

Build COMPETENCY  
across your TEAM



**Microsoft Partner**  
Gold Cloud Platform  
Silver Learning

# RDBMS Fundamentals





# Module 1

## Architecture of RDBMS

# What is DBMS :

- A **DBMS** is a software that allows creation, definition and manipulation of database, allowing users to store, process and analyze data easily. DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.

# Database Management Systems (DBMSs)

**DBMS:** a collection of general-purpose, application-independent programs providing services :

- Data Model all data stored in a well defined way
- Access control only authorized people get to see/modify it
- Concurrency control multiple concurrent applications access data
- Database recovery nothing gets accidentally lost
- Database maintenance

## Database Applications : Databases touch all aspects of our lives

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

### Some popular DBMS :

- Oracle
  - SQL Server
  - MySql
  - IBM DB2
  - PostgreSQL
  - Amazon SimpleDB (cloud based) etc.
- New application areas:
    - the Internet
    - On-Line Analytic Processing (OLAP)
    - data warehousing
    - embedded systems
    - multimedia
    - XML
    - data streams

# Data Model

- A data model is a collection of concepts and rules for the description of the structure of the database. Structure of the database means the data types, the constraints and the relationships for the description or storage of data respectively. .

# Flat File Model

- Flat File Model
  - A Flat File is a plain text or binary file which contains one record per line.
  - Within each such record, the individual fields are separated by delimiters such as commas or special characters or the fields themselves are designed to be of a fixed length.
  - 3. There are no structural relationships between the records.
- Various problems associated with this data model.
  - Data redundancy and inconsistency
  - Integrity Problem
  - Concurrency issues
  - Data integrity
  - Security, authorization and selective access
- Today, the Flat file Model is used only for the simplest and most basic of database solutions

Name	Group #	Occupation
Watts	1000	Model
Shelton	1000	Chef
Weber	1000	Chef
Tubbs	1001	Musician
Jones	1001	Musician
Carson	1001	Librarian

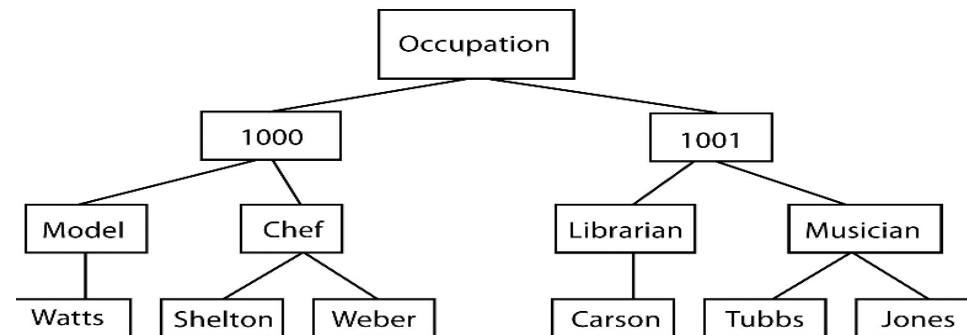
# Hierarchical Data Model

- Hierarchical Data Model

- The hierarchical data model is based on links between the various components / entities in the application.
- The Hierarchical Model structures data as a tree of records, with one-way links between them.

- Various problems associated with this data model.

- The links can be designed to expand vertically or horizontally.
- The only links in a hierarchical data model can be to a superior level or a subordinate level. This model is therefore not appropriate for defining real world entities and relationships.
- Additionally, on the deletion of a higher level entity, all its lower level entities stand to be orphaned.



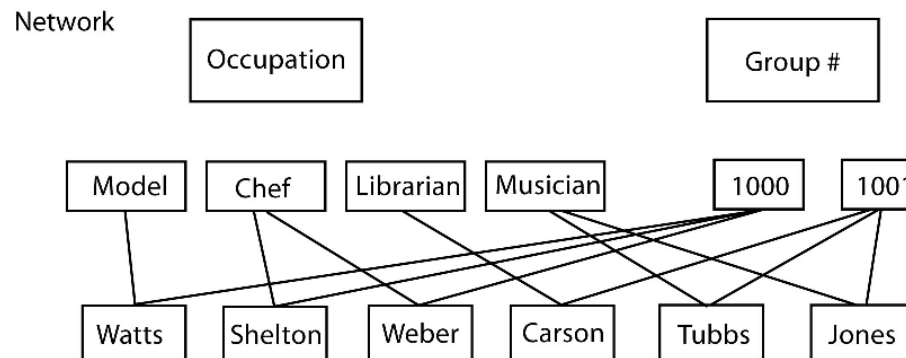


- Network Data Model

- The Network Data Model was conceived as a flexible way of representing objects and their relationships.
- Relationships between entities are not restricted by supporting “many-to-many” relationships which allows greater search flexibility within the dataset

- Various problems associated with this data model.

- Network models can become incredibly complex depending on the size of the databases and the number of interactions between the data points.



# DBMS AND RDBMS

	DBMS	RDBMS
1.	DBMS applications store <b>data as file</b> .	RDBMS applications store <b>data in a tabular form</b> .
2.	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3.	<b>Normalization is not</b> present in DBMS.	<b>Normalization is</b> present in RDBMS.
4.	DBMS does <b>not apply any security</b> with regards to data manipulation.	RDBMS <b>defines the integrity constraint</b> for the purpose of ACID (Atomicity, Consistency, Isolation and Durability) property.
5.	DBMS uses file system to store data, so there will be <b>no relation between the tables</b> .	in RDBMS, data values are stored in the form of tables, so a <b>relationship</b> between these data values will be stored in the form of a table as well.
6.	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7.	DBMS <b>does not support distributed database</b> .	RDBMS <b>supports distributed database</b> .
8.	DBMS is meant to be for small organization and <b>deal with small data</b> . it supports <b>single user</b> .	RDBMS is designed to <b>handle large amount of data</b> . it supports <b>multiple users</b> .
9.	Data Redundancy is common in this model leading to difficulty in maintaining the data.	Keys and indexes are used in the tables to avoid redundancy.
10.	Example DBMS are dBase, Microsoft Access, LibreOffice Base, FoxPro.	Example RDBMS are SQL Server, Oracle , MySQL, Maria DB, SQLite.

