DATABASE FUNDAMENTALS

SHARIZAN ABDUL JAMIL
SA'AMAH HASSAN
SITI FARTIMAH MOHAMED YUSOP

DATABASE FUNDAMENTALS

SHARIZAN ABDUL JAMIL
SA'AMAH HASSAN
SITI FARTIMAH MOHAMED YUSOP

Editor
Sa'amah Binti Hassan

Writer
Sharizan Binti Abdul Jamil
Sa'amah Binti Hassan
Siti Fartimah Binti Mohamed Yusop

Designer
Sharizan Binti Abdul Jamil

Published by:

POLITEKNIK METrO TASEK GELUGOR
No. 25, Jalan Komersial 2
Pusat Komersial Tasek Gelugor
13300 Tasek Gelugor
Pulau Pinang, Malaysia
Tel: 04-5732789
Fax: 04-5732087
Website: www.pmtg.edu.my

Cataloguing-in-Publication Data

Perpustakaan Negara Malaysia

A catalogue record for this book is available from the National Library of Malaysia

eISBN 978-967-2744-19-1

Synopsis

Database Fundamentals course engages students to apply business scenarios and create a data model - a physical database using SQL (Structured Query Language). Basic SQL syntax and the rules for constructing valid SQL statements are reviewed. This course culminates with a mini project that challenges students to design, implement, and demonstrate a database solution for a business or organization.

CONTENTS

CHAPTER 1

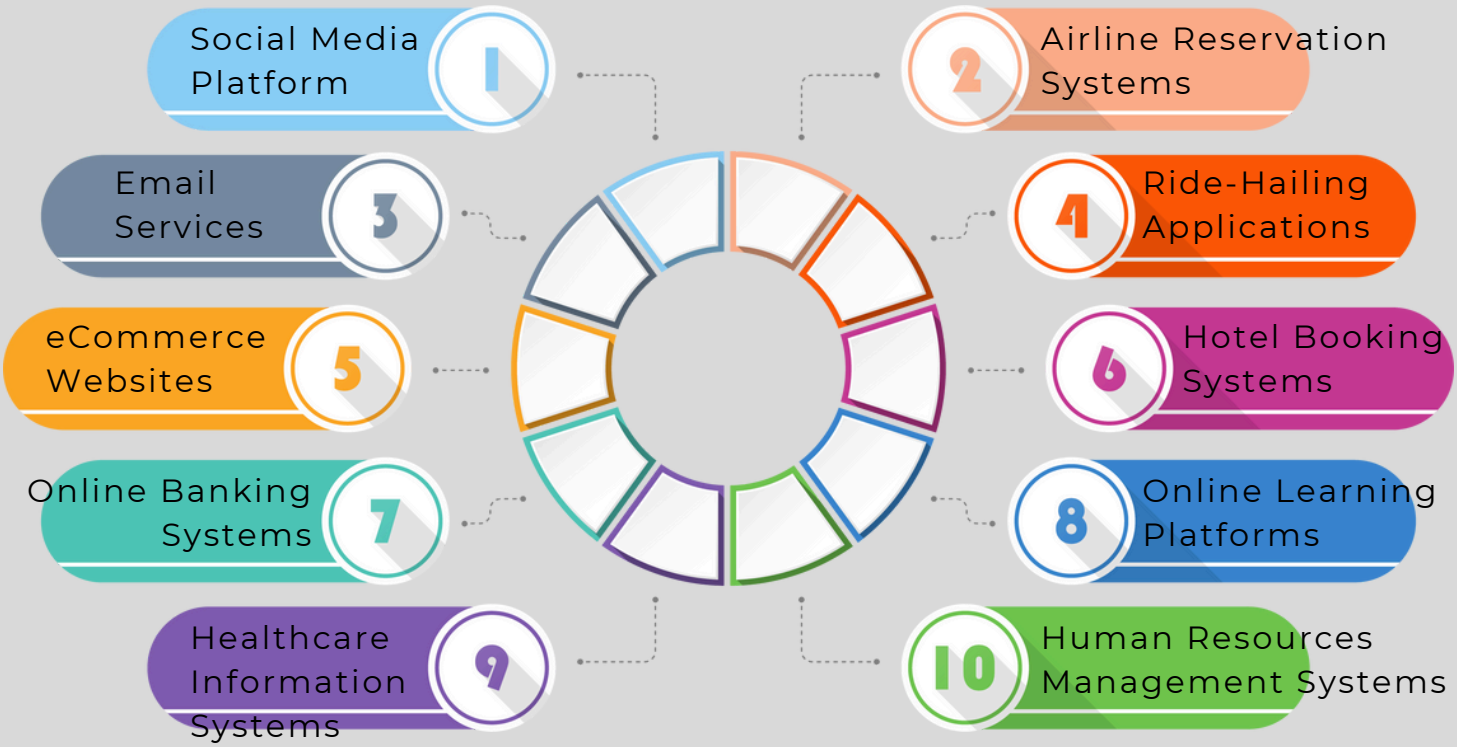
FUNDAMENTALS OF DATABASE MANAGEMENT SYSTEM

1.1 CONCEPT OF DATABASE

1.1.1 Definition of database

A database is an organized collection of structured information or data typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS along with the applications that are associated with them, are referred to as a database system often shortened to just database.

1.1.2 Database in the business world



1.1.3 Importance of databases to everyday life

Databases are important tools for storing and managing data. They allow to keep track of information and access it quickly and easily. It also helps to organize scattered data and help to make decisions for the future.

1.1 CONCEPT OF DATABASE

1.1.4 Database development process

Requirement Analysis

The database designers have to interview the customers (database users) to understand the proposed system, obtain and document the data and functional requirements. The requirements include data items, attributes, constraints, and relationships that hold between data.

Database Design

Starts with a conceptual data model and produces a specification of a logical schema; this will determine the specific type of database system (network, relational, object-oriented) that is required.

Implementation

Involves the construction of a database according to the specification of a logical schema. This will include the specification of an appropriate storage schema, security enforcement, external schema and so on.

1.1.5 Sharing concept of data in database

Data sharing gives multiple users or applications simultaneous, consistent, and high-fidelity access to the same datasets. That there is some software locking mechanism that prevents the same set of data from being changed by two people at the same time.

1.1 CONCEPT OF DATABASE

1.1.6 Properties of database

Completeness

- Ensures that users can access the data they want includes ad hoc queries, which would not be explicitly given as part of a statement of data requirements.
- Database has to support the requirements.
- It requires the complete understanding of database structure, relationship and constraint.

Integrity

- Ensures that data is both consistent (no contradictory data) and correct (no invalid data) and ensures that users trust the database.
- Database integrity ensures that data entered into the database is accurate, valid and consistent.
- Any applicable integrity constraint and data validation rules must be satisfied before permitting a change to the database.

Flexibility

- Ensures that a database can evolve (without requiring excessive effort) to satisfy changing user requirements.
- Ability to upgrade or change the functionality of database up to the current need.
- Ability to support wide area of data types.

Efficiency

- Ensures that users do not have unduly long response times when accessing data.
- The database should be able to perform effectively.
- The designer has to choose the right DBMS, the right access path in order to improve the efficiency.

Usability

- Ensures that data can be accessed and manipulated in ways which match user requirements.
- The database design significantly impacts the quality and usability of the data.
- A database design that is not properly normalized will introduce data update anomalies and data errors.
- A poorly designed database may place the entire organization at risk due to the incomplete or incorrect information.

1.2 CONCEPT OF DATABASE MANAGEMENT SYSTEM (DBMS)

1.2.1 DBMS Purpose

A database management system (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database.

The purpose of a DBMS is to organize data in a database and provide a way to store up and retrieve database information that is both convenient and efficient. The DBMS serves as the intermediary between the user and the database.

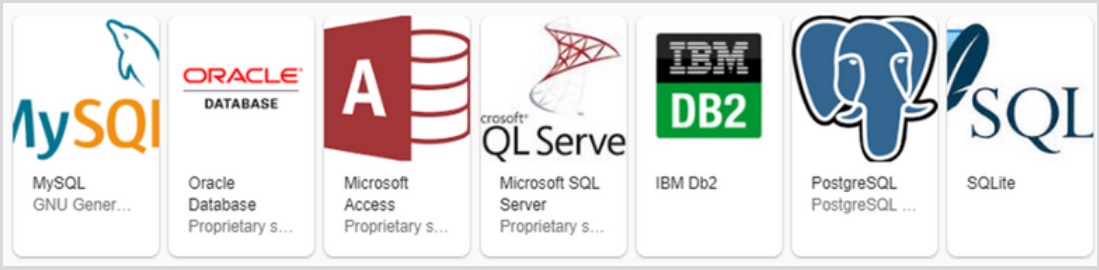
1.2.2 Various Common of DBMS

Server DBMS

- Oracle DBMS
- Microsoft SQL Server
- IBM DB2
- MariaDB
- MySQL, Firebird, PostgreSQL (Significant Open Source)
- MySQL(Open Source)

Desktop DBMS

- Microsoft Access
- FoxPro, Paradox, Approach, FileMaker Pro
- Base (LibreOffice
- Corel Paradox
- SQLite



1.2 CONCEPT OF DATABASE MANAGEMENT SYSTEM (DBMS)

1.2.3 Traditional Approach to information processing

Conventionally, the data were stored and processed using traditional file processing systems. In these traditional file systems, each file is independent of other file, and data in different files can be integrated only by writing individual program for each application. Any change to the data requires modifying all the programs that uses the data. This is because each file is hard coded with specific information like data type, data size etc.

1.2 CONCEPT OF DATABASE MANAGEMENT SYSTEM (DBMS)

1.2.4 Disadvantages of the traditional approach to information processing

Poor data security because anyone can easily modify and change the data stored in the files.

Data redundancy may happen when data is stored more than once in different files, that means duplicate data may occur in all these files.

Data isolation happen while files can be in different formats. If you want to extract data from two file, then you are required to which part of the file is needed and how they are related to each other.

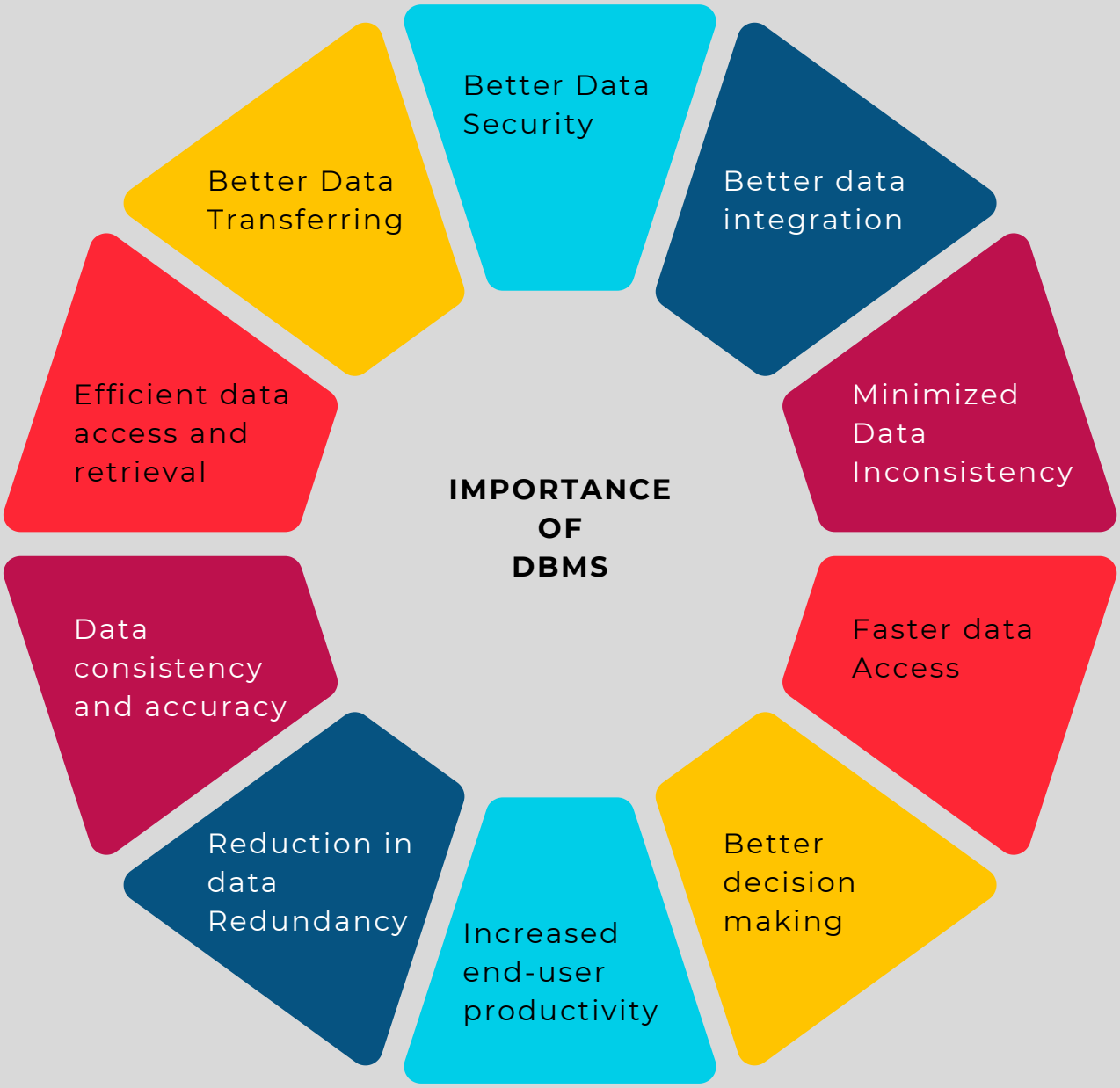
Program/data dependence refers to the coupling of data stored in files and the specific programs required to update and maintain those files such that changes in programs require changes to the data. In a tradition file environment, any change in a software program could require a change in the data accessed by that program.

Lack of flexibility
A traditional file system can deliver routine scheduled reports after extensive programming efforts, but it cannot deliver ad-hoc reports or respond to unanticipated information requirements in a timely fashion.

Concurrent access anomalies mean that is not easy to access data in a desired or efficient way.

1.2 CONCEPT OF DATABASE MANAGEMENT SYSTEM (DBMS)

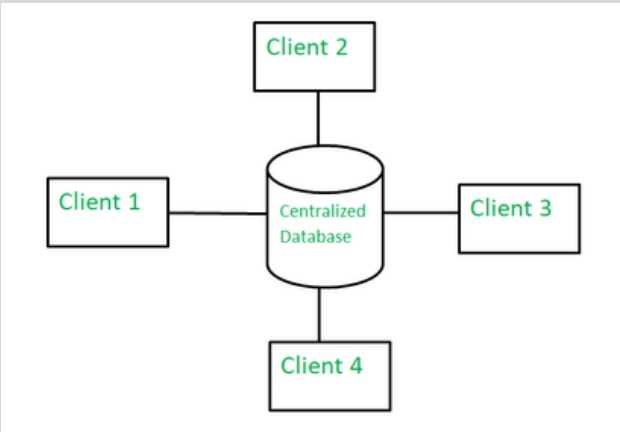
1.2.5 Importance of DBMS

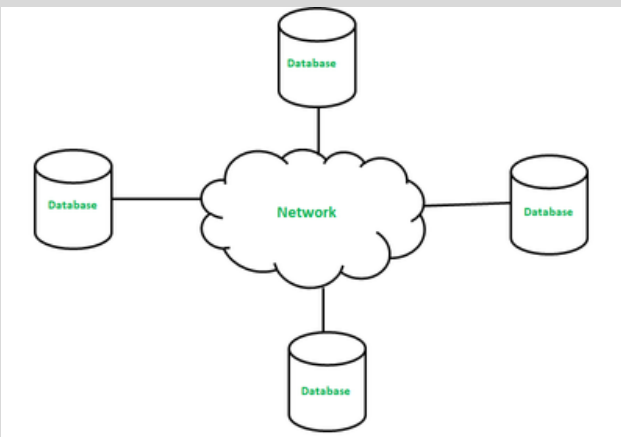


1.2 CONCEPT OF DATABASE MANAGEMENT SYSTEM (DBMS)

1.2.6 Database architecture

Basis of Comparison	Centralized Database	Distributed Database
Definition	It is a database that is stored, located as well as maintained at a single location only.	It is a database that consists of multiple databases which are connected with each other and are spread across different physical locations.
Advantages	1. Integrity of data 2. Security 3. Easy access to all information 4. Data is easily portable	1. High performance because of the division of workload. 2. High availability because of the readiness of available nodes to do work. 3. Independent nodes and better control over resources
Disadvantages	1. Data searching takes time 2. In case of failure of a centralized server, the whole database will be lost. 3. If multiple users try to access the data at the same time then it may create issues.	1. It is quite large and complex so difficult to use and maintain. 2. Difficult to provide security 3. Issue of data integrity 4. Increase in storage and infrastructure requirements 5. Handling failures is a quite difficult task

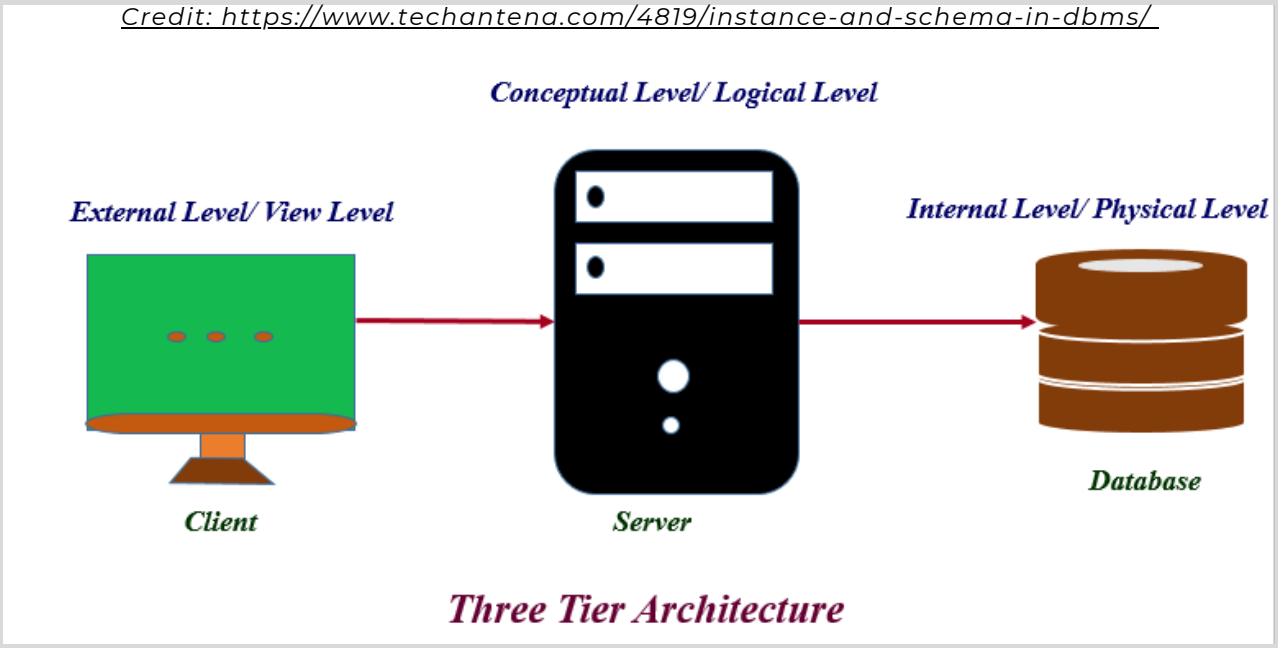




1.3 DATA MODEL IN DBMS

1.3.1 DBMS Architecture

Credit: https://www.techantena.com/4819/instance-and-schema-in-dbms/



1. External level

It is also called view level. Several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping. The user doesn't need to know the database schema details such as data structure, table definition etc. user is only concerned about data which is what returned back to the view level after it has been fetched from database (present at the internal level). External level is the top level of the Three Level DBMS Architecture.

