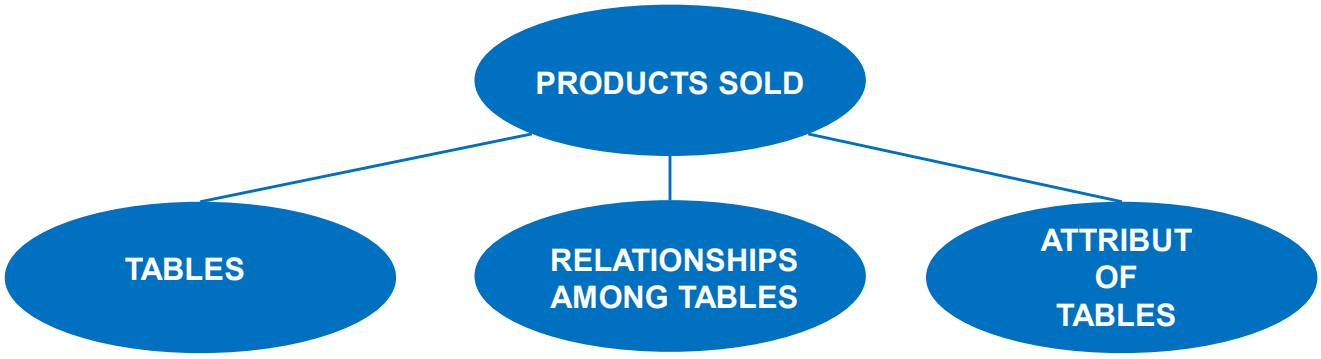
TRANSACTION MANAGEMENT: allow multiple data transaction in parallel done by different user at the same time, without losing of consistency.

7

Database example

EXAMPLE

We want to build a relational database for a company like **MERCADONA**, this database will focus on storing the data about the products sold.



8

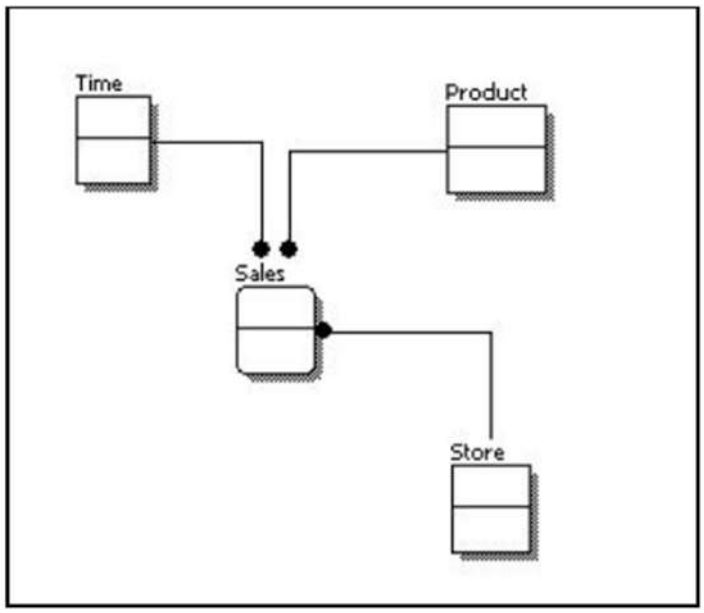
CONCEPTUAL stage

TABLES:

- **Time**:
- **Product**:
- **Sales**:
- **Store**:

RELATIONSHIPS:

- **Time - Sales**:
- **Product - Sales**:
- **Store - Sales:**



9

LOGICAL stage

TIME:

- **Date ID (ddmmyyyyhhmm)**
- **Date (dd/mm/yyyy hh:mm)**
- **Month**
- **Month Description**
- **Year**
- **Week**
- **Week Description**

PRODUCT

- **Product ID**
- **Product Description**
- **Category**
- **Category Description**
- **Unit Price**

STORE:

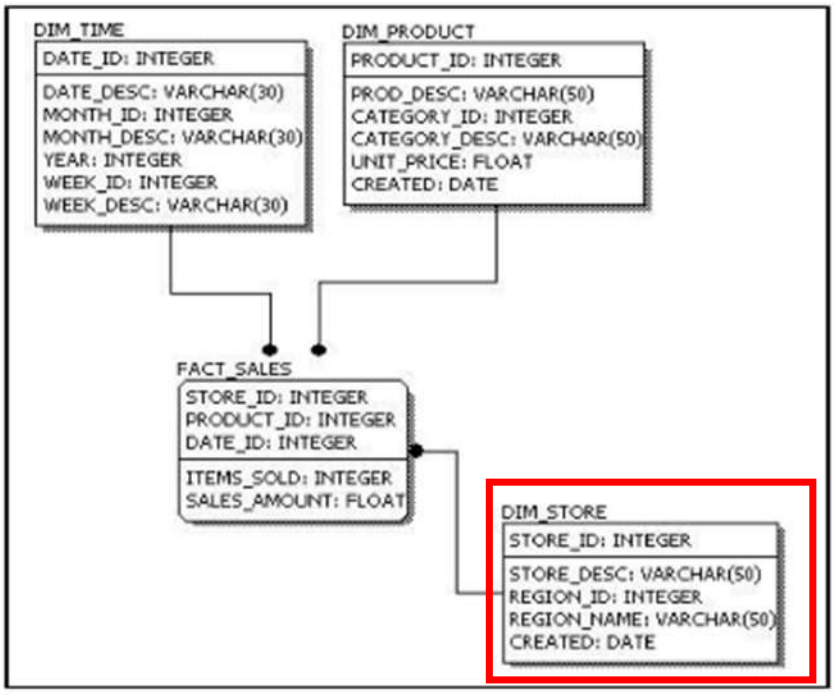
- **Store ID**
- **Store Description**
- **Region**
- **Region Description**
- **Created**

SALES:

- **Product ID**
- **Store ID**
- **Date ID**
- **Item Sold**:
- **Sales Amount**

10

PHYSICAL stage



11

Data Dictionary of table STORE

STORE

Attribute	Description	Data type	Domain	Null ?
Store ID	**Store Identification**	**INT(10)**	**number**	**No**
Store Description	Description	VARCHAR2(100)	string	Yes
Region	Place	INT(5)	number	No
Created	Date	CHAR(10)	dd/mm/yyyy	No

12

ISBN: 978-84-09-67484-8