# Transactions –ACID properties

- **Atomicity** : What if the OS crashed after \$100 was deposited to the first account'?  
DBMS must assure that the \$100 is withdrawn from the first account.
- **Consistency**: What if a transaction just deposited \$100 into an account'?  
The programmer must ensure that all transactions are consistent
- **Isolation**: What if another transaction computed the total bank balance after \$100 was deposited to the first account'?
- **Durability**: What if, after the commit, the OS crashed before the withdrawal was written to disk'?  
DBMS must assure that the withdrawal was at least logged.

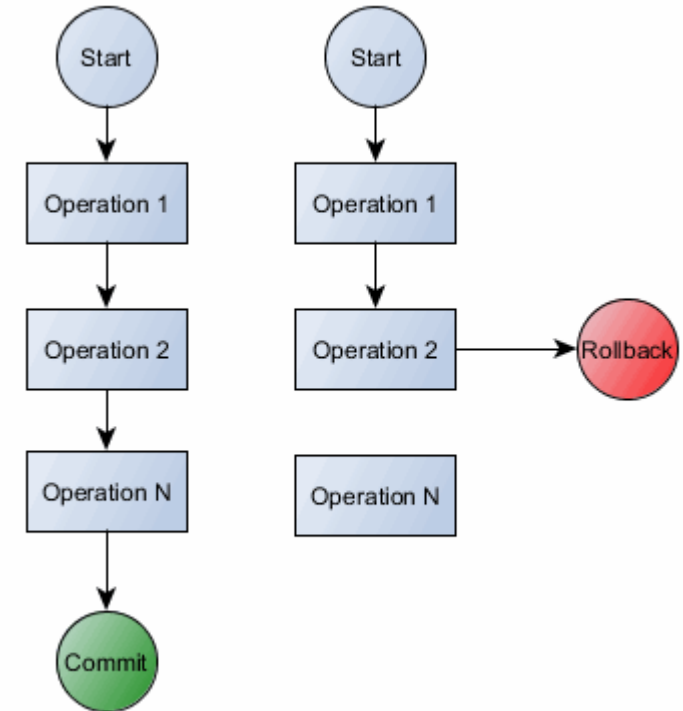
# Transactions –ACID properties

**Atomic** : A transaction occurs entirely or not at all

**Consistency** : each transaction preserves the consistency of the database

**Isolated** : concurrent transactions do not interface with each other

**Durable** : once completed, a transaction's changes are permanent

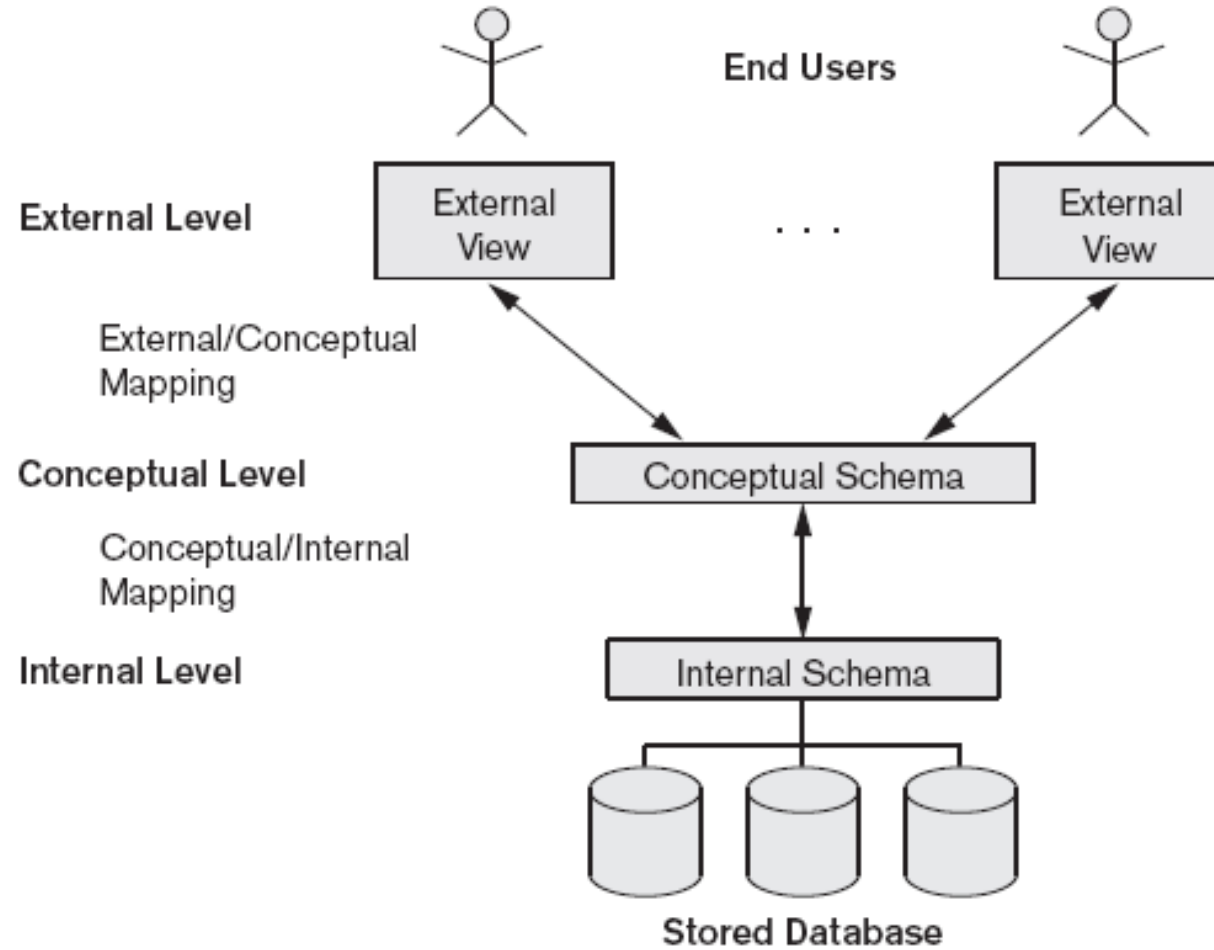


# Relational Database Model

- A **relational database management system (RDBMS)** is the basis for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by **E. F. Codd**
- RDBMS is a collection of tables that are connected in such a way that that data can be accessed without reorganization of the tables. The tables are created such that each column represents a particular attribute

- **A schema** is a description of the data interface to the database
- The **internal schema(Physical schema)** describes the physical grouping of the data and the use of the storage space. This schema level decides the way how the data is actually physically stored on some storage medium
- The **conceptual(logical) schema** describes the basic construction of the data structures or database objects. This schema level describes what data are stored in the database in terms of objects, and what relationships exist among those objects
- The **external schema(application)** of a specific application, generally, only highlights that part of the logical scheme which is relevant for its application.