2. Conceptual level

It is also called logical level. The whole design of the database such as relationship among data, schema of data etc. are described in this level. Database constraints and security are also implemented in this level of architecture. This level is maintained by database administrator (DBA).

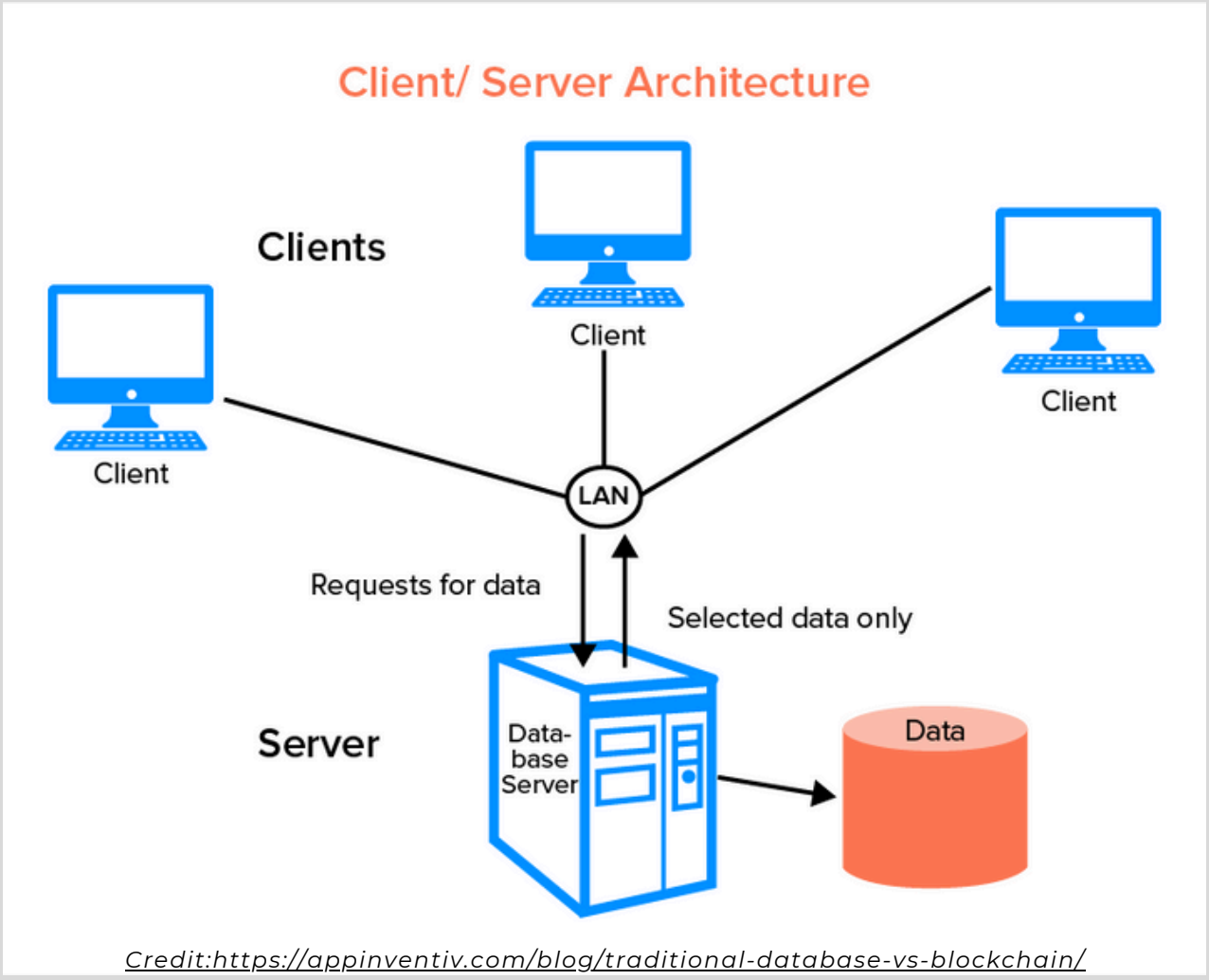
3. Internal level

This level is also known as physical level. This level describes how the data is actually stored in the storage devices. This level is also responsible for allocating space to the data. This is the lowest level of the architecture.

1.3 DATA MODEL IN DBMS

1.3.2 Client-server Architecture of DBMS

Client-server architectures are one of the most popular architectures used in database management systems (DBMS). This architecture integrates the use of multiple servers to ensure the efficient delivery of data between users, applications, and databases. The client sends a request for the data stored in the database and the server responds to the request by sending it back to the user.



Credit:https://appinventiv.com/blog/traditional-database-vs-blockchain/

1.3 DATA MODEL IN DBMS

1.3.3 Categories of DBMS

Desktop database

- Microsoft Access
- FoxPro
- FileMaker Pro
- Paradox
- Lotus

Server database

- Oracle
- Microsoft SQL Server
- IBM
- DB2

1.3.4 Benefits of using desktop database and server solution

Desktop Database

- User-friendly
- Easy management
- Low running costs
- Easy to use

Server Database

- Flexibility
- Security
- Scalability
- Wider integration
- Powerful performance

1.3 DATA MODEL IN DBMS

1.3.5 DBMS users

DATABASE ADMINISTRATION

- The DBA is a person or a group of persons who is responsible for the management of the database.
- The DBA is responsible for authorizing access to the database by grant and revoke permissions to the users, for coordinating and monitoring its use, managing backups and repairing damage due to hardware and/or software failures and for acquiring hardware and software resources as needed.

APPLICATION PROGRAMMERS

- Application Programmers are responsible for writing application programs that use the data base.
- These programs could be written in General Purpose Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc. to manipulate the database.
- These application programs operate on the data to perform various operations such as retaining information, creating new information, deleting or changing existing information.

END USERS

- End Users are the people who interact with the database through applications or utilities.

- **Casual End Users:** These Users occasionally access the database but may need different information each time. They use sophisticated database Query language to specify their requests. For example: High level Managers who access the data weekly.
- **Native End Users:** These users frequently query and update the database using standard types of Queries. The operations that can be performed by this class of users are very limited and effect precise portion of the database. For example: Reservation clerks for airlines/hotels check availability for given request and make reservations.
- **Standalone end Users/Online End Users:** Those End Users who interact with the database directly via online terminal or indirectly through Menu or graphics-based Interfaces. For example: User of a text package, library management software that store variety of library data such as issue and return of books for fine purposes.

Exercises:

1. Identify a definition of database.
2. State FIVE (5) examples of Database Management System (DBMS).
3. Identify FIVE (5) importances of having a DBMS.
4. Identify FIVE (5) properties of database.
5. Differentiate Centralized Database and Distributed Database.

1. Introduction to database concepts

CHAPTER 2

RELATIONAL DATA MODEL

2.1 RELATIONAL DATABASES

2.1.1 Relational Database Management System (RDBMS)

A relational database management system (RDBMS) is a program that allows you to create, update, and administer a relational database. Most relational database management systems use the SQL language to access the database.

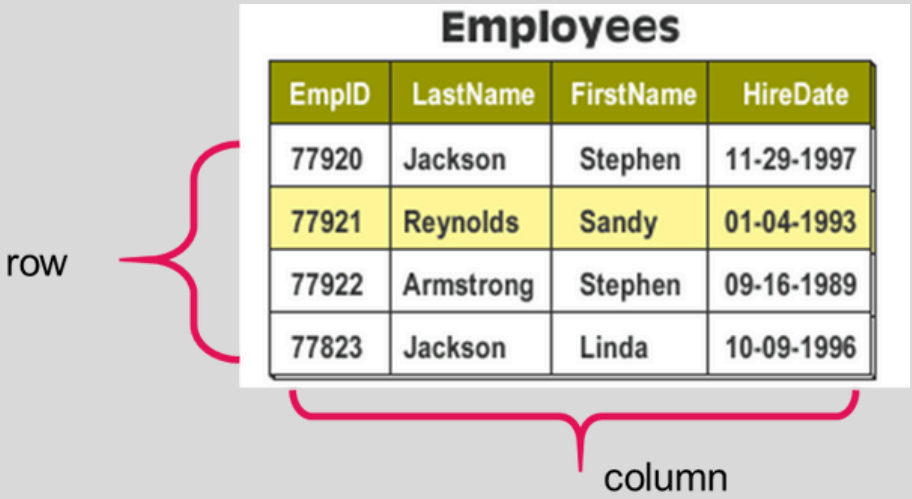
Relational database is visualized by using table that consist of rows and data column.

2.1.2 RDBMS Package

2.1 RELATIONAL DATABASES

2.1.3 Components of a relational database structural terminology

Relation (Table): A table is a fundamental component of a relational database. It represents a collection of related data organized in rows and columns. Each table has a name and consists of columns (fields) and rows (records) that store the actual data.



- **Tuple (Row of record):** Represents a single instance or entry of data within a table. It contains values for each column that correspond to a specific record or entity.

Student

studentID	name	birthDate	address	telNo	department
A3517	Shamim	03.05.92	No 534 Tmn Murni	0131112233	Jtmk
A4350	Chong	22.07.92	123 Jalan Delima	0115671234	Jtmk
A1234	Halim	06.08.92	21 Jln Meru	0123456789	Jtmk

Tuple

2.1 RELATIONAL DATABASES

2.1.3 Components of a relational database structural terminology

- **Cardinality of a Relation (Number of rows):** The number of tuples(rows) its contains.

Student

studentID	name	birthDate	address	telNo	department
A3517	Shamim	03.05.92	No 534 Tmn Murni	0131112233	Jtmk
A4350	Chong	22.07.92	123 Jalan Delima	0115671234	Jtmk
A1234	Halim	06.08.92	21 Jln Meru	0123456789	Jtmk

Cardinality = 3

- **Attribute (Column or field):** Represents a specific data element within a table. It defines the type of data that can be stored, such as text, numbers, dates, or binary data. Each column has a name and a specific data type associated with it.

Attribute

Student

studentID	name	birthDate	address	telNo	department

- **Degree of a relation (Number of columns):** The number of attributes it contains.

studentID	name	birthDate	address	telNo	department
A3517	Shamim	03.05.92	No 534 Tmn Murni	0131112233	Jtmk
A4350	Chong	22.07.92	123 Jalan Delima	0115671234	Jtmk
A1234	Halim	06.08.92	21 Jln Meru	0123456789	Jtmk

Degree = 6

2.1 RELATIONAL DATABASES

2.1.3 Components of a relational database structural terminology

- **Primary Key (Unique identifiers):** A unique identifier for each row in a table. It ensures the uniqueness and integrity of data within the table. A primary key column cannot contain duplicate or null values.

- **Domain (Pool of values of specific attributes of relation):** Set of possible values for an attribute. Each simple attribute of entity is associated with value set (domain of values). It specifies the sets value that may be assigned to that attribute for each individual entity.

[(studentID, A1234), (name, Halim), (birthdate, 6.8.92), (address, 21 Jln Meru), (telNo, 012345678), (department, Jtmk)]

Student

studentID	name	birthDate	address	telNo	department
A1234	Halim	06.08.92	21 Jln Meru	0123456789	Jtmk

- The domain for the date of birth (*birthdate*) contains sub domain : ***day, month, year***.

2.1 RELATIONAL DATABASES

2.1.4 Application areas of RDBMS

The RDBMS typically provides data dictionaries and metadata collections useful in data handling. These programmatically support well-defined data structures and relationships. RDBMS use complex algorithms that support multiple concurrent user access to the database, while maintaining data integrity. Security management, which enforces policy-based access, is yet another overlay service that the RDBMS provides for the basic database as it is used in enterprise settings. RDBMS support the work of database administrators (DBAs) who must manage and monitor database activity. Relational database management systems are central to key applications, such as banking ledgers, travel reservation systems and online retailing.

2.1.5 Properties of tables in a relational database

- A table has a name that is distinct from all other tables in the database.
- There are no duplicate rows; each row is distinct.
- Entries in columns are atomic. The table does not contain repeating groups or multivalued attributes.
- Entries from columns are from the same domain based on their data type including:
  - number (numeric, integer, float)
  - character (string)
  - date
  - logical (true or false)
- Operations combining different data types are disallowed.
- Each attribute has a distinct name.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.

2.1 RELATIONAL DATABASES

2.1.6 Characteristics of relation scheme

Relation Schema is a named of a relation defined by a set of attributes and domain name pairs.

Relation name: Is distinct from all other relation names in relational schema. Cannot have two Student relation in the database.

Attribute name: Each attribute has a distinct name. Order of attributes has no significance. Order of tuples has no significance, theoretically.

Domains: Values of an attribute are all from the same domain.

Common convention:

RelationName (attribute_1, attribute_2,....., attribute_n)

2.1.7 Relation Instance

Relation instance is a finite set of tuples in the RDBMS system at any given time.

- The tuples in an instance is known as extension of a relation, which usually changes.
- The relation instance change when tuple is updated, deleted or inserted.
- Example of instance for corresponding database schema is shown below :

LECTURER

staffNo	icNo	name	position	dept	telNo
1001	1234567	Ah Lek	Lecturer	JTMK	1111
1002	2345678	Ali Mamat	Professor	JMSK	2222
1003	3456789	Fatty Lim	Lecturer	JPA	3333

1. *(1001, 1234567, Ah Lek, Lecturer, JTMK, 1111)*
2. *(1002, 2345678, Ali Mamat, Professor, JMSK, 2222)*
3. *(1003, 3456789, Fatty Lim, Lecturer, JPA, 3333)*

2.1.8 Relation Keys

Refers to the important attribute in an entity. Determine the uniqueness of a row in given table. Identifiers for each row. An attribute or more than one attributes can be declared as keys depending on situations.

Types of keys:

- **Primary Key (PK):** Is an attribute that uniquely identify each row. Each table must have primary key. Cannot be NULL value to maintain Entity Integrity. Primary key is the one that officially declared as the row identifier inside a specific table.

- **Candidate Key (CK):** Is a single field or the least combination of fields that uniquely identifies each record in the table. There can be more than one candidate keys in a relation. However, the instance of relation cannot be used to prove attributes or combination of attributes is a candidate key.

  Criteria for the candidate keys:

  1. It must contain unique values.
  2. It must not contain null values.
  3. It contains the minimum number of fields to ensure uniqueness.
  4. It must uniquely identify each record in the table.

- **Foreign Key (FK):** Is an attribute whose value match the primary key values in related tables. Creates a relationship between relations.

2.1 RELATIONAL DATABASES

2.1.9 Important of Integrity Rules

To have a good design, a database must have integrity rules. Constraint or restriction that apply to all instances of the database.

Integrity rules consists of:

- **Entity Integrity**

Requirement: All Primary Key entries are unique, and no part of a primary key may be NULL

Purpose: Guarantee that each entity will have a unique identity and ensures that for each key value can properly refer primary key values.

- **Referential Integrity**

Requirement: A Foreign Key may have either a NULL entry, as long as it is not a part of its table's primary key or an entry that matches the primary key value in a table to which is related. (Every non null foreign key value must reference an existing primary key value).

Purpose: Makes it possible for an attribute NOT to have a corresponding value, but will be impossible to have an invalid entry. The enforcement of the referential integrity rules makes it impossible to delete a row in one table whose primary keys has mandatory matching foreign key values on another table.

2.2 OPERATORS OF RELATIONAL ALGEBRA

2.2.1 The Relational Algebra

- The formal description of how a relational database operates.
- An interface to the data stored in the database itself.
- The mathematics which underpins SQL operations.
- Operators in relational algebra are not necessarily the same as SQL operators, even if they have the same name. For example, the SELECT statement exists in SQL, and also exists in relational algebra. These two uses of SELECT are not the same.
- The DBMS must take whatever SQL statements the user types in and translate them into relational algebra operations before applying them to the database.

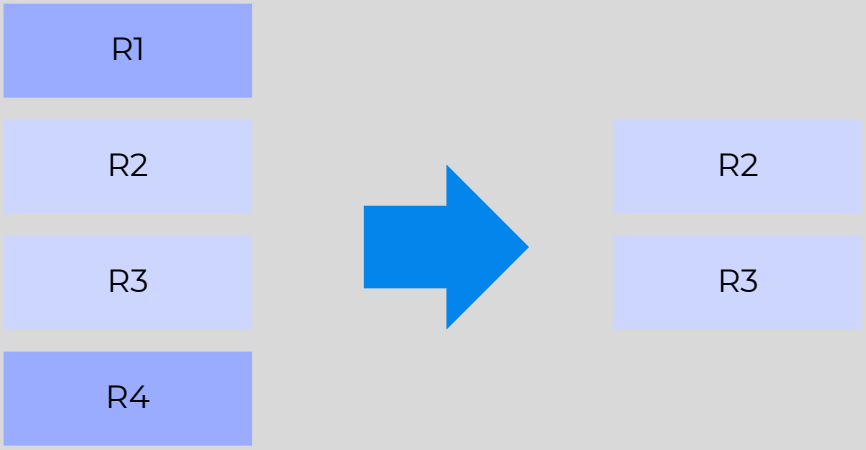
2.2.2 The fundamental operators used to retrieve information from a relational database

2.2.2 The fundamental operators used to retrieve information from a relational database

RESTRICT (SELECT)

Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation (or table) which satisfy the given condition.