# RDBMS Architecture



# Schema Architecture and Data Independence

**Data independence:** possibility to change the schema at one level without having to change it at the next higher level (nor having to change programs that access it at that higher level)

**logical data independence:** an external schema (and programs that access it) is insulated from changes that does not concern it in in the community schema (and in the physical schema)

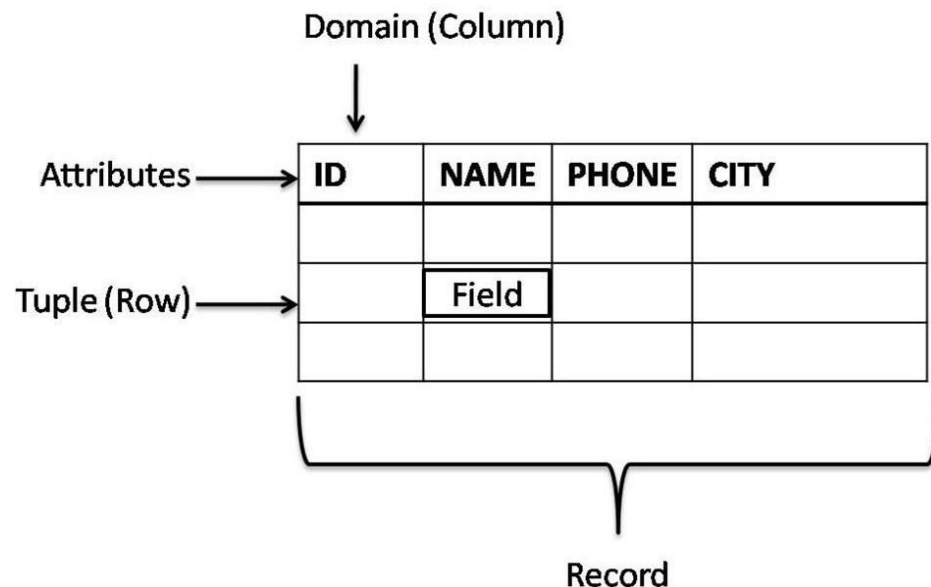
**Physical data independence:** the community and external schemas are insulated from changes in the physical schema





# Relational Model

- Relational Model defines a database as a **collection of tables (relations)** which contain all data composed of horizontal **tuples** commonly known as **rows**, and vertical **attributes**, commonly known as **columns**.



P-ID	Name	Prenome	City
1	Müller	Hans	Oberdorf
2	Meier	Jakob	Hinterwil
3	Keiser	Josef	Unterdorf
...	...	...	...

# Relational Model

- The TABLE is an entirely logical structure.
- All the data including the metadata about the table itself is scattered over different areas of the RDBMS, and not in one single location.
- All the **metadata** is stored in the RDBMS' **Data Dictionary tables**.
- The complexities of how the data is actually stored and later retrieved are handled by the RDBMS.
- AS the end user, we need not concern ourselves with pulling the data from
- different locations.



## Module 2

### Normalization and database design

# Why Normalize ? – Avoiding Data Anomalies

- Normalization is the process of removing redundant data from your tables to improve storage efficiency, data integrity, and scalability.
- Normalization generally involves splitting existing tables into multiple ones, which must be re-joined or linked each time a query is issued.

# Types Of Anomalies

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.

- **Redundancy**
- **Update anomaly**
- **Insert anomaly**
- **Delete Anomaly**

# Steps of Normalization

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)
- Sixth Normal Form (6NF)

- However, in most practical applications, normalization **achieves its best** in **3<sup>rd</sup> Normal Form**.

# Normalization

Project	Project	Project	Project	Employee	Employee	Dept	Dept	Hourly
PC010	Pensions	M Phillips	24500	S10001	A Smith	L004	IT	22
	Pensions		24500	S10030	L Jones	L023	Pensio	18.5
	Pensions		24500	S21010	P Lewis	L004	IT	21
PC045	Salaries	H Martin	17400	S10010	B Jones	L004	IT	21.75
	Salaries		17400	S10001	A Smith	L004	IT	18
	Salaries		17400	S31002	T Gilbert	L028	Datab	25.5
	Salaries		17400	S13210	W Richards	L008	Salary	17
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Datab	23.25
	HR System		12250	S21010	P Lewis	L004	IT	17.5
	HR System		12250	S10034	B James	L009	HR	16.5

# Unnormalized Table

Example of a table not in 1NF

Project Code	Projec	Project	Projec	Emplo	Emplo	Depart	Depart	Hourly
PC010	Pension	M Phillips	24500	S10001	A Smith	L004	IT	22
PC010	Pension	M Phillips	24500	S10030	L Jones	L023	Pension	18.5
PC010	Pension	M Phillips	24500	S21010	P Lewis	L004	IT	21
PC045	Salaries	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries	H Martin	17400	S10001	A Smith	L004	IT	18
PC045	Salaries	H Martin	17400	S31002	T	L028	Databa	25.5
PC045	Salaries	H Martin	17400	S13210	W	L008	Salary	17
PC064	HR	K Lewis	12250	S31002	T	L028	Databa	23.25
PC064	HR	K Lewis	12250	S21010	P Lewis	L004	IT	17.5
PC064	HR	K Lewis	12250	S10034	B	L009	HR	16.5

- It violates the 1NF because:
  - It has multivalued columns or Attribute values are not single.
  - Repeating groups exists.



## First Normal Form (1 NF)

- **Rule**
- An attribute (column) of a table cannot hold multiple values.
- It should hold only atomic values.

Project Code	Project Title	Project Manager	Project Budget	Employee Number	Employee Name	Dept No	Dept Name	Hourly Rate
PC010	Pension	M Phillips	24500	S10001	A Smith	L004	IT	22
PC010	Pension	M Phillips	24500	S10030	L Jones	L023	Pensio	18.5
PC010	Pension	M Phillips	24500	S21010	P Lewis	L004	IT	21
PC045	Salaries	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries	H Martin	17400	S10001	A Smith	L004	IT	18
PC045	Salaries	H Martin	17400	S31002	T Gilbert	L028	Datab	25.5
PC045	Salaries	H Martin	17400	S13210	W Richards	L008	Salary	17
PC064	HR	K Lewis	12250	S31002	T Gilbert	L028	Datab	23.25
PC064	HR	K Lewis	12250	S21010	P Lewis	L004	IT	17.5
PC064	HR	K Lewis	12250	S10034	B James	L009	HR	16.5

## Second Normal Form (2 NF )

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key

### Repeating Attributes Removed



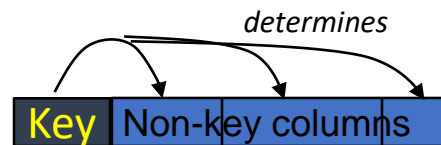
<b>Project Code</b>	<b>Project Title</b>	<b>Project Manager</b>	<b>Project Budget</b>
PC010	Pension	M Phillips	24500
PC045	Salaries	H Martin	17400
PC064	HR	K Lewis	12250

<b><u>Project Code</u></b>	<b>Employee Number</b>	<b>Employee Name</b>	<b>Dept No</b>	<b>Dept Name</b>	<b>Hourly Rate</b>
PC010	S10001	A Smith	L004	IT	22
PC010	S10030	L Jones	L023	Pensions	18.5
PC010	S21010	P Lewis	L004	IT	21
PC045	S10010	B Jones	L004	IT	21.75
PC045	S10001	A Smith	L004	IT	18
PC045	S31002	T Gilbert	L028	Database	25.5
PC045	S13210	W Richards	L008	Salary	17
PC064	S31002	T Gilbert	L028	Database	23.25
PC064	S21010	P Lewis	L004	IT	17.5
PC064	S10034	B James	L009	HR	16.5

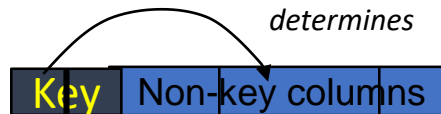
Before we Move to 2NF understand **Functional Dependency**

# Functional Dependency

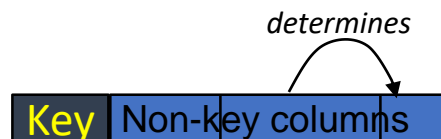
- Functional Dependence:
  - The key column must be sufficient for determining values of the non-key columns.
  - An attribute depends on another attribute if the change of its value is caused by a change of that other attribute's value.
- From this perspective, the goal of the normalization process is to establish the *Full Functional Dependence* between the key and non-key columns, driving out other dependencies (*Partial Functional Dependence*, and *Transitive Dependence*).



***Full Functional Dependence***  
(Key determines non-key values)



***Partial Functional Dependence***  
(Part of Key determines some non-key values)



***Transitive Dependence*** (Non-key attribute determines another non-key)

## Example 1:for Functional Dependency

Rollno	StudentName	Course	Marks	Grade
1	Tom	Maths	91	A
2	Peter	Maths	76	B
3	John	Maths	58	C
4	Michel	Maths	70	B
5	Hary	Maths	49	D
1	Tom	Science	77	B
2	Peter	Science	96	A
3	John	Science	38	D
4	Michel	Science	75	B
5	Hary	Science	59	C

Marks attribute is not a key attribute, but marks determines grade, Hence it can be concluded that key attributes are determinants but not all the determinants are key attributes.