The SELECT operation can be viewed as a horizontal filter of the relation.



- Syntax: *σ predicate/condition (relation/table name)*
- Meaning: Choose expressions from relation that fulfil the conditions in the predicate and write it in a new relation as an output.
- *Predicate/condition* is a 'restriction' statement in the form of X Θ Y where
  - X is the attribute name in H
  - Y is the value from attribute domain X
  - Θ is the comparison operator such as =,<,>,<=,>=,~= or the logical operator such as ~(NOT), ^ (AND), v (OR) or the combinations between both operations.

2.2 OPERATORS OF RELATIONAL ALGEBRA

2.2.2 The fundamental operators used to retrieve information from a relational database

PROJECT

Project operator is denoted by *pi* (**π**) symbol and it is used to select desired attributes (or columns) from a table (or relation).

The PROJECT operation can be viewed as the vertical filter of the relation.

Syntax:
πcolumn_name1, column_name2 …..column_nameN(relation/table name)

2.2.2 The fundamental operators used to retrieve information from a relational database

JOIN

JOIN allows information to be combined from two or more tables. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

- Following are the types of JOIN:
  - Natural
  - Inner
  - Outer

NATURAL JOIN

A natural join links tables by selecting only the rows with common values in their common attribute(s).

- We can perform a Natural Join only if there is at least one common attribute that exists between two relations.
- In addition, the attributes must have the same name and domain.
- Natural join does not use any comparison operator.

HoD

Dept	Head
CS	Alex
ME	Maya
EE	Mira

Courses

CID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME

HoD ⋈ Course

Dept	Head	CID	Course
CS	Alex	CS01	Database
ME	Maya	ME01	Mechanics

INNER JOIN

An inner join is a join that only returns matched records from the tables that are being joined.

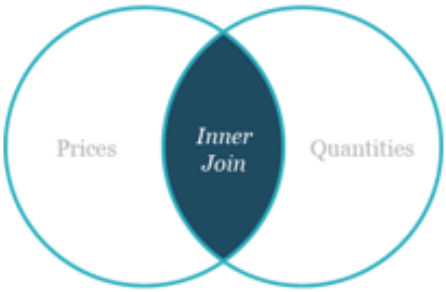
- The INNER JOIN creates a new result table by combining column values of two tables (table A and table B) based upon the join-predicate.
- The query compares each row of table A with each row of table B to find all pairs of rows which satisfy the join-predicate.
- When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	$3
Avocados	$4
Kiwis	$2
Onions	$1
Melons	$5
Oranges	$5
Tomatoes	$6

TABLE 2: QUANTITIES

PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92



QUERY RESULT FOR INNER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	$3	45
Avocados	$4	63
Kiwis	$2	19
Onions	$1	20
Melons	$5	66

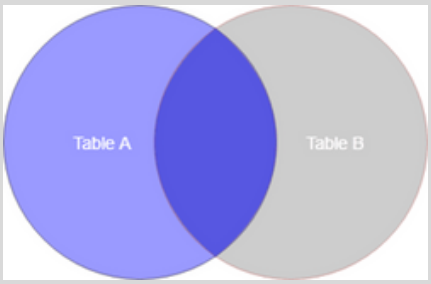
OUTER JOIN

In an outer join, the matched pairs would be retained, and any unmatched values in the other table would be left null.

- Notice that much of the data is lost when applying a join to two relations.
- In some cases, this lost data might hold useful information.
- An outer join retains the information that would have been lost from the tables, replacing missing data with nulls.
- There are three forms of the outer join, depending on which data is to be kept:
  - left outer join (keep data from the left-hand table)
  - right outer join (keep data from the right-hand table)
  - full outer join (keep data from both tables).

LEFT OUTER JOIN (R ⟕ S)

- All the tuples from the Left relation, R, are included in the resulting relation.
- If there are tuples in R without any matching tuple in the Right relation S, then the S-attributes of the resulting relation are made NULL.



Courses

A	B
100	Database
101	Mechanics
102	Electronics

⟕

HoD

A	B
100	Alex
102	Maya
104	Mira

Courses ⟕ HoD

A	B	C	D
100	Database	100	Alex
101	Mechanics	Null	Null
102	Electronics	102	Maya

2.2 OPERATORS OF RELATIONAL ALGEBRA

RIGHT OUTER JOIN (R ⟖ S)

- All the tuples from the Right relation, S, are included in the resulting relation.
- If there are tuples in S without any matching tuple in R, then the R-attributes of resulting relation are made NULL.

Courses

A	B
100	Database
101	Mechanics
102	Electronics

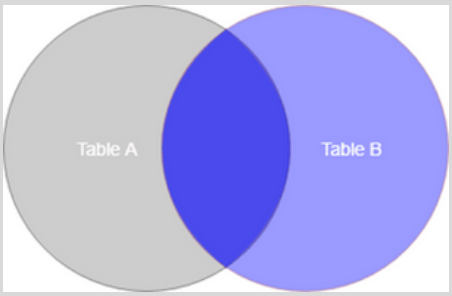
⟖

HoD

A	B
100	Alex
102	Maya
104	Mira

Courses ⟖ HoD

A	B	C	D
100	Database	100	Alex
102	Electronics	102	Maya
Null	Null	104	Mira

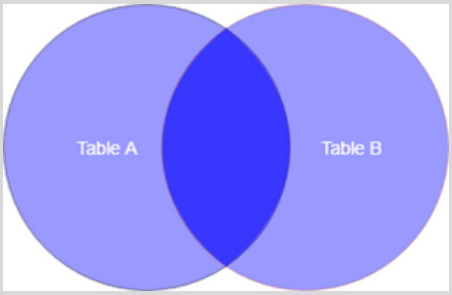


FULL OUTER JOIN (R ⟗ S)

- All the tuples from both participating relations are included in the resulting relation.
- If there are no matching tuples for both relations, their respective unmatched attributes are made NULL.

Courses ⟗ HoD

A	B	C	D
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya
---	---	104	Mira

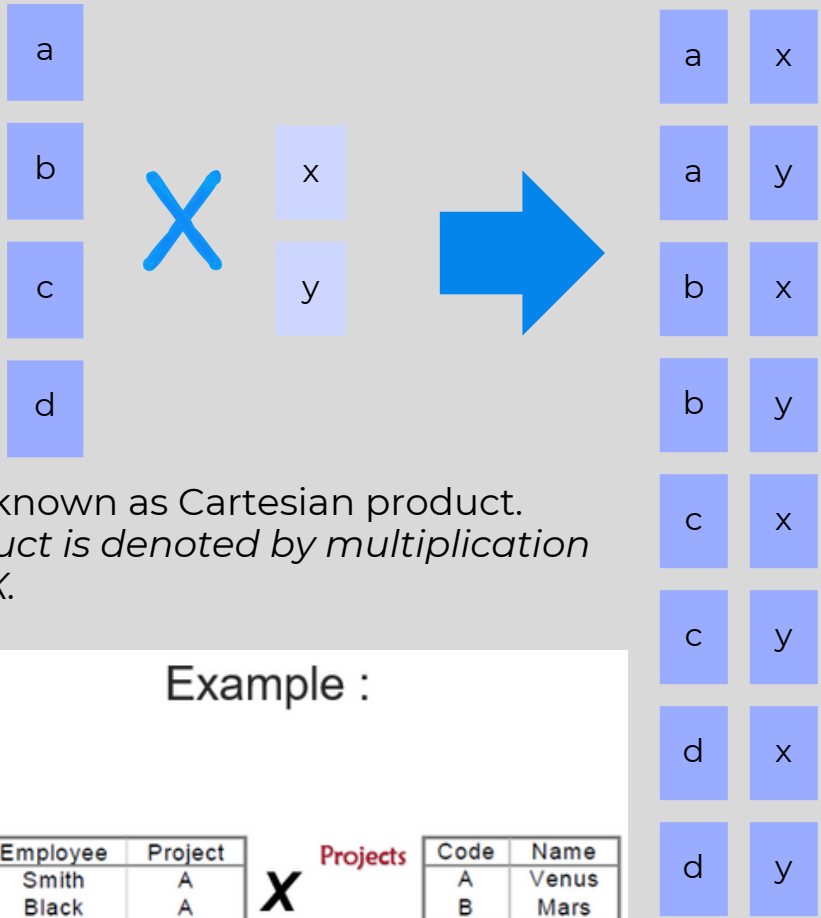


2.2.2 The fundamental operators used to retrieve information from a relational database

CROSS PRODUCT

Cross product (*x*) yields all possible pairs of rows from two tables.

- The Cross product operation combines tuples of one relation with all the tuples of the other relation.
- Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2.



- Also known as Cartesian product.
- *Product is denoted by multiplication symbol X.*

Example :

Employees

Employee	Project
Smith	A
Black	A
Black	B

X

Projects

Code	Name
A	Venus
B	Mars

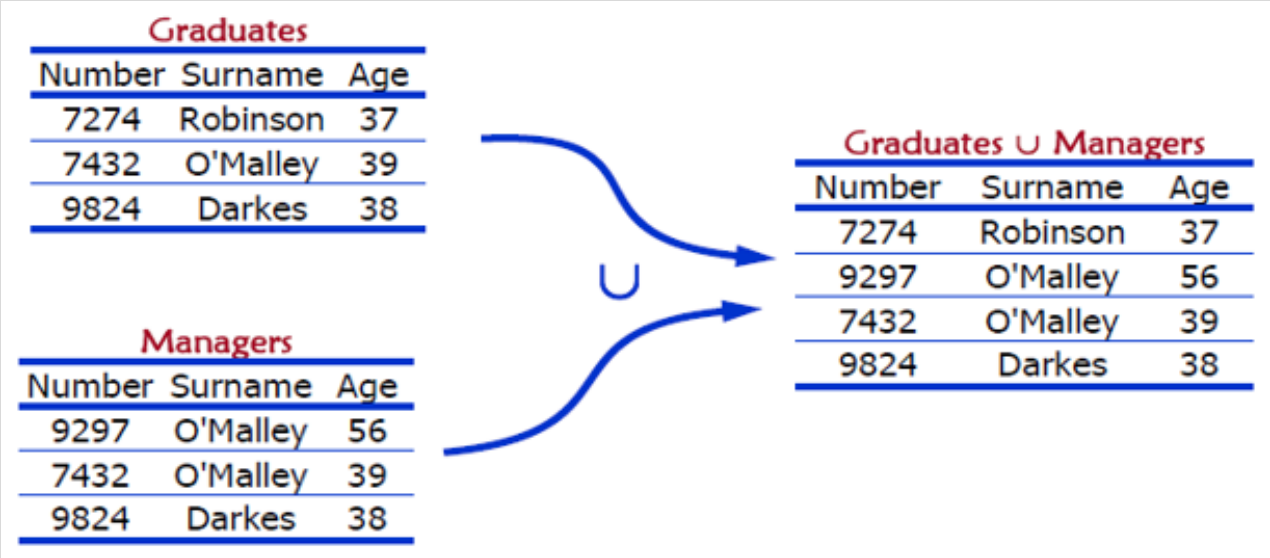
Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	A	Venus
Smith	A	B	Mars
Black	A	B	Mars
Black	B	B	Mars

2.2.3 The traditional set of operators for relational tables:

a. Union

- Relation that includes all tuples that are either in R or in S or in both R and S.
- Table must have the same number of attributes.
- Duplicate tuples are eliminated.
- Symbol: ∪
- Syntax: Л <attribute list>(R) ∪ Л <attribute list>(R)

Example:

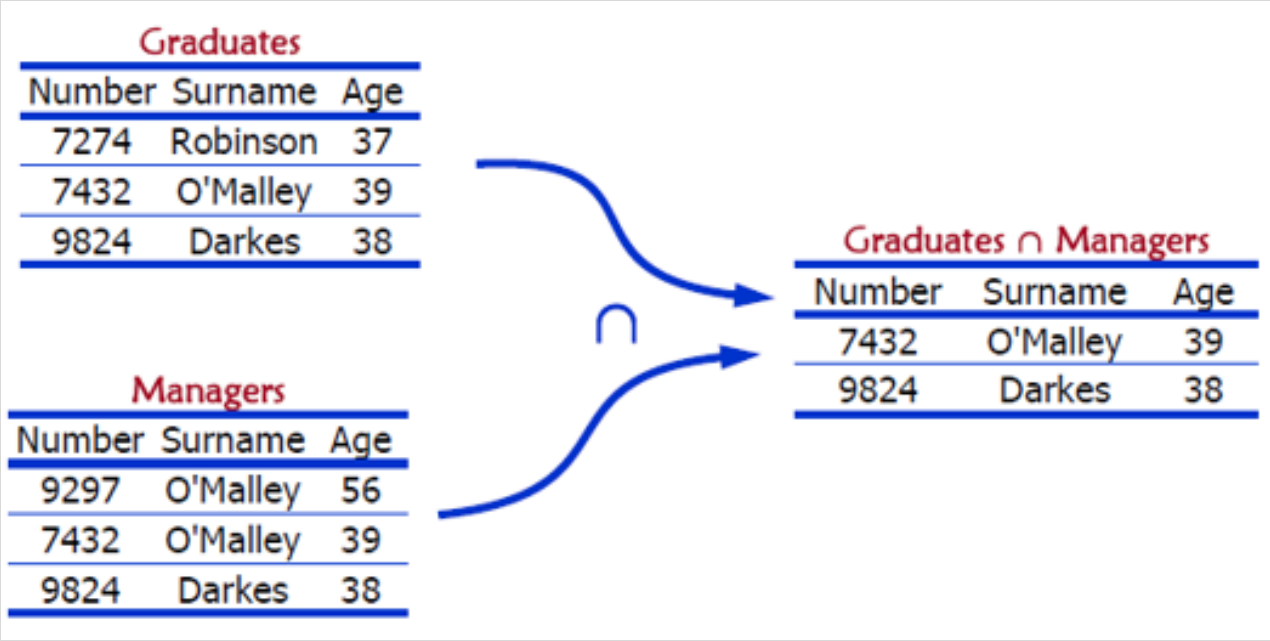


2.2.3 The traditional set of operators for relational tables:

b. Intersection

- Only those tuples that appear in both of the named relation are given as an output result.
- The two operands must be type compatible.
- Symbol: ∩
- R ∩ S
- Syntax: Л < attribute list>(R) ∩ Л < attribute list>(R)

Example:

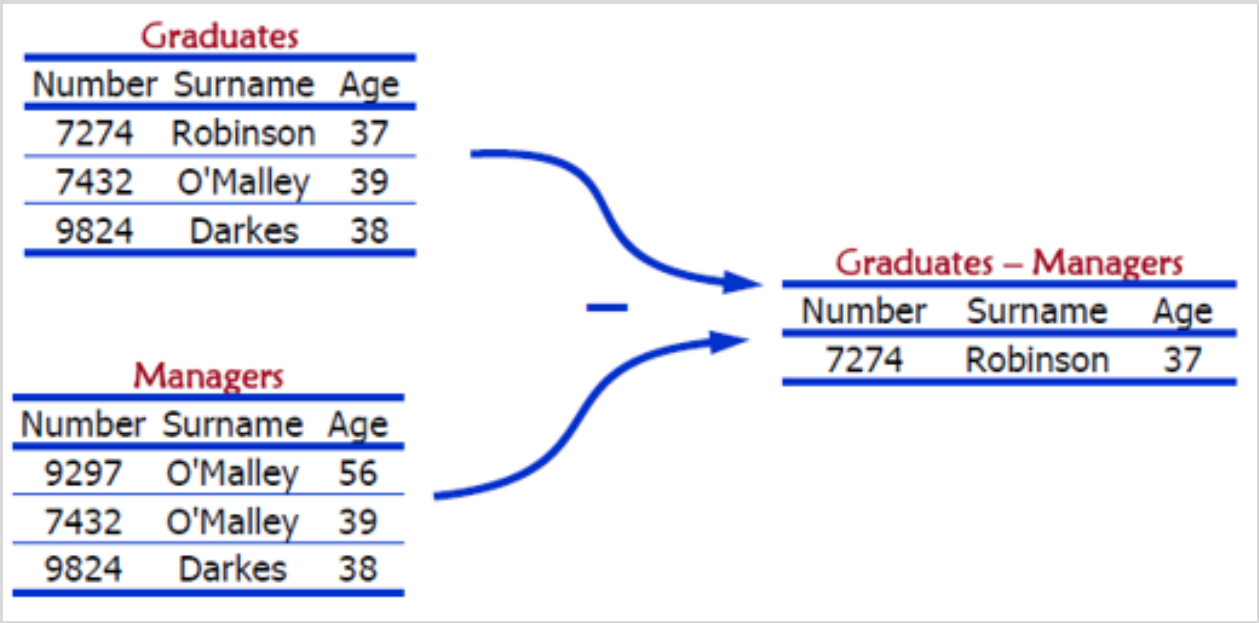


2.2.3 The traditional set of operators for relational tables:

c. Difference

- Subtracts from the first named relation those tuples that appear in the second named relation and create a new relation.
- The two operands must be type compatible.
- Symbol: -(minus)
- Syntax: Л < attribute list>(R) - Л < attribute list>(R)

Example:



2.2.4 Union compatibility

- The operand relations R 1 (A 1 , A 2 , ..., A n) and R 2 (B 1 , B 2 , ..., B n) must have the same number of attributes, and the domains of corresponding attributes must be compatible, that is, dom(A i)=dom(B i) for i=1, 2, n.
- In other words, two relations are union compatibles if they have the same degree and the same number attribute and domains

2.2.5 Example of union, intersect, and difference set operators based on tables given

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW106EU

PropertyForRent

property No	street	city	postcode	type	rooms	rent	owner No	staff No	branch No
PA14	16 Holhead	Aberden	AB75SU	House	6	650	C046	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	C087	SL41	B005
PG4	6 Lawrence St	Glasgow	G119QX	Flat	3	350	C040		B003
PG36	2 Manor Rd	Glasgow	G324QX	Flat	3	375	C093	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	C087	SG37	B003
PG16	5 Novar Dr	Glasgow	G129AX	Flat	4	450	C093	SG14	B003

2.2.5 Example of union, intersect, and difference set operators based on tables given

1. List all cities where there is in branch office or in property for rent.

Result1 <<— Л city (Branch)
Result2 <<— Л city (PropertyForRent)
Result_Union <<— Result1 ∪ Result2

city
London
Aberdeen
Glasgow
Bristol

Result

Result_Union <<— Л city (Branch) ∪ Л city (PropertyForRent)

2. List all cities where there is both a branch office and at least one property for rent.

Result1 <<— Л city (Branch)
Result2 <<— Л city (PropertyForRent)
Result_Intersection <<— Result1 ∩ Result2

city
Aberdeen
London
Glasgow

Result

Result_Intersection <<— Л city (Branch) ∩ Л city (PropertyForRent)

3. List all cities where there is a branch office but no properties for rent.

Result1 <<— Л city (Branch)
Result2 <<— Л city (PropertyForRent)
Result_Intersection <<— Result1 - Result2

city
Bristol

Result

Л city (Branch) - Л city (PropertyForRent)

Exercises:

1. Write a symbol for each of the following relational algebra operators:
   - Intersection
   - Project
   - Cross Product
2. Define the primary key and foreign key.
3. Define the degree and cardinality.
4. Differentiate between entity and referential integrity.
5. Explain THREE (3) components of relational database structural terminology.

CHAPTER 3

NORMALIZATION AND ENTITY RELATIONSHIP DIAGRAM

3.1 NORMALIZATION CONCEPT

3.1.1 Normalization concept

Normalization is the process of decomposing relations with anomalies to produce smaller, well-structured relation. Normalization is a process for assigning attributes to entities to determine whether our chosen entities, attributes and primary keys are appropriate and suitable for the system.

By doing normalization, redundancies can be reduced, and anomalies can be eliminated.

Normalization process can be divided into a few levels called Normal Forms (NF). The NF that will be covered in this subject are:

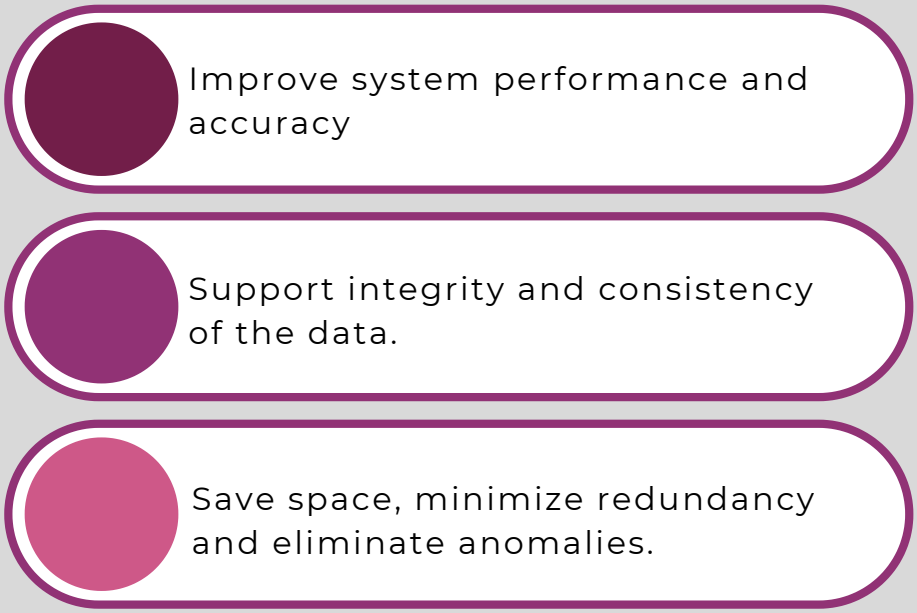
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)

3.1.2 Purpose of normalization in database models

The goal of normalization is to create a set of relational tables that are free of redundant data and that can be consistently and correctly modified.

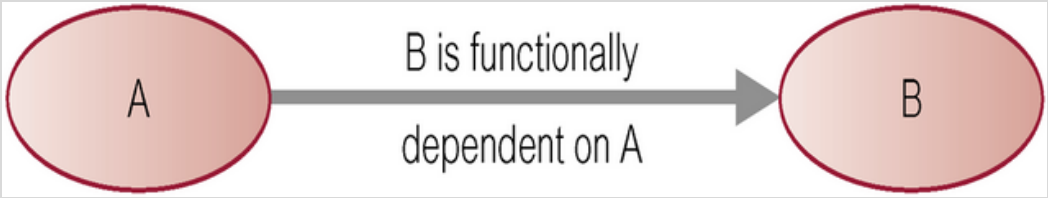
3.1 NORMALIZATION CONCEPT

3.1.3 Importance of normalization in database



3.1.4 Definition of Functional Dependencies (FD)

Describe relationships between attributes in a relation. For any relation R, attribute B is functionally dependent on attribute A if, for every valid instance of A, that value of A uniquely determines the value of B. The functional dependency of B on A is represented by an arrow, as follows: A→ B. An attribute may be functionally dependent on two (or more) attributes rather than on a single attribute.



3.1.5 Definition of Transitive Dependencies (TD)

Occurs when an attribute is functionally dependent on another non key attribute. For example, if A → B and B → C, then A → C. That is, if B depends on A, and C depends on B, then C depends on A. This is called transitive dependency.

3.1 NORMALIZATION CONCEPT

3.1.6 The various types of normal forms:

a. First Normal Form (1NF)

A relation in which the intersection of each row and column contains one and only one value. First normal form still contains redundant data. Redundancy causes problem called update anomalies.

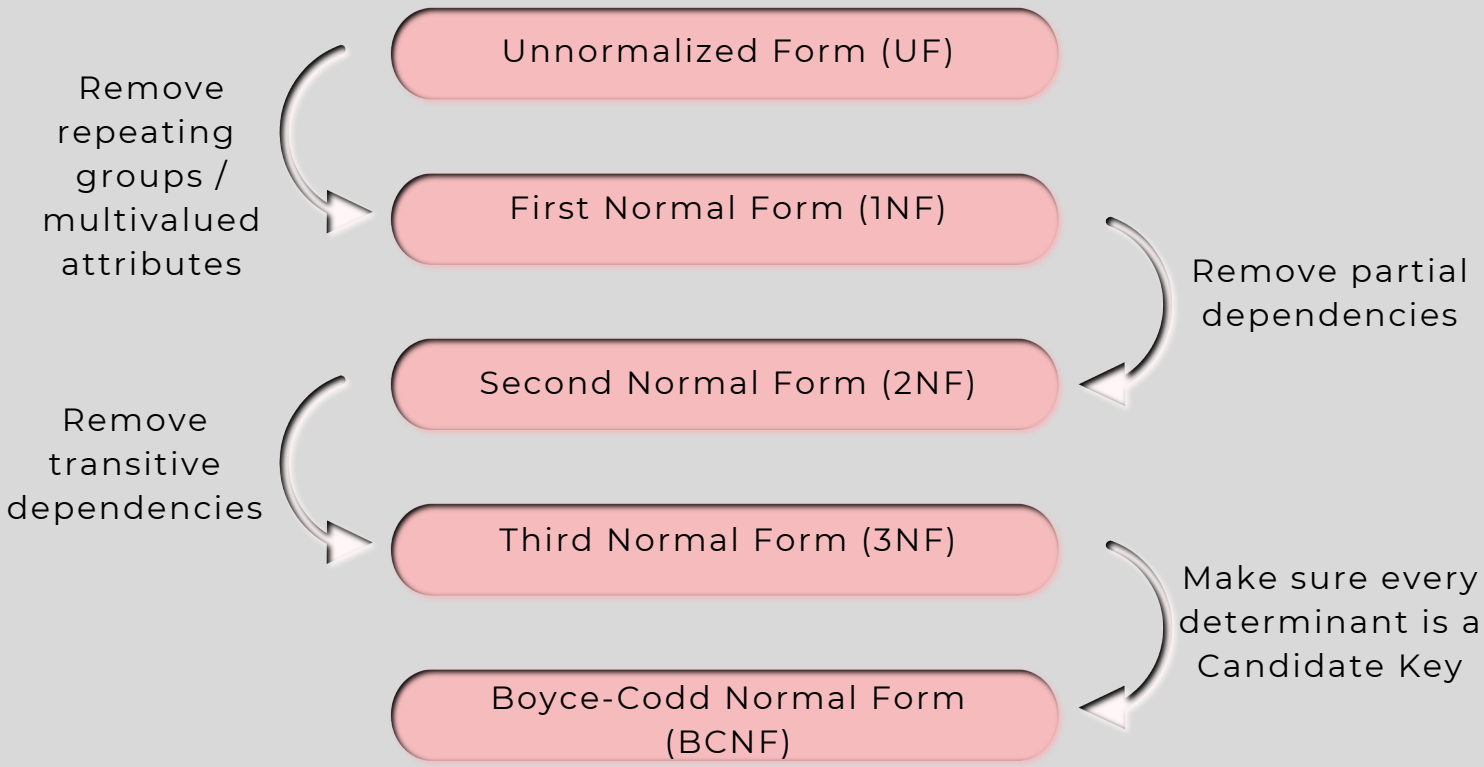
b. Second Normal Form (2NF)

A relation is in first normal form and every non candidate key attribute is fully functionally dependent on any candidate key. A table that is in first normal form and every non primary key attribute is fully dependent on the primary key. Involve in removing partial dependencies.

c. Third Normal Form (3NF)

A relation is in the first and second normal form and in which not non candidate key attribute is transitively dependent on any candidate key. All columns in a relational table are dependent only upon the primary key. Involve in removing transitive dependencies.

3.1.7 Steps in Normalization Process



3.1 NORMALIZATION CONCEPT

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Unnormalized Form

Unnormalized relation is a table that contains one or more repeating groups. Repeating groups mean that the attribute value is not atomic in each tuple.

studentID	name	major	gradUnit	courseCode	courseName	unit	gradePoint
F1050	Azura	Computer Science	118	CS001	System Introduction	2	2.5
				CS002	Computer Systems	4	3.0
M2115	Chong	Information System	125	IS100	Information Systems	4	3.0

Table above shows unnormalized table because an entry of *studentID* F1050 corresponds to more than one *courseCode*.

3.1 NORMALIZATION CONCEPT

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

First Normal Form (1NF)

A relation is said to be in 1NF if each attribute of each row contains one and only one atomic value.

In general, there are two ways to transform an unnormalized table to a table or relation in first normal form.

1. Remove repeating groups by entering appropriate data in the empty columns of tuples containing the repeating data.
2. Identify one primary key in unnormalized table. Remove the repeating groups then copy them into new table with primary key.
3. Next identify primary key for both relations.

There are 4 basic rules that a table should follow to be in 1st Normal Form :

HOW TO ACHIEVE 1ST NORMAL FORM?

1. each column should contain atomic values
2. a column should contain value that are of the same type
3. each column should have a unique name (same names leads to confusion at the time of data retrieval)
4. order in which data is saved doesn't matter

3.1 NORMALIZATION CONCEPT

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Decomposition to 1NF

The second approach will decompose unnormalized table into two new table.
As an example, the table below was decomposed into two tables which are STUDENT and COURSE with the following schemas:

STUDENT (studentID, name, major, gradUnit)
COURSE (studentID, courseCode, courseName, unit, gradePoint)

STUDENT

studentID	name	major	gradUnit
F1050	Azura	Computer Science	118
M2115	Chong	Information System	125

COURSE

studentID	courseCode	courseName	unit	gradePoint
F1050	CS001	System Introduction	2	2.5
F1050	CS002	Computer Systems	4	3.0
M2115	IS100	Information Systems	4	3.0

Output

studentID	name	major	gradUnit	courseCode	courseName	unit	gradePoint
F1050	Azura	Computer Science	118	CS001	System Introduction	2	2.5
				CS002	Computer Systems	4	3.0
M2115	Chong	Information System	125	IS100	Information Systems	4	3.0

3.1 NORMALIZATION CONCEPT

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Second Normal Form (2NF)

A relation is said to be in 2NF if it is in first normal form and every non-primary key attribute is fully dependent on the primary key.

Guideline:

A relation H in 1NF can be decomposed to few relations *H0*, *H1*, *H2*, …. *Hn* in 2NF or in higher Normal Forms by applying guidelines below:

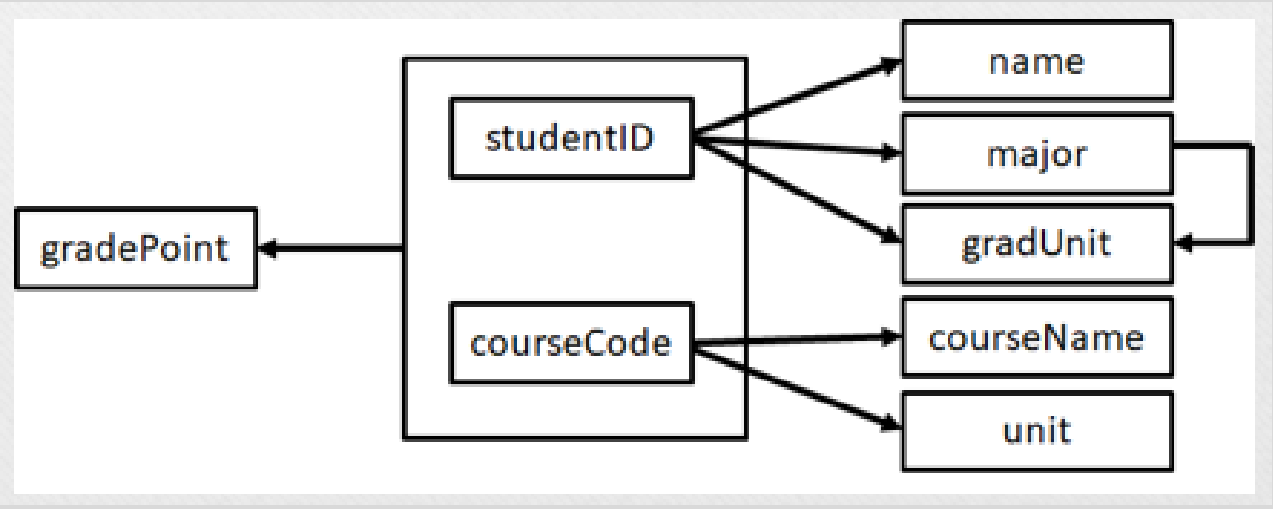
1. List all functional dependencies in H
2. Identify primary key for H and proper subset for that primary key.
3. Group all attributes which are fully dependent on primary key (***a***) into one relation (say *H0*) with primary key attributes (***p***). The output is *H0* ***(p,a).***
4. Group all attributes which are partially dependent on the primary key into one relation: with a copy of primary key attribute subset which functionally dependent on it. For example, from (***b***), grouping will produce:

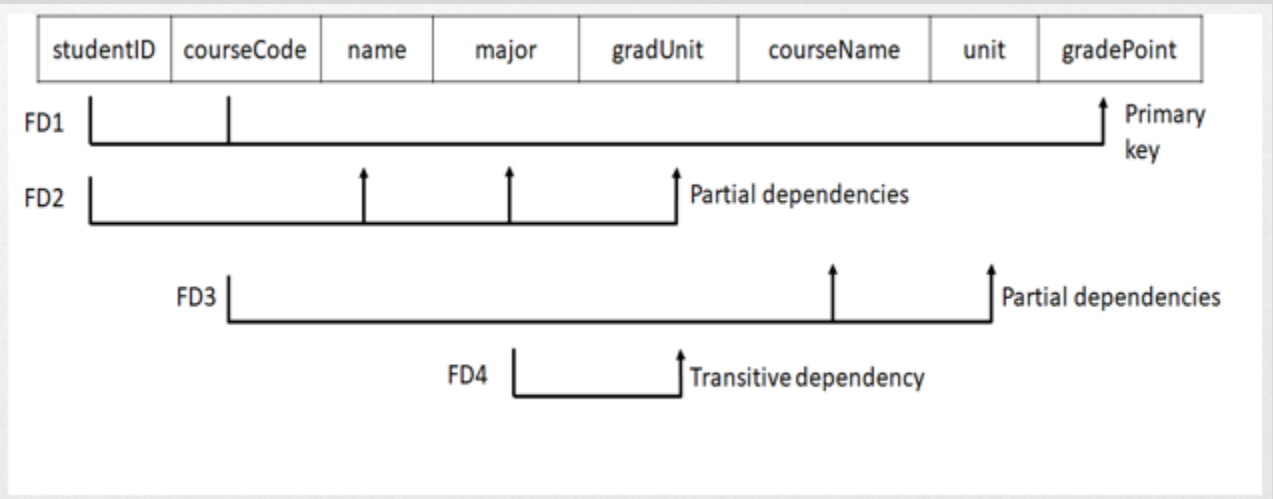
H1 (p1,a1), H2 (p2,a2),…..Hn(pn,an)

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Second Normal Form (2NF)

Those problems in STUDENTGRADE relational schema still exist because there are two themes which are not related to each other. There are student theme and course theme. When we enter student theme, columns with course theme will be null. Therefore, if we want to delete the course theme, unfortunately data on student theme will also be deleted. Functional dependency in STUDENTGRADE relation is shown in functional dependency diagram in figure below:





3.1 NORMALIZATION CONCEPT

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Decomposition to 2NF

- *gradePoint* is fully dependent on the primary key
- *(name, major, gradUnit)* are functionally dependent on *studentID* but partially dependent on the primary key
- (*courseName, unit*) are functionally dependent on *courseCode* but partially dependent on primary key.

STUDENT

studentID	name	major	gradUnit
F1050	Azura	Computer Science	118
M2115	Chong	Information System	125

COURSE

studentID	courseCode	courseName	unit	gradePoint
F1050	CS001	System Introduction	2	2.5
F1050	CS002	Computer Systems	4	3.0
M2115	IS100	Information Systems	4	3.0

GRADE

studentID	courseCode	gradePoint
F1050	CS001	2.5
F1050	CS002	3.0
M2115	IS100	3.0

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Third Normal Form (2NF)

A relation that is in 1NF and 2NF and in which no non-primary key attribute is transitively dependent on the primary key. Relation in 2NF can be transformed into 3NF by decomposing into new relations so that transitive dependency does not exist.