Also, Rollno and Course together (called **composite key** attribute) defines EXACTLY ONE value of marks. This can be symbolically represented as **Rollno,Course** → **Marks**. Thus, a determinant can be composite in nature

## Example 2: for Functional Dependency

Order_No	Order_date	Item_code	Quantity	Price_per_unit
1456	32565	3687	52	50.4
1456	32565	4627	38	60.2
1456	32565	3214	20	17.5
1886	33667	4629	45	20.25
1886	33667	4627	30	60.2
1788	33698	4627	40	60.2

The following Functional Dependencies exist :

Order\_No  $\rightarrow$  Order\_date, Item\_code  $\rightarrow$  Price\_per\_unit, Order\_No+Item\_code  $\rightarrow$  Quantity

## Continued with our example of Functional Dependency

We say an attribute, B, has a *functional dependency* on another attribute, A, if for any two records, which have the same value for A, then the values for B in these two records must be the same.

<b>Project</b>	<b>Employee</b>	<b>Employee</b>	<b>Dept No</b>	<b>Dept</b>	<b>Hourly</b>
PC010	S10001	A Smith	L004	IT	22
PC010	S10030	L Jones	L023	Pensions	18.5
PC010	S21010	P Lewis	L004	IT	21
PC045	S10010	B Jones	L004	IT	21.75
PC045	S10001	A Smith	L004	IT	18
PC045	S31002	T Gilbert	L028	Database	25.5
PC045	S13210	W Richards	L008	Salary	17
PC064	S31002	T Gilbert	L028	Database	23.25
PC064	S21010	P Lewis	L004	IT	17.5
PC064	S10034	B James	L009	HR	16.5

Employee name, Department No and Department Name are dependent upon Employee No only.

Remove any **-key attributes (partial Dependencies)** that only depend on part of the key to a new table.

## Third Normal Form (3 NF)

<u>Project Code</u>	Project Title	Project Manager	Project Budget
PC010	Pensions	M Phillips	24500
PC045	Salaries	H Martin	17400
PC064	HR System	K Lewis	12250

<u>Project Code</u>	Employee No.	Hourly Rate
PC010	S10001	22
PC010	S10030	18.5
PC010	S21010	21
PC045	S10010	21.75
PC045	S10001	18
PC045	S31002	25.5
PC045	S13210	17
PC064	S31002	23.25
PC064	S21010	17.5
PC064	S10034	16.5

Employee No.	Employee Name	Depa No.	Depa Name
S10001	A Smith	L004	IT
S10030	L Jones	L023	Pensio
S21010	P Lewis	L004	IT
S10010	B Jones	L004	IT
S31002	T Gilbert	L028	Datab
S13210	W Richards	L008	Salary
S10034	B James	L009	HR

# Boyce Codd Normal Form (BCNF)

A relation is said to be in Boyce Codd Normal Form (BCNF) if and only if all the determinants are candidate keys. BCNF relation is a strong 3NF, but not every 3NF relation is BCNF.

StudentId	EmailID	CourseID	Marks
101	Davis@myuni.edu	M4	82
102	Daniel@myuni.edu	M4	62
101	Davis@myuni.edu	H6	79
103	Sandra@myuni.edu	C3	65
104	Evelyn@myuni.edu	B3	77
102	Daniel@myuni.edu	P3	68
105	Susan@myuni.edu	P3	89
103	Sandra@myuni.edu	B4	54
105	Susan@myuni.edu	H6	87
104	Evelyn@myuni.edu	M4	65

StudentId	EmailID
101	Davis@myuni.edu
102	Daniel@myuni.edu
103	Sandra@myuni.edu
104	Evelyn@myuni.edu
105	Susan@myuni.edu

Student Id	CourseID	Marks
101	M4	82
102	M4	62
101	H6	79
103	C3	65
104	B3	77
102	P3	68
105	P3	89
103	B4	54
105	H6	87
104	M4	65



# Fourth Normal Form

## Properties of 4NF:-

- there is no multivalued dependency(**a** --->> **b**) in the relation or
- there are multivalued dependencies but the attributes, which are multivalued dependent on a specific attribute, are dependent between themselves.