Guideline:

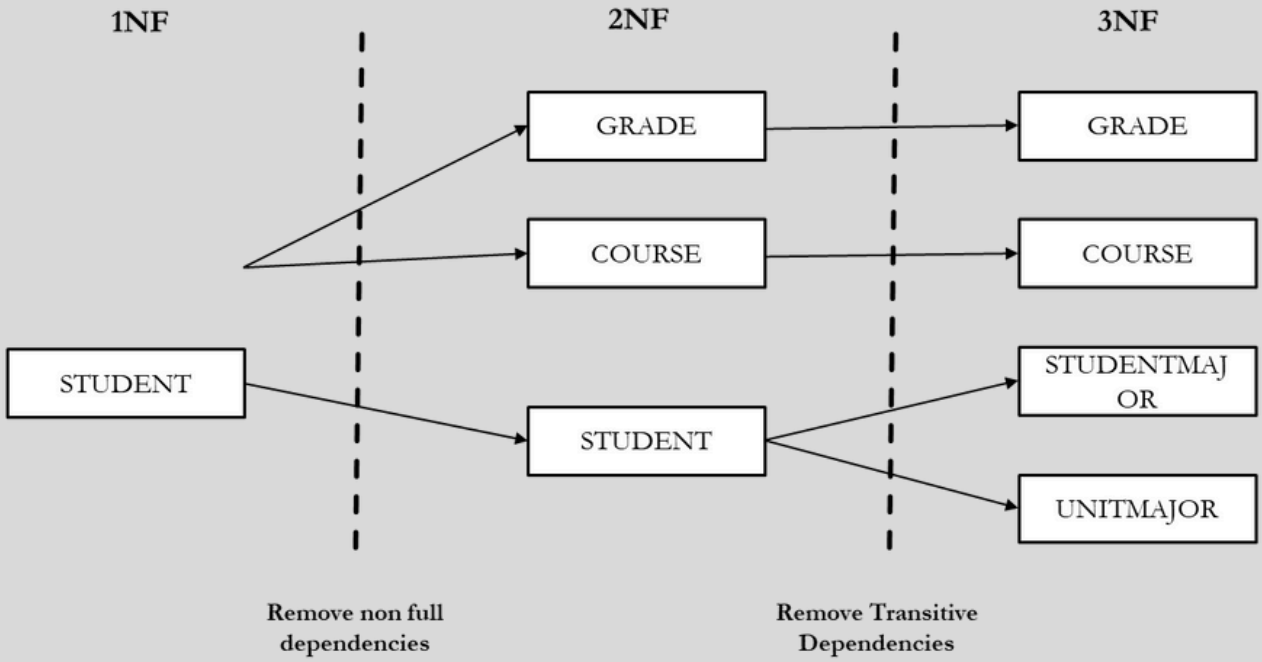
1. Identify transitive dependency and non-primary key determinant (s)
2. Group attributes which are transitively dependent with a copy of determinant(s) into a new relation. The remaining attributes should be grouped into another new relation.
3. For H relation with attributes A, B and C where A B and B C, then decompose H into H1 and H2 with the following schemas:

H1 (p1,a1), H2 (p2,a2),.....Hn(pn,an)

3.1 NORMALIZATION CONCEPT

3.1.8 The rule of First, Second and Third Normal Form to resolve a violation in the model

Decomposition of STUDENTGRADE relation into 3NF



Collection of instances for 3NF Relations from decomposition of STUDENTGRADE relation

STUDENTMAJOR

studentID	name	major
F1050	Azura	Computer Science
M2115	Chong	Information System
M4213	Karim	Software Engineering
F2020	Zarina	Information Systems
F4000	Ann	Information Systems

COURSE

courseCode	courseName	unit
CS001	System Introduction	2
CS002	Computer Systems	4
IS100	Information Systems	4
SE222	C++	4

UNITMAJOR

major	gradUnit
Computer Science	118
Information System	125
Software Engineering	130

GRADE

studentID	courseCode	gradePoint
F1050	CS001	2.5
F1050	CS002	3.0
M2115	IS100	3.0
M4213	SE222	3.0
F2020	IS100	3.5
F4000	IS100	2.0

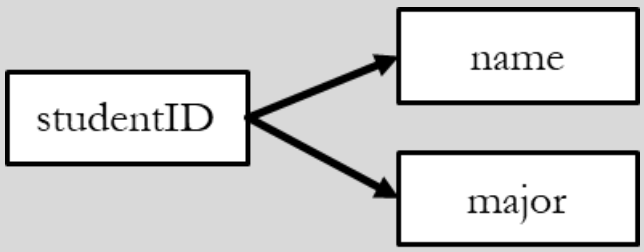
3.1.9 Boyce Codd Normal Forms (BCNF)

A relation is in BCNF of and only if every determinant is a candidate key. Advance version of normal form. Based on the concept of determinants. BCNF is considered to be part of 3NF. It is perceived to be lower than 4NF but higher than 3NF. However, you may have a table that is in 3NF but not in BCNF. Advance version of the 3NF deal with relational tables that has Multiple candidate keys, Composite candidate keys, Candidate keys that overlapped.

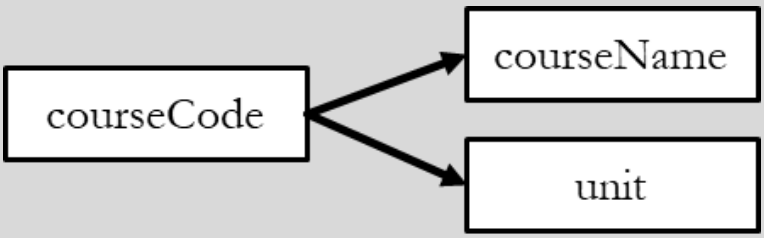
How to change a relation in 3NF to BCNF?

First, we find the determinant in that relation, then identify that all determinants are candidate keys. Determinant is defined as attribute (one or more) which is on the left side of the arrow in functional dependency statement. If given a functional dependency diagram of a BCNF relation, make sure there is no arrow except from candidate key attribute. If an arrow comes from non-candidate key attribute, that relation is not in BCNF.

Based on BCNF definition, it is clearly shows that relation STUDENTMAJOR, COURSE, UNITMAJOR and GRADE are in BCNF because each relation is in 3NF form and has only one candidate keys.

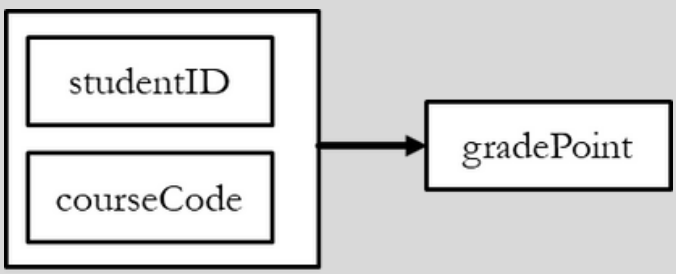


a) STUDENTMAJOR



b) COURSE

c) UNITMAJOR



d) GRADE

3.1 NORMALIZATION CONCEPT

3.1.10 The concepts of Normalization using an appropriate desktop database by creating tables and relationships

Scenario:

A company obtains parts from a number of suppliers. Each supplier is located in one city. A city can have more than one supplier located there and each city has a status code associated with it. Each supplier may provide many parts. The company creates a simple relational table to store this information that can be expressed in relational notation as:

SUPPLIER_PART(s_id, status, city, p_id, qty)

Unnormalize Form

A table that contains one or more repeating groups.

s_id	status	city	p_id	qty
S1	20	London	P1	300
			P2	200
			P3	400
			P4	200
			P5	100
			P6	100
S2	10	Paris	P1	300
			P2	400
s3	10	Paris	P2	200
S4	20	London	P2	200
			P4	300
			P5	500

SUPPLIER_PART(s_id, status, city, p_id, qty)

3.1 NORMALIZATION CONCEPT

3.1.10 The concepts of Normalization using an appropriate desktop database by creating tables and relationships

First Normal Form

Definition: A relation in which the intersection of each row and column contains one and only one value.

s_id	status	city	p_id	qty
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	500

SUPPLIER_PART(s_id, status, city, p_id, qty)

1NF
Supplier_part (s_id, status, city, p_id, qty)

FD
s_id, p_id ⟶ city, status, qty

2NF
FD
s_id ⟶ city, status
s_id, p_id ⟶ qty

Supplier_part (s_id, city, status)
Part (s_id, p_id, qty)

3.1 NORMALIZATION CONCEPT

3.1.10 The concepts of Normalization using an appropriate desktop database by creating tables and relationships

Second Normal Form

s_id	part_id	quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	500

Part (s_id, p_id, qty)

s_id	status	city
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London

Supplier (s_id, city, status)

3.1 NORMALIZATION CONCEPT

3.1.10 The concepts of Normalization using an appropriate desktop database by creating tables and relationships

2NF to 3NF

PART is already in 3NF. The non-key column, qty, is fully dependent upon the primary key (s_id, part_id)

2NF

Part
s_id, p_id ⟶ qty

Transitive Dependency! city is determined both by the primary key s_id and the non-key column status.

s_id ⟶ city, status

Transitive dependency

s_id (A) ⟶ status (B)
status (B) ⟶ city (C)
s_id (A) ⟶ city (C)

Supplier

s_id	status	city
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London

Supplier_status

s_id	status
S1	20
S2	10
S3	10
S4	20

Status_city

status	city
10	Paris
20	London

3.1 NORMALIZATION CONCEPT

3.1.10 The concepts of Normalization using an appropriate desktop database by creating tables and relationships

3NF relations

Part

s_id	part_id	quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	500

Supplier_status

s_id	status
S1	20
S2	10
S3	10
S4	20

Status_city

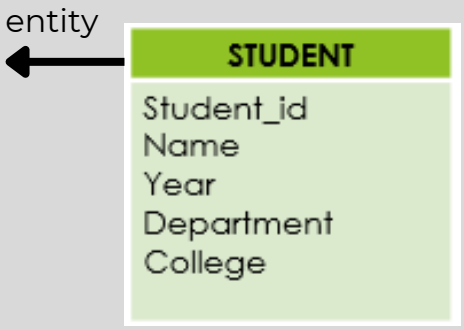
status	city
10	Paris
20	London

3NF
Part (s_id, part_id, qty)
Supplier_status (s_id, status)
Status_city (status, city)

3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.1 Entity, attribute and relationship

Entity: A collection of people, places, objects, events, concepts of interest (a table). Entity instances a member of the entity: a person, a place, an object (a row in a table).



Examples of entities:

- Person: Employee, Student, Patient
- Place: Store, Building
- Object: Machine, product, and Car
- Event: Sale, Registration, Renewal
- Concept: Account, Course

Guidelines for naming and defining entity types:

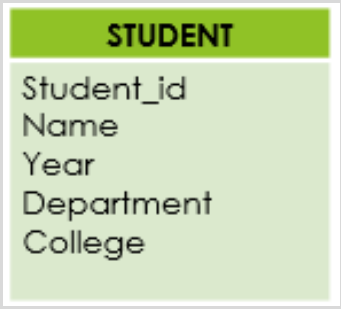
- An entity type name is a singular noun
- An entity type should be descriptive and specific
- An entity name should be concise
- Event entity types should be named for the result of the event, not the activity or process of the event.

3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.1 Entity, attribute and relationship

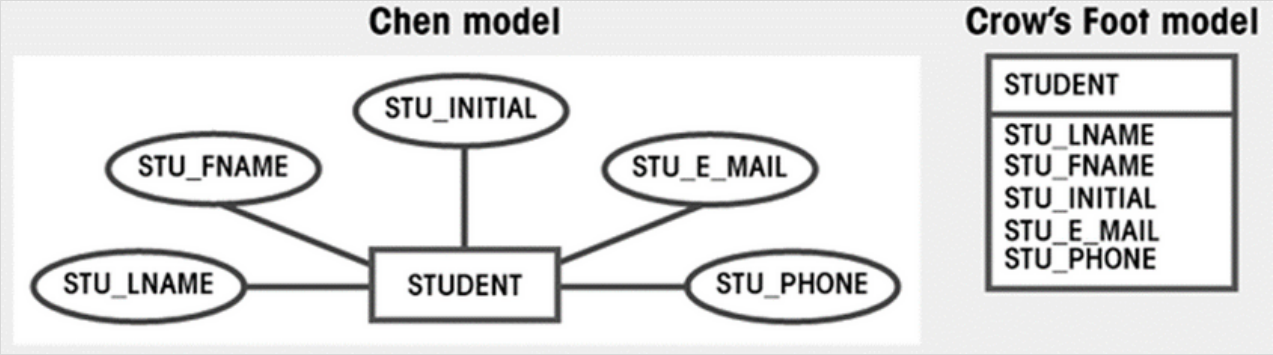
Attribute: Property or characteristic of interest of an entity (a field in a table).

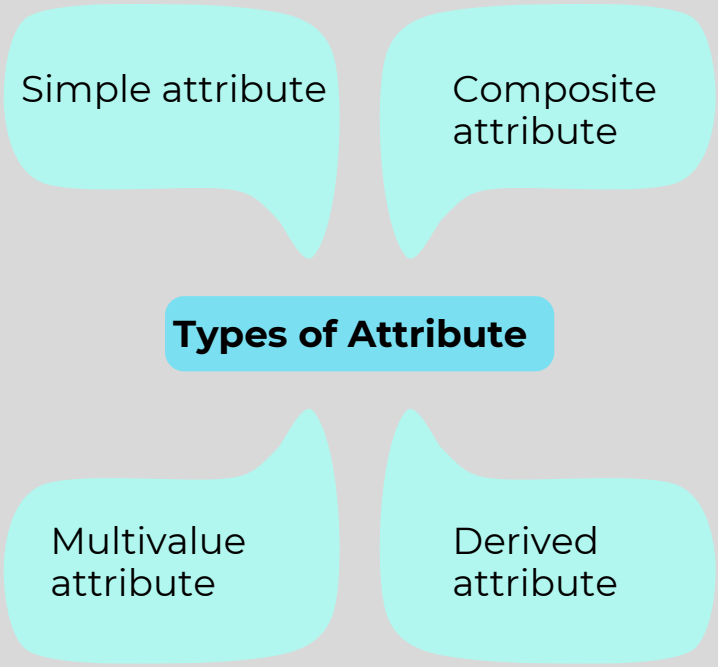
Example: Student_id, Student_Name, Home_Address



Guidelines for naming attributes:

- An attribute name is a noun
- An attribute name should be unique
- To make an attribute name unique and clear, each attribute name should follow a standard format.



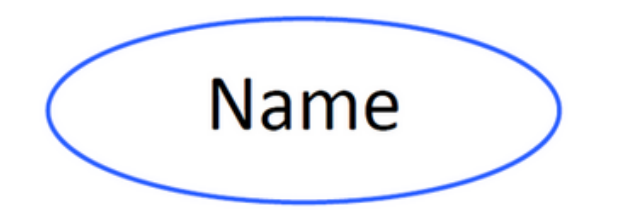


3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.1 Entity, attribute and relationship

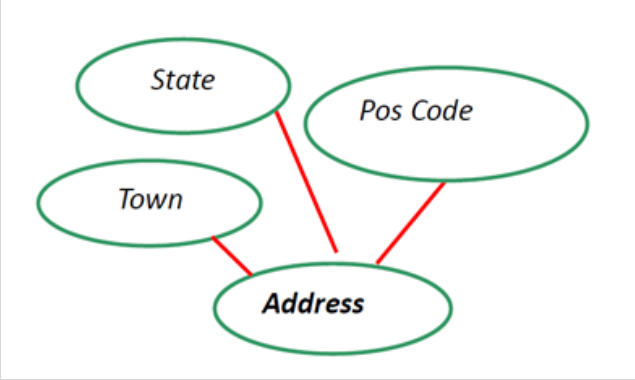
Simple Attribute
An attribute that holds a single value. Cannot be subdivided.

Example:
The majority of people have only one name



Composite Attribute
An attribute that can be further subdivided to additional attributes.

Example:
An address comprises of city, postcode and state.

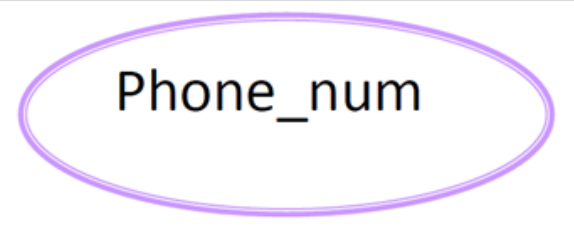


3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.1 Entity, attribute and relationship

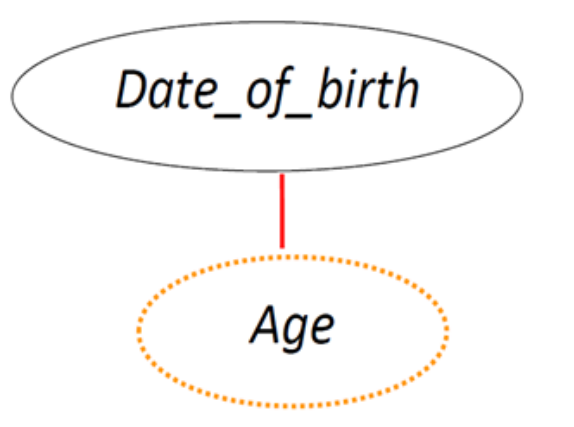
Multivalue Attribute
An attribute that has more than one value

Example:
A staff may have 2 tel_no which are home tel_no and mobile tel_no



Derived Attribute
An attribute where the values can be calculated from related attribute.

Example:
Age can be known from Date_of_Birth



3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.1 Entity, attribute and relationship

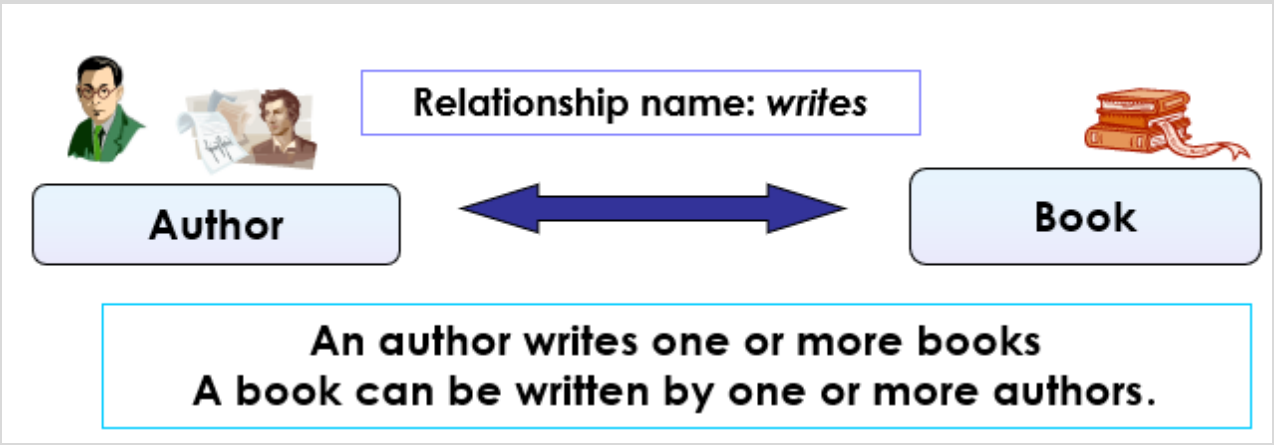
Relationship: Association between entities (corresponds to primary key foreign key equivalencies in related tables).

Verb usually used to described relationships. Examples: Students take Courses – Students and Courses are entities, and take is the relationship.

Relationships are one-to-one, one-to-many, many-to-many.

Notation/symbol – diamond/ single line

Example:



3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.1 Entity, attribute and relationship

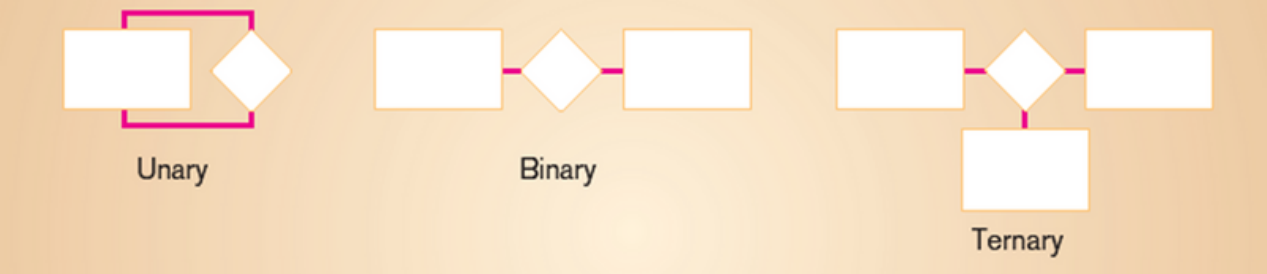
Degree of Relationships

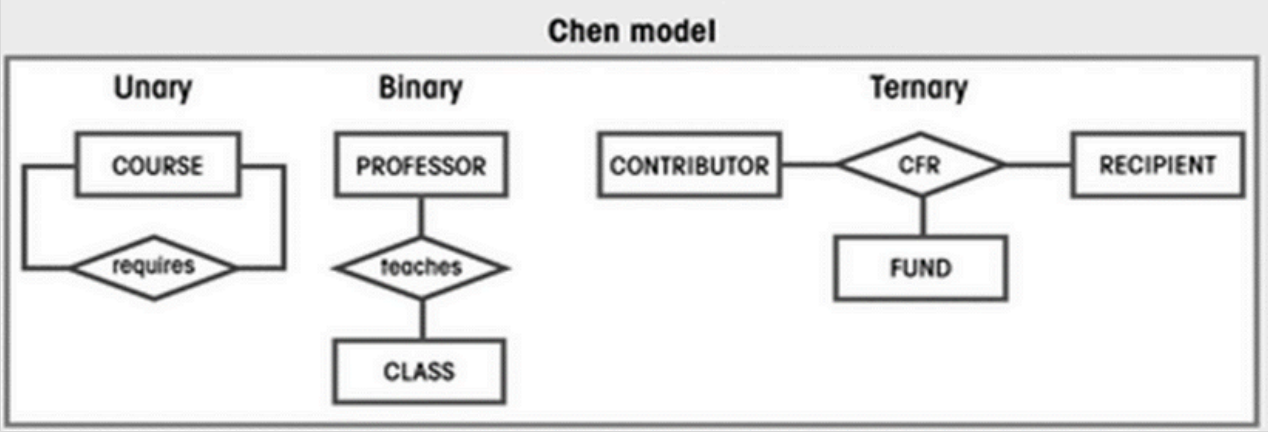
Degree: number of entity types that participate in a relationship.

Three cases:

- Unary: between two instances of one entity types.
- Binary: between the instances of two entity types
- Ternary: among the instances of three entity types

Example:



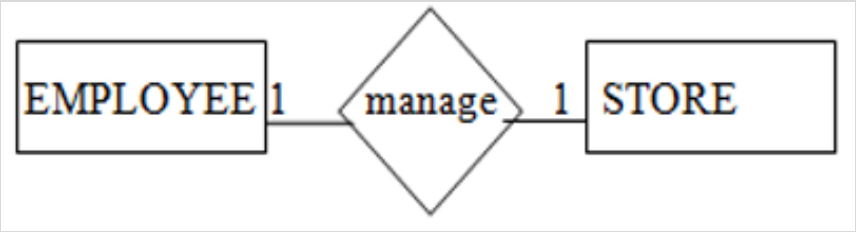


3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.2 Relationship cardinality

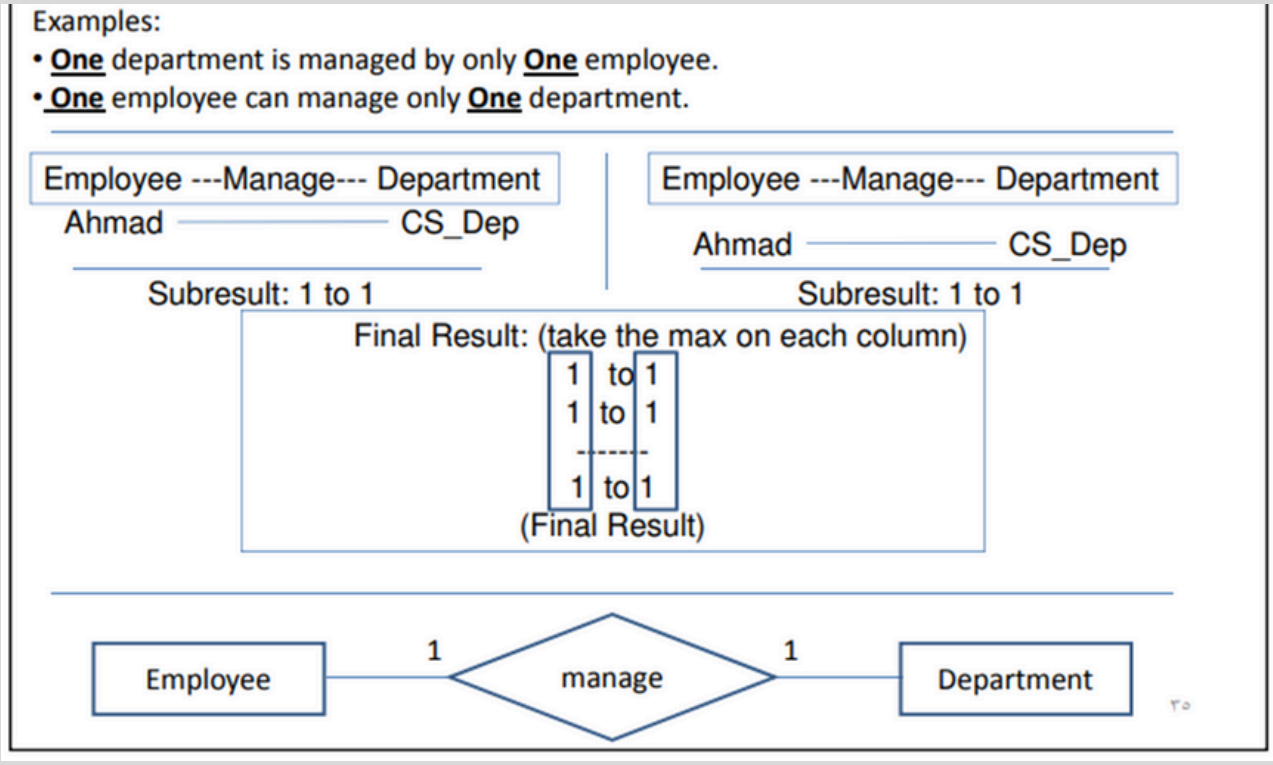
a. One to one (1:1)

Have cardinality of one and only one in both directions. Is mostly used to split an entity in two to provide information concisely and make it.



- Employer manage one store,
- each store is manage by an employer

- Each computer must contain one and only one motherboard.
- Each motherboard must be contained in one and only one computer.

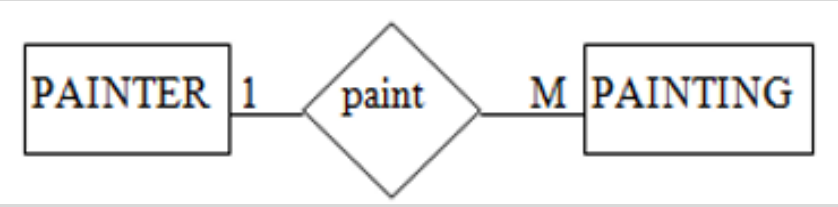


3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

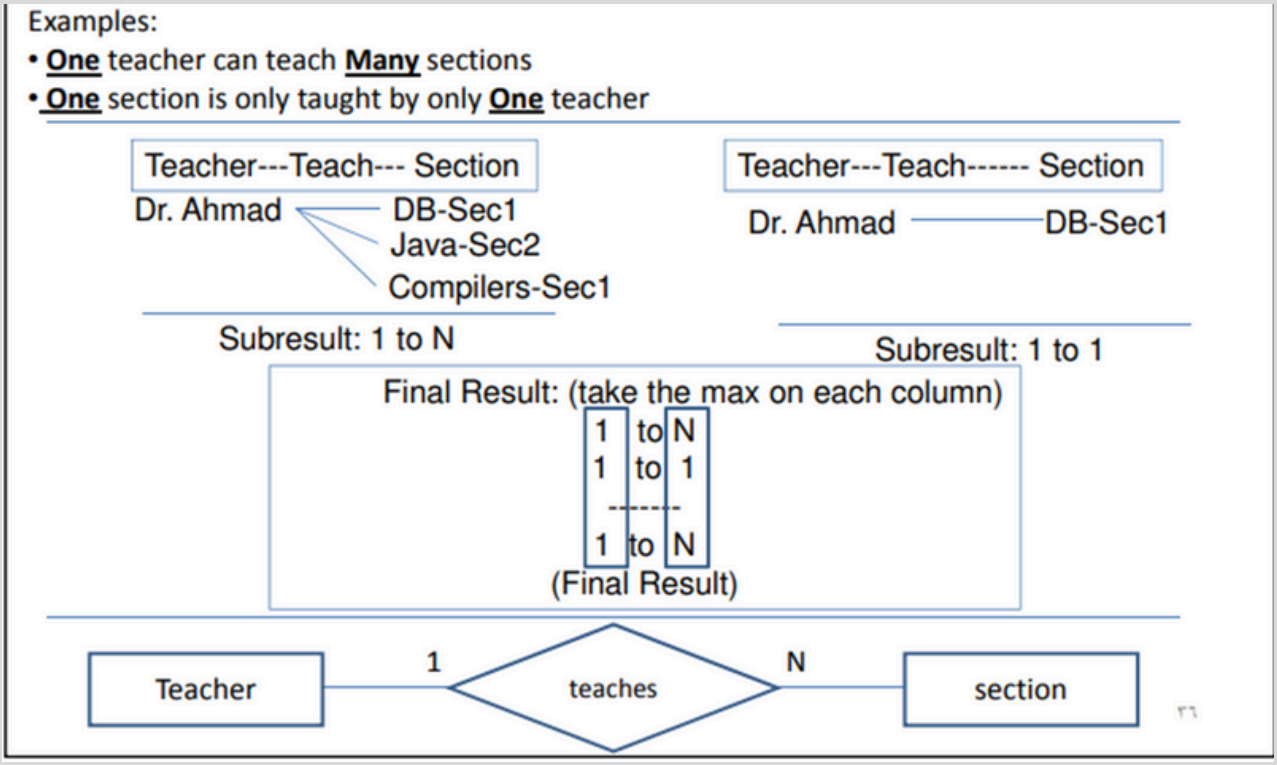
3.2.2 Relationship cardinality

b. One to many (1:M)

Have cardinality of one or more in one direction and one and only one in the other direction. A one-to-many relationship refers to the relationship between two entities X and Y in which an instance of X may be linked to many instances of Y, but an instance of Y is linked to only one instance of X.



- A painter can paint many painting
- each painting is painted by one painter.

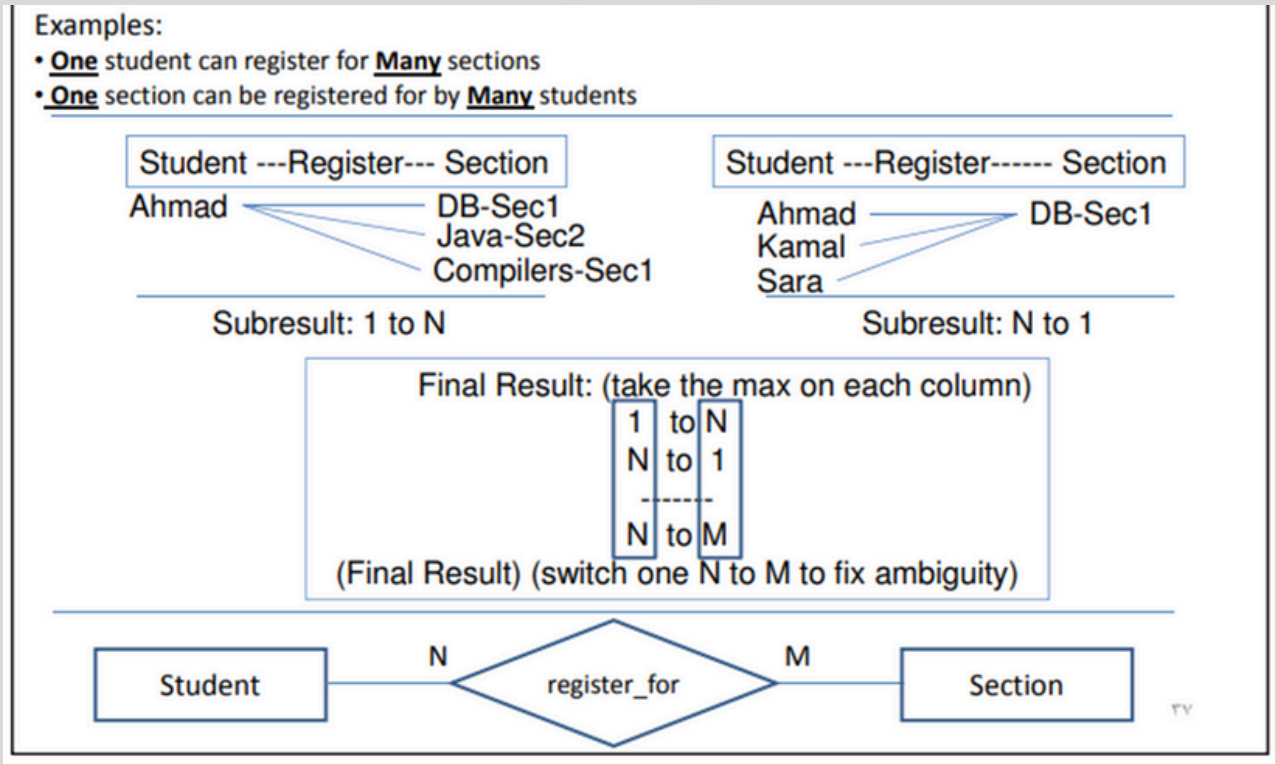


3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.2 Relationship cardinality

c. Many to many (M:N)

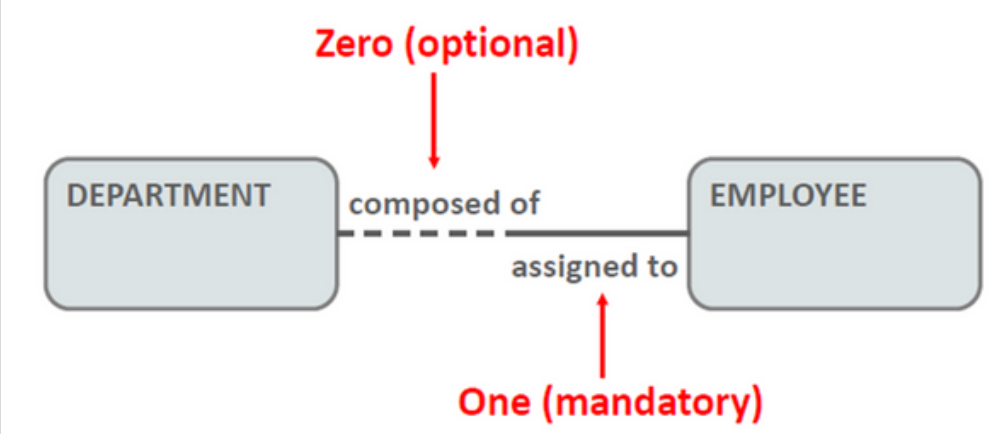
Have cardinality of one or more in both directions. A many-to-many relationship refers to the relationship between two entities X and Y in which X may be linked to many instances of Y and vice versa. Note that a many-to-many relationship is split into a pair of one-to-many relationships in a physical ERD that use a composite entity.



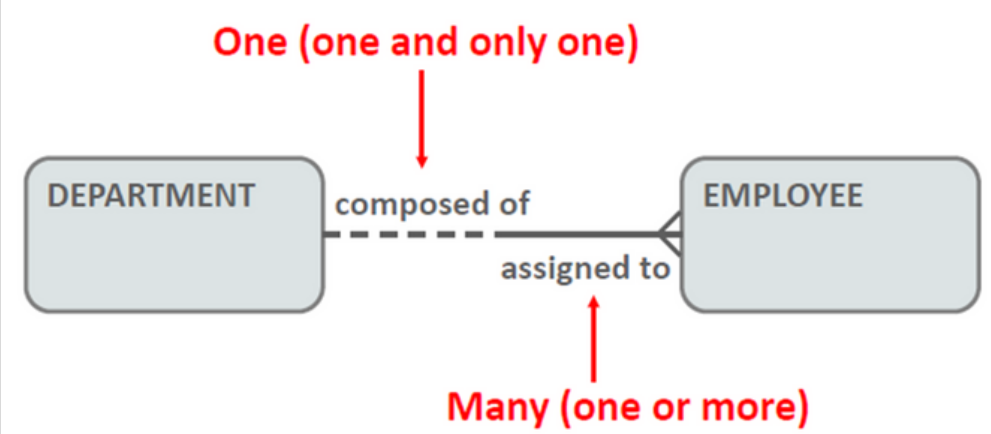
3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.3 Relate entities by applying the rules of cardinality

What is the minimum cardinality in each direction?