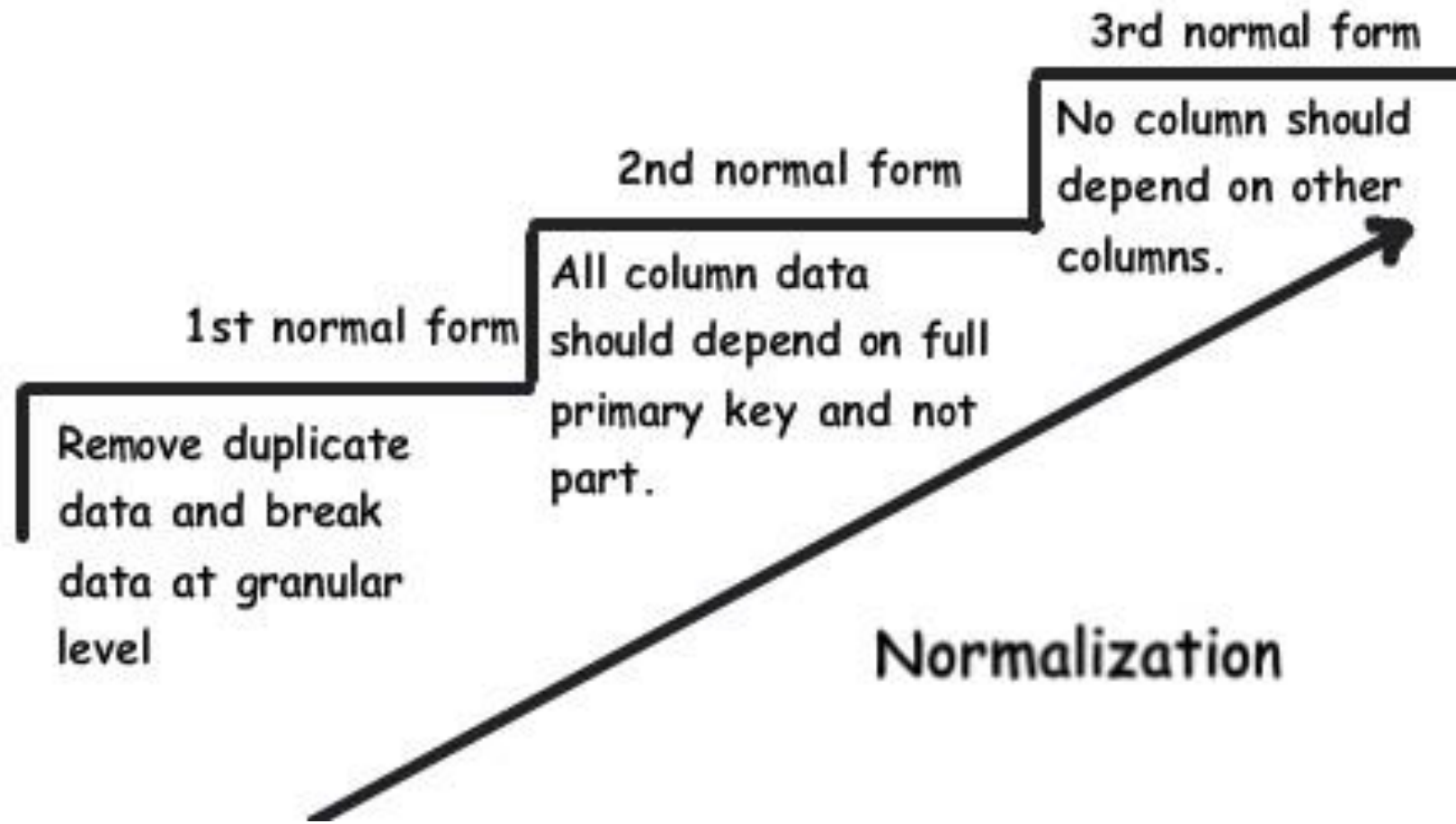
# Fifth Normal Form

- A table is in fifth normal form (5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables.

## Properties of 5NF:-

- Anomalies can occur in relations in 4NF if the primary key has three or more fields.
- 5NF is based on the concept of join dependence - if a relation cannot be decomposed any further then it is in 5NF.
- Pair wise cyclical dependency means that:
  - For any one you must know the other two (cyclical).

# Summery



# Summary of Normal Forms

- 1) If a table has repeating sections, there is huge redundancy, different classes are mixed together, and all anomalies occur. Split the table, so that classes are clearly differentiated. Result: **1NF**.

**1NF:** A table is in 1NF if it does not have repeating sections.

- 2) If a table has a combined key, non-key columns may depend on just a part of the primary key, and so there is partial functional dependency. Split the table so that in new tables non-keys depend on the entire key. Result: **2NF**.

**2NF:** A table is in 2NF if it is in 1NF and non-key columns depend on the entire combined key.

- 3) If a non-key depends on another non-key, there is transitive dependency. Split the table so that in new tables each non-key depends on the key and nothing but the key. Result: **3NF**.