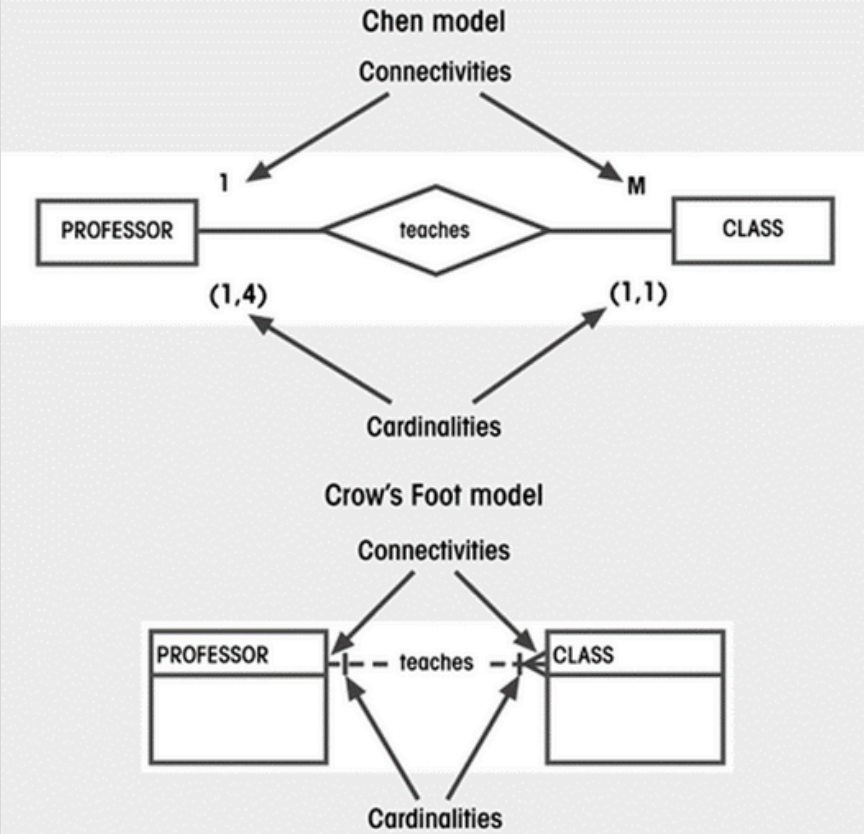


What is the maximum cardinality in each direction?



3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.3 Relate entities by applying the rules of cardinality



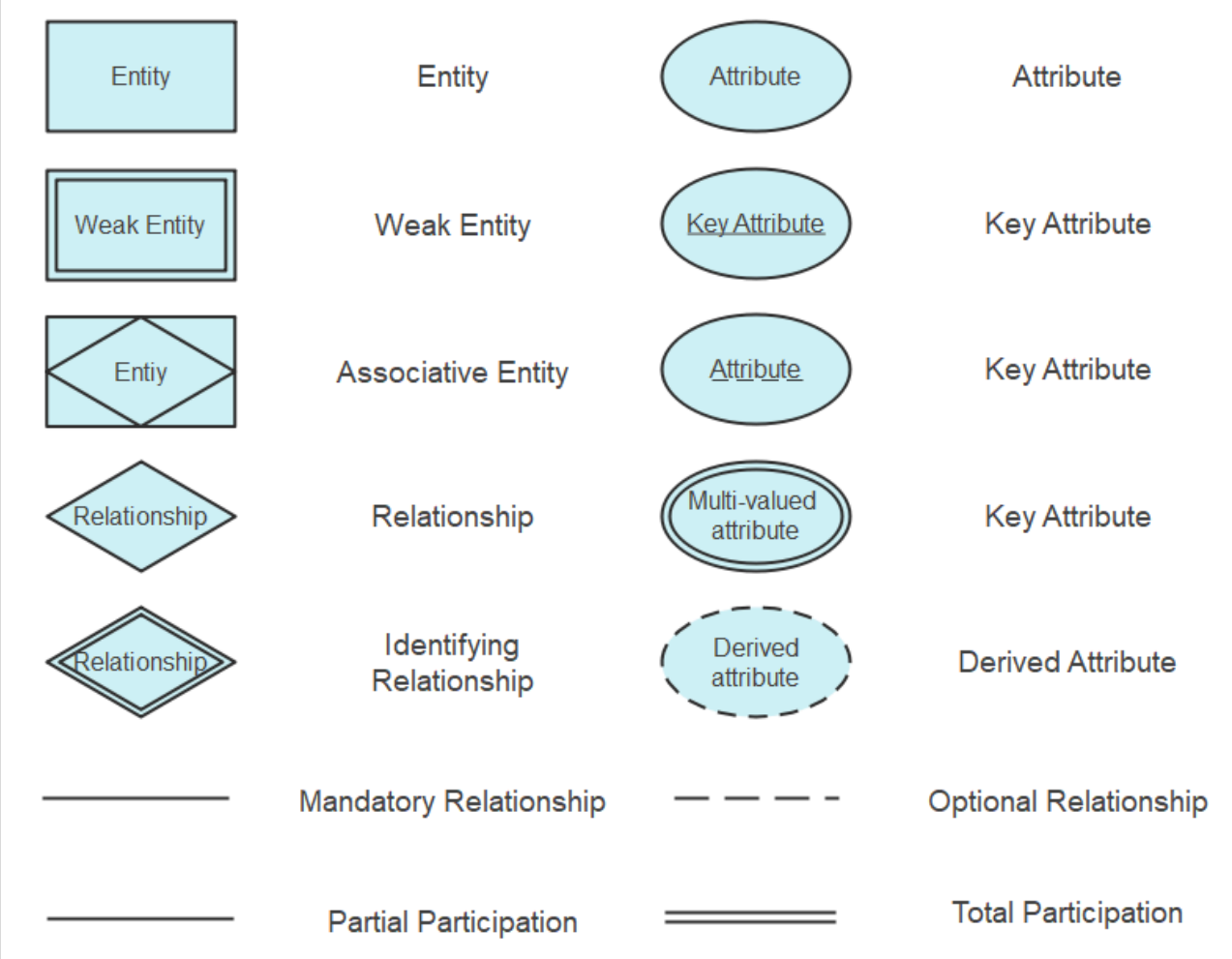
Relationship Cardinality	Minimum Instances	Maximum Instances	Graphical Notation
Zero or One	0	1	
Exactly One	1	1	
One or More	1	>1	
Zero, One or More	0	>1	
More than One	>1	>1	

Relationship Cardinality symbol

3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.4 ERD notations for:

a. Entity
b. Weak entity
c. Attributes
d. Key attribute
e. Multivalued attribute
f. Derived attribute
g. Relationships
h. Cardinality



Credit: https://www.edrawsoft.com/er-diagram-symbols.html

3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.5 Steps in creating ERD

1. **Identify the entities:** Start by identifying the key objects or concepts that need to be represented in the database. These become the entities in the ER diagram. Entities are usually nouns like 'Student', 'Course', or 'Teacher'.
2. **Identify the attributes:** For each entity, list the properties or characteristics that need to be stored in the database. These are the entity's attributes. Attributes are usually properties that describe or identify an entity. For example, a 'Student' entity may have attributes like 'Student_ID', 'First_Name', 'Last_Name', and 'Email'.
3. **Identify the relationships:** Determine how the entities interact with each other. This helps define the relationships in the ER diagram. Relationships are verbs that describe interactions between entities. For example, a 'Student' entity 'Enrolls in' a 'Course' entity.
4. **Set the cardinality:** The next step is to set the cardinality of the relationships, which specifies how many instances of an entity relate to each instance of another entity. Cardinalities can be one-to-one (1:1), one-to-many (1:M), or many-to-many (M:N).
5. **Identify and set constraints:** Constraints are rules that help to ensure the integrity and accuracy of the data. They can be of different types, such as entity integrity (no duplicate rows), referential integrity (foreign key values) must have a match in the corresponding table), and domain integrity (values in a column must be of a specific data type).
6. **Draw the ERD:** Finally, represent all these elements graphically. Entities are represented by rectangles. Attributes are represented by either ovals connected to their respective entities or as line items inside the entity rectangles. Relationships are represented by lines connecting to the related entities, and cardinalities are represented on the relationship lines connecting entities.

3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.6 Construct ERD that represent entities and attributes according to diagramming conventions based on a given scenario

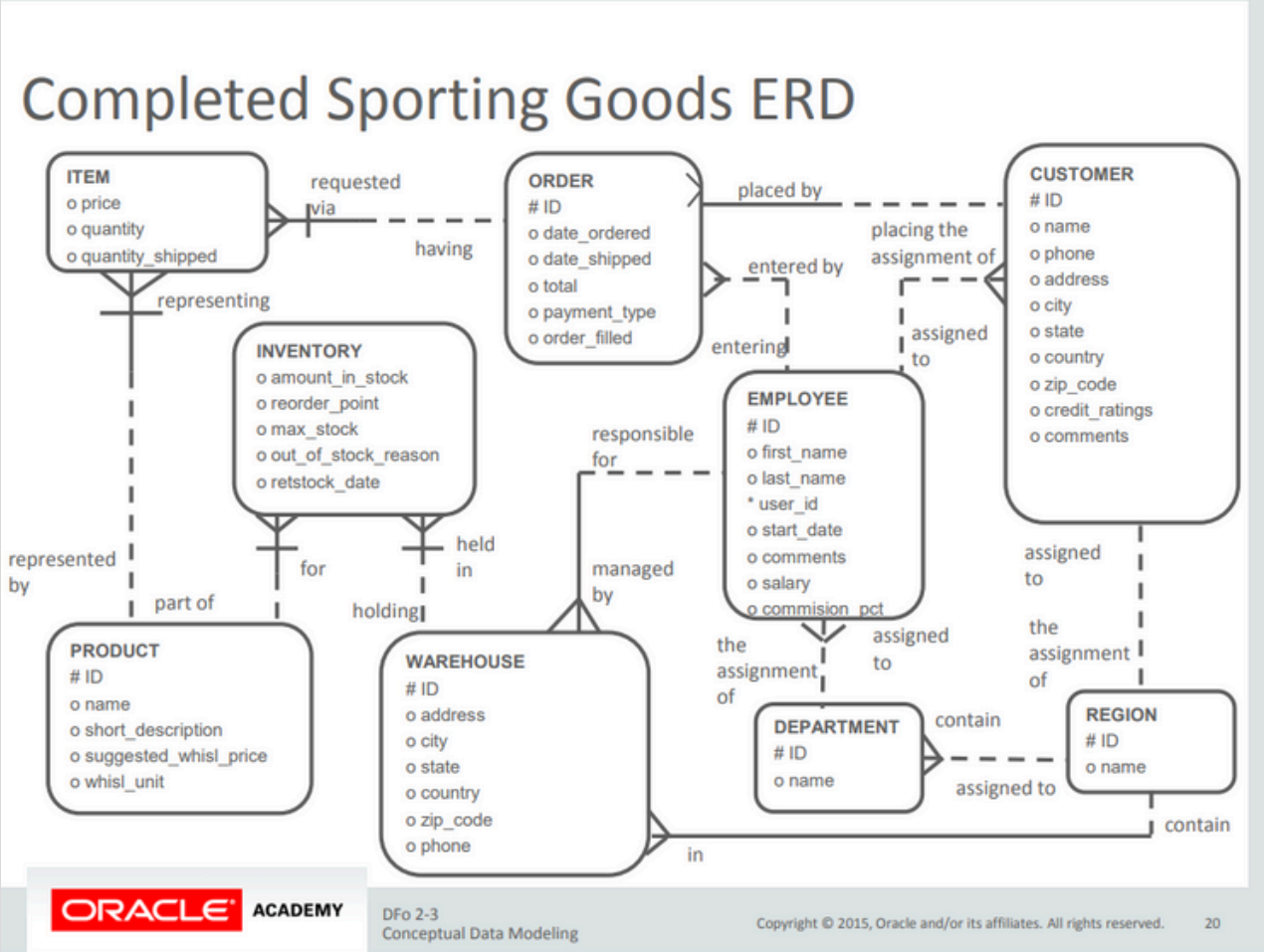
Sporting Goods Business Scenario

I'm a manager of a sporting goods wholesale company that operates worldwide to fill orders of retail sporting goods stores. The stores are our customers (some of our people prefer to call them our clients). Right now, we have fifteen customers worldwide, but we're trying to expand our customer base by about 10% each year starting this year. Our two biggest customers are in the United States: Big John's Sports Emporium in San Francisco, California, and Women's Sports in Seattle, Washington.

For each customer, we must track an ID and a name. We may also track an address (including the city, state, zip code, and country) and a phone number. We maintain warehouses in different regions to fill our customer orders. For each order, we must track an ID. We may also track the date ordered, date shipped, and payment type if the information is available.

Our order entry personnel are well versed in our product line. We hold frequent meetings with Marketing to learn about new products. The result is better customer satisfaction because we can answer customer questions. We deal with a few select customers and maintain a specialty product line. For each product, we must know the ID and name. Occasionally we must also know the description, suggested price, and unit of sale. When necessary, we also want to be able to track very long descriptions of our products and pictures of our products.

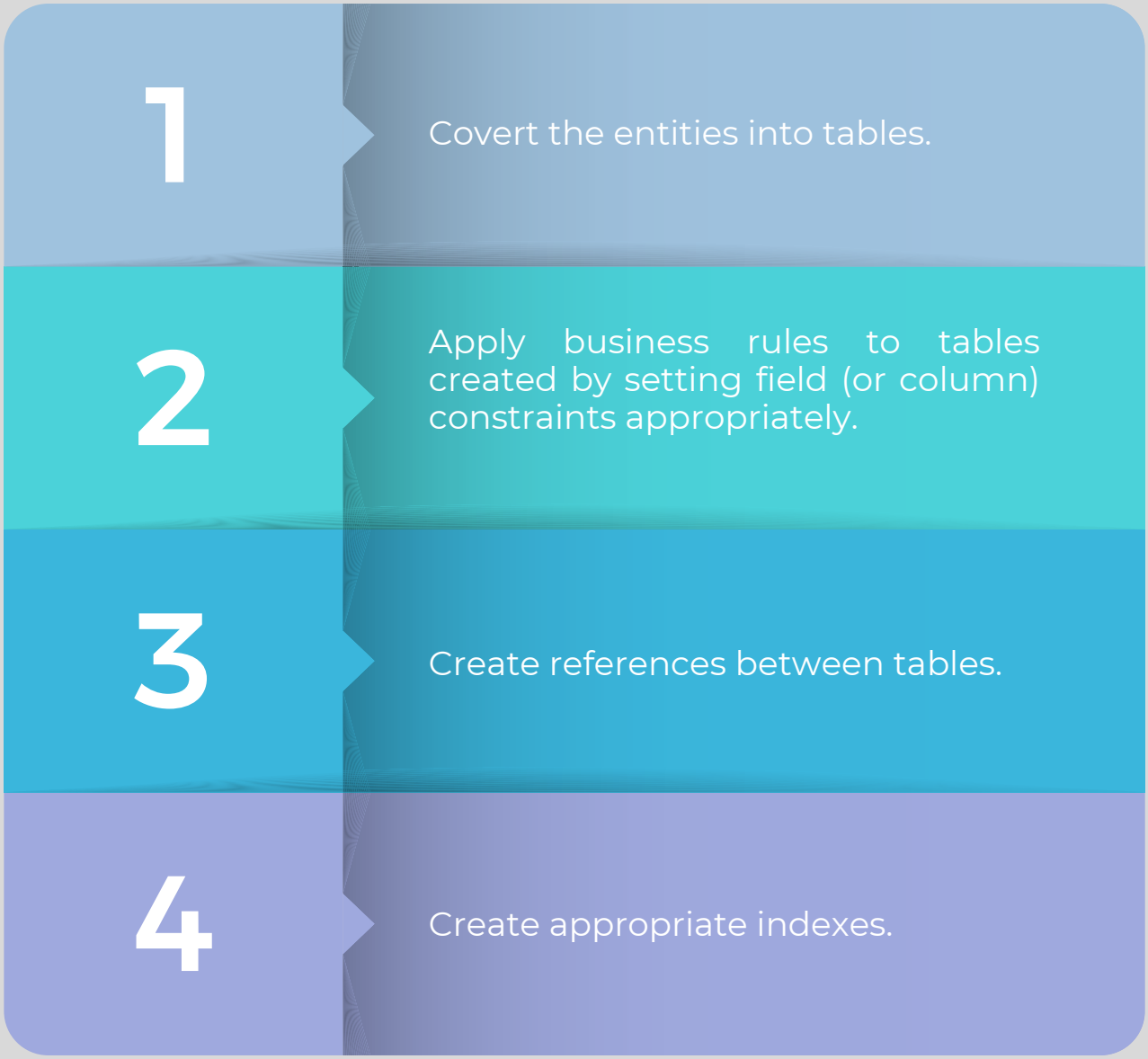
3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT



Credit: Oracle Academy

3.2 ENTITY RELATIONSHIP DIAGRAMS (ERD) IN DATABASE DEVELOPMENT

3.2.7 Transform an ERD into physical database design



Credit: http://www.dhdurso.org/articles/database-design.html

Exercises:

1. State FIVE (5) advantages of normalization process in a database.
2. Identify THREE (3) basic elements in Entity Relationship Diagram (ERD) and describe each of the elements.
3. Explain relationship cardinality:

   a. One to one (1:1)
   b. One to many (1:M)
   c. Many to many (M:N)
4. Define normalization concept.
5. Differentiate between functional and transitive dependencies.

CHAPTER 4

STRUCTURED QUERY LANGUAGE (SQL)

4.1 SQL COMMANDS TO A DATABASE

4.1.1 The use of Structured Query Language (SQL)

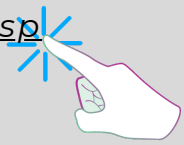
Structured query language (SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values.

4.1.2 SQL data types

For every database, data types are primarily classified into three categories.

- Numeric Datatypes (BigInt, Int,smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real)
- Date and Time Database (Date, Time, Datetime, Timestamp, Year)
- String Database (char, varchar, text, image)

You can refer more SQL Data Types at w3school :
https://www.w3schools.com/sql/sql_datatypes.asp



4.1 SQL COMMANDS TO A DATABASE

4.1.3 Types of SQL statements

a. Data Definition Language (DDL)

SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

b. Data Manipulation Language (DML)

The SQL commands that deal with the manipulation of data present in the database. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

c. Transaction Control language (TCL)

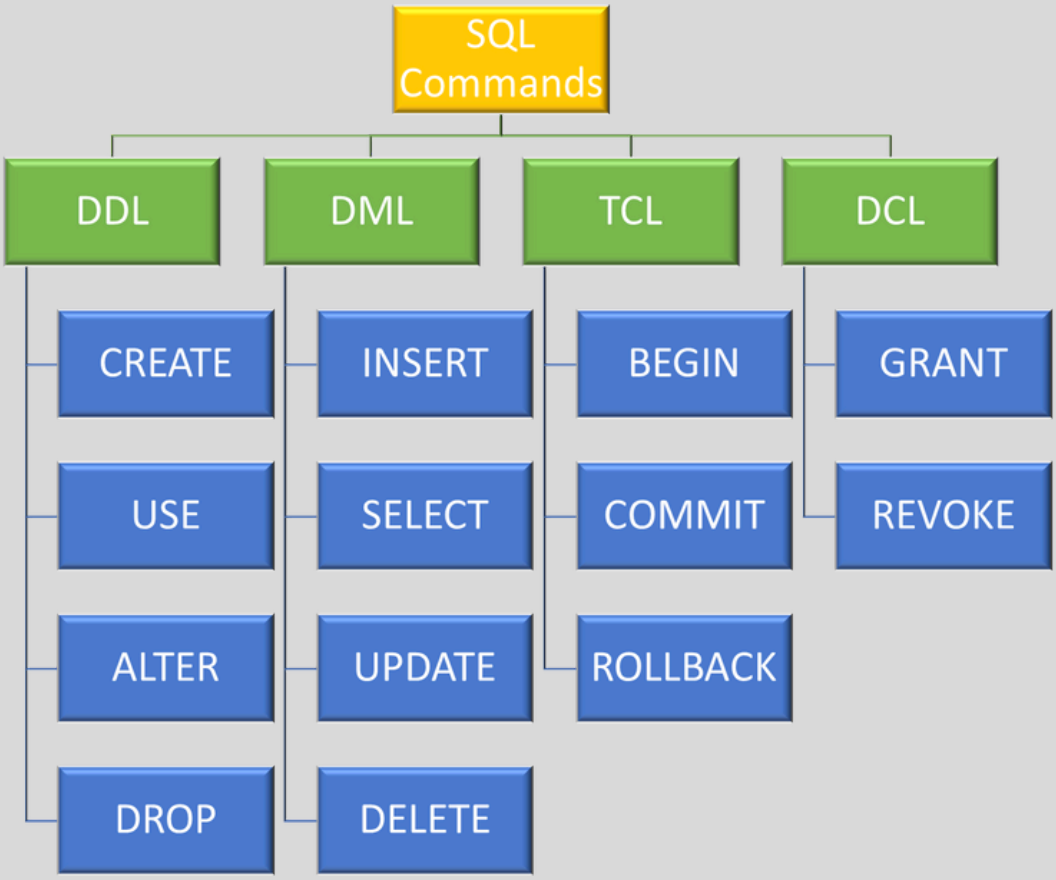
Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group are successfully completed. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure.

BEGIN Transaction – opens a transaction
COMMIT Transaction – commits a transaction
ROLLBACK Transaction – ROLLBACK a transaction in case of any error

d. Data Control Language (DCL)

Mainly deal with the rights, permissions, and other controls of the database system.

GRANT – allows users to read/write on certain database objects
REVOKE – keeps users from read/write permission on database objects


SQL Commands
DDL
DML
TCL
DCL
CREATE
USE
ALTER
DROP
INSERT
SELECT
UPDATE
DELETE
BEGIN
COMMIT
ROLLBACK
GRANT
REVOKE

4.1 SQL COMMANDS TO A DATABASE

4.1.4 DDL statements in defining the table structure

A schema is the collection of multiple database objects, which are known as schema objects. These objects have direct access by their owner schema. Below table lists the schema objects:

- Table - to store data
- View - to project data in a desired format from one or more tables
- Sequence - to generate numeric values
- Index - to improve performance of queries on the tables
- Synonym - alternative name of an object

4.1.5 Types of constraints in DDL statements: column constraints and table constraints

A constraint is a restriction on the data that can be stored in a table. MySQL allows to create constraints associated with a specific column or with a table in general. Almost all constraints can be used in both forms without modification:

Constraint	Column	Table
CHECK	Yes	Yes
NOT NULL	Yes	No*
UNIQUE	Yes	Yes
PRIMARY KEY	Yes	Yes
FOREIGN KEY	No	Yes

*: NOT NULL cannot be used as a table constraint. However, you can approximate the results by using IS NOT NULL as the statement within a CHECK table constraint.

4.1 SQL COMMANDS TO A DATABASE

4.1.5 Types of constraints in DDL statements: column constraints and table constraints

Column constraints are constraints attached to a single column. They are used to determine whether a proposed value for a column is valid or not. Column constraints are evaluated after the input is validated against basic type requirements (like making sure a value is a whole number for int columns). Column constraints are great for expressing requirements that are limited to a single field. They attach the constraint condition directly to the column involved.

Constraints are commonly used in SQL:

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY - Uniquely identifies a row/record in another table
CHECK - Ensures that all values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column when no value is specified
INDEX - Used to create and retrieve data from the database very quickly

4.1 SQL COMMANDS TO A DATABASE

4.1.6 Functions of the basic DDL commands:

a. CREATE- create a new Table, database, schema

Syntax for create database:
CREATE DATABASE database_name;

Syntax for create table:
CREATE TABLE table_name
(Column_name datatype(size));

b. USE- choose the database to work with

Syntax for use database:
USE database_name;

c. ALTER- alter existing table, column description

Syntax for alter table:
ALTER TABLE table_name
ADD Column_name datatype;

or

ALTER TABLE table_name
MODIFY (Column datatype);
[, Column datatype]... ;

or

ALTER TABLE table_name
DROP COLUMN column_name;

d. DROP- delete existing objects from database

DROP TABLE table_name;

or

DROP DATABASE database_name;

4.1 SQL COMMANDS TO A DATABASE

4.1.7 The uses of DML statements

Use to manipulate data inside the database. DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

4.1.8 The functions of the following DML commands:

a. INSERT- inserts new data into a database

Syntax:
INSERT INTO table_name [(Column [,Column....])] VALUES (Value[, Value...});

b. SELECT- extracts data from a database

Syntax for Select all columns:
SELECT * FROM table_name;

Syntax for Select specific columns:
SELECT column_name FROM table_name;

c. UPDATE- updates data in a database

Syntax:
UPDATE table
SET Column = Value [, Column = Value, ...] [WHERE condition]

d. DELETE- deletes data from a database

Syntax:
DELETE [FROM table]
[WHERE condition];

4.1 SQL COMMANDS TO A DATABASE

4.1.9 The functions SQL advanced commands:

a. DISTINCT - **eliminate duplicate rows**

Syntax:
SELECT DISTINCT column_name FROM table_name;

b. WHERE - **limiting the rows selected**

Syntax:
SELECT * | {DISTINCT column/expression [alias]...}
FROM table_name
[WHERE condition(s)};

c. AND & OR - filter records based on more than 1 condition

SELECT * FROM table_name
WHERE condition
AND condition;

SELECT * FROM table_name
WHERE condition
OR condition;

d. ORDER BY - sort the result-set in ascending or descending order

Syntax:
SELECT column_name FROM table_name ORDER BY column_name ASC|DESC;

e. WILDCARDS - search for specified pattern in a column

Syntax:
SELECT * FROM table_name WHERE table_name LIKE ' %';

4.1 SQL COMMANDS TO A DATABASE

4.1.9 The functions SQL advanced commands:

f. IN - specify multiple values in a WHERE clause

Syntax:
SELECT column_name FROM table_name WHERE column_name IN (value1, value2);

OR

SELECT column_name FROM table_name WHERE column_name IN (SELECT statement);

BETWEEN - selects values within a given range

Syntax:
SELECT column_name FROM table_name WHERE column_name BETWEEN value1 AND value2;

g. INTERSECT - return the result of 2 or more SELECT statements

Syntax:
SELECT expression FROM table_name WHERE condition INTERSECT SELECT expression FROM table_name WHERE condition;

h. GROUP BY - used with aggregate functions (COUNT, MAX, MIN, SUM, AVG)

Syntax:
SELECT column_name FROM table_name WHERE condition GROUP BY column_name ORDER BY column_name;

HAVING - used to filter based on the result of a function

Syntax:
SELECT column_name FROM table_name WHERE condition GROUP BY column_name HAVING condition ORDER BY column_name;

4.1 SQL COMMANDS TO A DATABASE

4.1.10 The functions SQL Aggregate Functions:

a. MIN - Can be used for any data type

Syntax:
SELECT MIN (column_name) FROM table_name WHERE condition;

b. MAX - Can be used for any data type

Syntax:
SELECT MAX (column_name) FROM table_name WHERE condition;

c. AVG - Can be used for numeric data

Syntax:
SELECT AVG (column_name) FROM table_name WHERE condition;

d. COUNT - Returns the number of rows in a table

Syntax:
SELECT COUNT (column_name) FROM table_name WHERE condition;

e. SUM - Can be used for numeric data

Syntax:
SELECT SUM (column_name) FROM table_name WHERE condition;

Kahoot!

Exercises:

1. Give the definition of Data Definition Language (DDL).
2. State TWO (2) SQL commands under the DDL category.
3. Give THREE (3) function commands in DML.
4. List FOUR (4) function commands in SQL aggregate functions.
5. Write SQL statement to delete existing database.

CHAPTER 5

DATABASE TRANSACTION MANAGEMENT

5.1 DATABASE TRANSACTION MANAGEMENT

5.1.1 Transaction system

A transaction is a unit of work that should be processed reliably without interference from other users and without loss of data due to failures. Examples of transactions are withdrawing cash at an ATM, making an airline reservation and registering for a course.

5.1.2 Transaction processing systems category

a. Batch transaction processing system

Through batch processing, a transaction processing system (TPS) interprets sets, or batches, of data by grouping items based on similarities. Batch processing can create a time delay because it reviews several sets of data simultaneously, requiring more computing power.

Example: A customer pays for a subscription service at the end of the month, The TPS system processes the transactions as a batch because they occur at the same time. In this case, a delay in processing transactions is acceptable because the system only interprets batches once per month.

b. Online Transaction Processing System (OLTP)

Online transaction processing is a type of data system that aims to execute a series of transactions carried out in online banking, such as purchases or order entry.

Example: Month end tax calculation, data transformation, data analysis, data transformation.

c. Real time transaction processing system

Real-time processing is a method to process transactions as they appear. This helps prevent delays in processing and can provide a more accurate result.

Example: An e-commerce website might use a TPS to process credit card transactions in real-time to ensure payment before the company starts its fulfillment process. Processing transactions in real-time also helps the company identify and address errors quickly, as well as increase its overall response times.

5.1 DATABASE TRANSACTION MANAGEMENT

5.1.3 Properties of database transaction

Atomicity

Either all operations of a transaction are reflected in the database or none of them (all or nothing).

Consistency

If the database was a consistent state before the transaction started, it will be in a consistent state after the transaction has been executed.

Isolation

If transactions are executed in parallel, the effects of an ongoing transaction must not be visible to other transactions.

Durability

After a transaction finished successfully, its changes are persistent and will not be lost such as on system failure.

5.1.4 Perform transaction of a given database using SQL statements

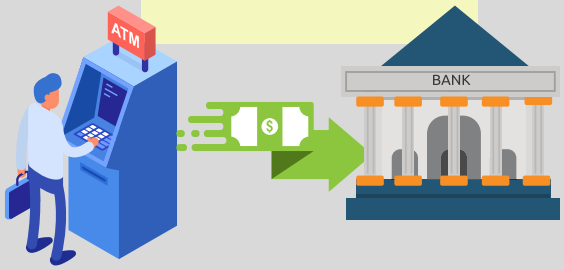
```
CREATE PROCEDURE transfer (accountA CHAR(10), accountB CHAR(10), amount DECIMAL(12,2))
BEGIN
    DECLARE currentBalance DECIMAL (12,2);
    SELECT balance INTO currentBalance
    FROM account
    WHERE account.accountNumber = accountA;

    IF (currentBalance > amount) THEN
        UPDATE account
        SET account.balance = balance - amount
        WHERE account.accountNumber = accountA;

        UPDATE account
        SET account.balance = account.balance + amount
        WHERE account.accountNumber = accountB;
    ENDIF
END
```

Example:

Faris wants to transfer RM 700 from bank account (314-229) to landlord's account (889-752)



5.1.5 Use START, TRANSACTION and COMMIT statements

```
start transaction
read (A)
A = A - 700
write (A)
read (B)
B = B + 700
write (B)
commit
```

5.1.6 Purpose of concurrency control

The purpose of concurrency control is to prevent two different users (or two different connections by the same user) from trying to update the same data at the same time. Concurrency control can also prevent one user from seeing out of date data while another user is updating the same data.

5.1.7 Problems of concurrency control

a. Lost update

Successfully completed update is overridden by another user. Lost updates occur when two or more transactions select the same row and then update the row based on the value originally selected. Each transaction is unaware of other transactions. The last updates overwrite updates made by the other transactions which results in lost data.

b. Uncommitted data

Occurs when one transaction can see intermediate results of another transaction before it has committed. Uncommitted dependency occurs when a second transaction selects a row that is being updated by another transaction. The second transaction is reading data that has not been committed yet and may be changed by the transaction updating the row.

c. Inconsistent retrieval

Occurs when transaction reads several values but second transaction updates some of them during execution of first.

5.1.8 Concurrency control with locking methods

A lock guarantees exclusive use of a data item to a current transaction. For example, transaction B does not have access to a data item that is currently being used by transaction A. A transaction acquires a lock prior to data access; the lock is released (unlocked) when the transaction is completed so that another transaction can lock the data item for its exclusive use. Most multiuser DBMSs automatically initiate and enforce locking procedures. All lock information is managed by a lock manager.

a. Lock granularity

Indicates the level of lock use. Locking can take place at the following levels: database, table, page, row or even field.

b. Lock types

Regardless of the level of locking, the DBMS may use different lock types:

1. Binary Locks: Have only two states which are locked (1) or unlocked (0).

2.Shared/Exclusive Locks: An exclusive lock exists when access is reserved specifically for the transaction that locked the object. The exclusive lock must be used when the potential for conflict exists. A shared lock exists when concurrent transactions are granted read access on the basis of common lock. A shared lock produces no conflict as long as all the concurrent transactions are read only.

c. Two phase locking to ensure serialize ability

A method of concurrency control in DBMS that ensures serializability by applying a lock to the transaction data which blocks other transactions to access the same data simultaneously. Two Phase Locking protocol helps to eliminate the concurrency problem in DBMS.

d. Deadlocks

A deadlock occurs when two transactions wait indefinitely for each other to unlock data. The three basic techniques to control deadlocks are:

- **Deadlock prevention:** A transaction requesting a new lock is aborted when there is the possibility that a deadlock can occur. if the transaction is aborted, all changes made by this transaction are rolled back and all locks obtained by the transaction are released. The transaction is then rescheduled for execution.
- **Deadlock detection:** The DBMS periodically tests the database for deadlocks. if a deadlock is found one of the transactions is aborted (rolled back and restarted) and the other transaction are continuing.
- **Deadlock avoidance:** The transaction must obtain all of the locks it needs before it can be executed. This technique avoids the rollback of conflicting transactions by requiring that locks be obtained in succession.

5.1.9 Database recovery management

a. Database recovery

To restore a database to a consistent state after a failure or error has occurred. The main goal of recovery techniques is to ensure data integrity and consistency and prevent data loss.

b. Transaction recovery

To isolate the changes made to recoverable resources by a transaction. This means that when the transaction makes a change, that change is seen only by that transaction. After the transaction reaches the commit point, all changes that the transaction made are visible to other transactions. If an application decides to back out a transaction rather than complete it, then the resource manager backs out all changes that the transaction made. This transactional recovery is a major part of transaction processing.

c. Database back-up

The process of backing up the operational state, architecture and stored data of database software. It enables the creation of a duplicate instance or copy of a database in case the primary database crashes, is corrupted or is lost.

Exercises:

1. Describe TWO (2) properties of database transaction.
2. Explain the purpose of concurrency control.
3. State TWO (2) problems of concurrency control.
4. Describe deadlocks in database transaction management.
5. Differentiate between database recovery and transaction recovery.

REFERENCES

1. Khang. A. (2020). Relational Database - Design Rules and SQL Coding Conventions. edXOps Foundation. Brussel.
(ISBN: 979-8612037262)

2. Python. M. (2020). SQL for beginners: The simplified beginner's guide, to learn and understand SQL language computer programming, data analytics, database design and server. Including basic project and exercise. Kindle Edition. Seattle.
(ASIN: B084CSQK5X)

3. Coronel. C. & Morris, S. (2018). Database Systems: Design, Implementation, & Management 13th Edition. Cengage Learning, Inc.
(ISBN: 9781337627900)

4. Hogan. R. (2018). A Practical Guide to Database Design. Taylor & Francis Ltd.
(ISBN: 9781138578067)

5. Pannerselvam. R. (2018). Database Management Systems. Third Edition. PHI Learning Private Limited.
(ISBN: 9387472108)

6. Parmar. N., Aharwal. P. A., & Shukla, P. K. (2018). Database Management Systems: A Practical Approach. LAP Lambert Academic Publishing.
(ISBN: 6139925169)

POLITEKNIK METrO TASEK GELUGOR
No. 25, Jalan Komersial 2
Pusat Komersial Tasek Gelugor
13300 Tasek Gelugor
Pulau Pinang, Malaysia
Tel: 04-5732789
Fax: 04-5732087
Website: www.pmtg.edu.my

e ISBN 978-967-2744-19-1

9 789672 744191