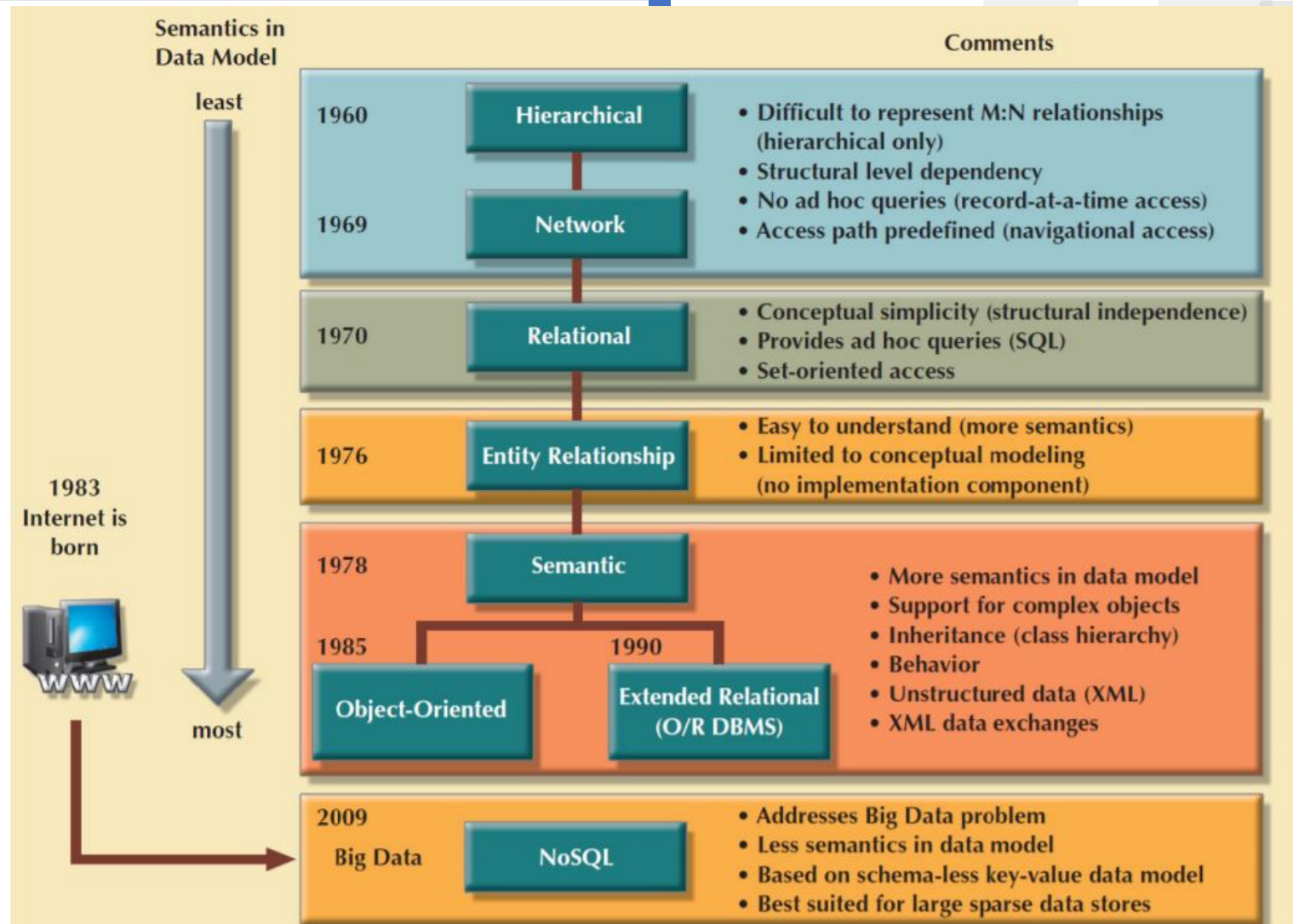
**3NF:** A table is in 3NF if it is in 2NF and all non-key columns depend on the key only.



## Module 2

Database design

# Evolution of ER Models



# Entity Relationship Model

- An **entity–relationship model (ER model)** is a systematic way of describing and defining a business process.
- Widely accepted standard for data modeling
- Introduced by Chen in 1976
- Graphical representation of entities and their relationships in a database structure
- Entity relationship diagram (ERD) – Uses graphic representations to model database components – **Entity** is mapped to a **relational table**

# Relationships

- A relationship is an association among (two or more) entities.

## Types :

- **1:1**
  - An academic department is chaired by one professor; a professor may chair only one academic department.
- **1:M**
  - A customer may generate many invoices; each invoice is generated by one customer.
- **M:N**
  - An employee may have earned many degrees; a degree may have been earned by many employees.



# Keys

- **Superkey**
- **Composite key**
- **Candidate key**
- **Primary key**
- **Alternate key**
- **Secondary key**
- **Foreign key**



# Data Model Basic Building Blocks Terminology

- **Entity:** anything about which data are to be collected and stored
- **Attribute:** a characteristic of an entity
- **Relationship:** describes an association among entities
  - – One-to-many (1:M) relationship
  - – Many-to-many (M:N or M:M) relationship
  - – One-to-one (1:1) relationship
- **Constraint:** a restriction placed on the data

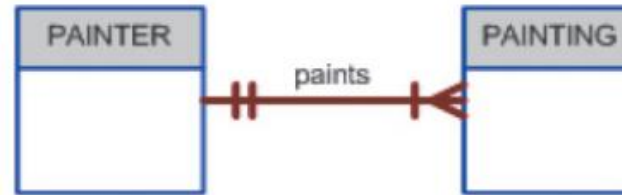
# ER Model Notations

## Chen Notation

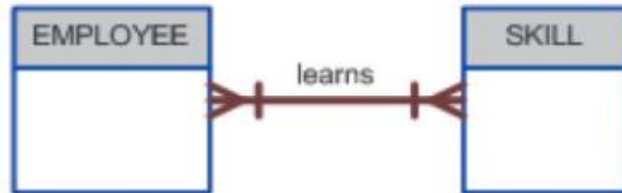
## Crow's Foot Notation

## UML Class Diagram Notation

**A One-to-Many (1:M) Relationship:** a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.




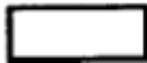









**A Many-to-Many (M:N) Relationship:** an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.



**A One-to-One (1:1) Relationship:** an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.



# Notations Used in ER Model

<b>Rectangle</b> represents entity type	
<b>Double/Bold rectangle</b> represent weak entity type.	 or 
<b>Diamond</b> represent', relationship type.	
<b>Double/Bold diamond</b> represents weak relationship type.	 or 
<b>Ellipse</b> represents attribute-type.	
<b>Double ellipse</b> represents multivalued attribute:	
<b>Dashed ellipse</b> denotes-derived attribute	
<b>Line</b> a link attribution entity sets and entity sets to relationship sets	
<b>Double lines</b> which indicate total participation of an entity in a relationship set <i>i.e.</i> , each entity in the entity set occurs in atleast one relationship in that relationship set.	

## Example : Requirement of business model -The Flight Database

The **flight database** stores details about an **airline's fleet, flights, and seat bookings**. Consider the following requirements list:

- The airline has one or more airplanes.
- An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
- An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
- Each flight is carried out by a single airplane.
- A passenger has given names, a surname, and a unique email address.
- A passenger can book a seat on a flight.

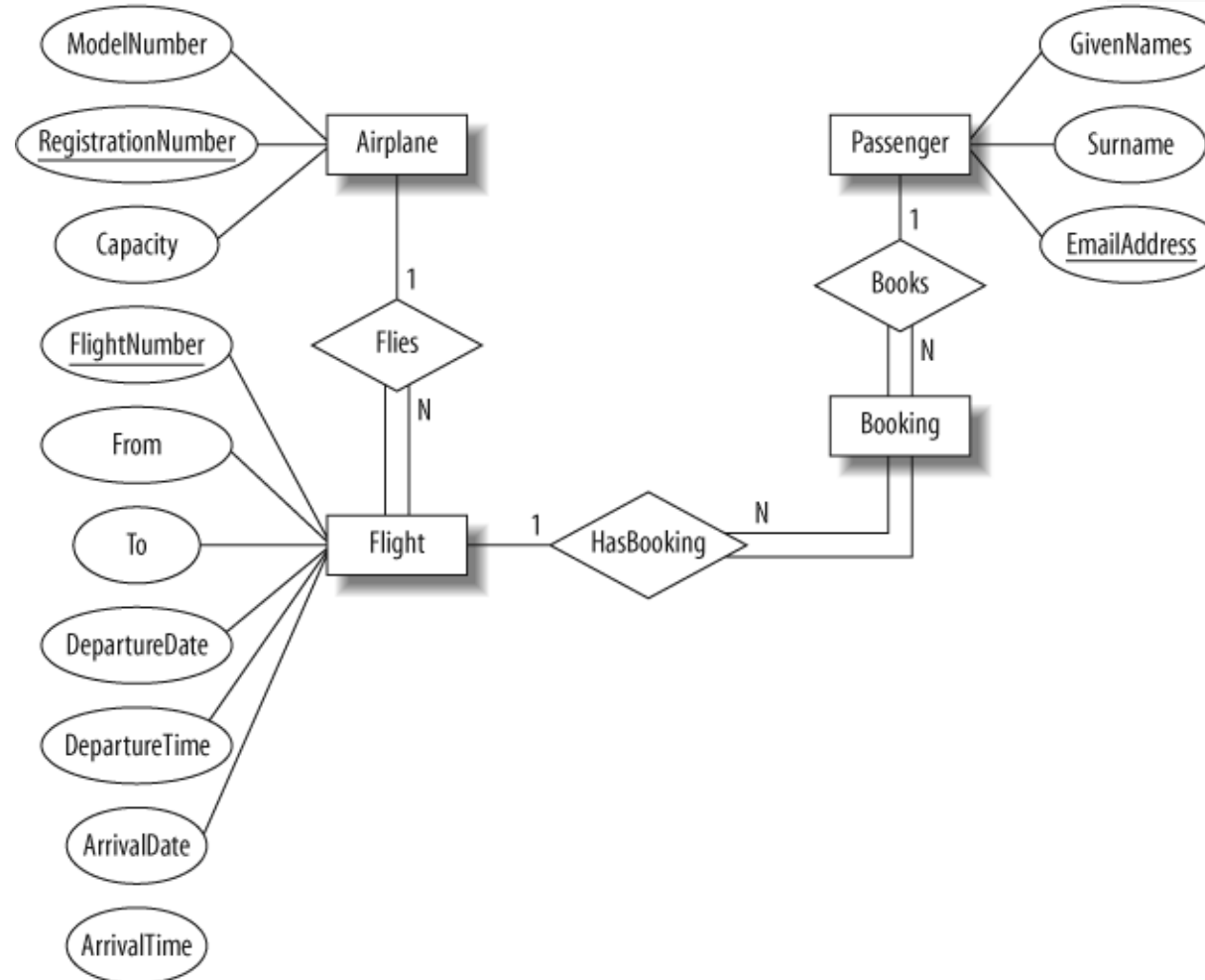
## Explanation of above ER Model

- An Airplane is uniquely identified by its RegistrationNumber, so we use this as the primary key.
- A Flight is uniquely identified by its FlightNumber, so we use the flight number as the primary key.
- The departure and destination airports are captured in the From and To attributes,
- and we have separate attributes for the departure and arrival date and time.
- Because no two passengers will share an email address,  
we can use the EmailAddress as the primary key for the Passenger entity.
- An airplane can be involved in any number of flights, while each flight uses exactly one airplane,  
so the Flies relationship between the Airplane and Flight relationships has cardinality 1:N;  
because a flight cannot exist without an airplane,  
the Flight entity participates totally in this relationship.
- A passenger can book any number of flights,  
while a flight can be booked by any number of passengers.  
we could specify an M:N Books relationship between the Passenger and Flight relationship,
- Considering the issue more carefully shows that there is a hidden entity here: the booking itself.  
We capture this by creating the intermediate entity Booking  
and 1:N relationships between it and the Passenger and Flight entities.

- An **Airplane** is uniquely identified by its **RegistrationNumber**, so we use this as the **primary key**.
- A **Flight** is uniquely identified by its **FlightNumber**, so we use the flight number as the **primary key**.
- The departure and destination airports are captured in the **From and To attributes**, and we have separate attributes for **the departure and arrival date and time**.
- Because no two passengers will share an email address, we can use the **Email Address** as the **primary key** for the Passenger entity.
- An airplane can be involved in any number of flights, **while each flight uses exactly one airplane**, so the Flies relationship between the Airplane and Flight relationships has cardinality **1:N**;
- A passenger can book any number of flights, while a flight can be booked by any number of passengers. we could specify **an M:N Books** relationship between the **Passenger** and **Flight** relationship.
- There is a hidden entity here: the **booking** itself. **1:N** relationships between **Book** entity and the **Passenger and Flight** entities.

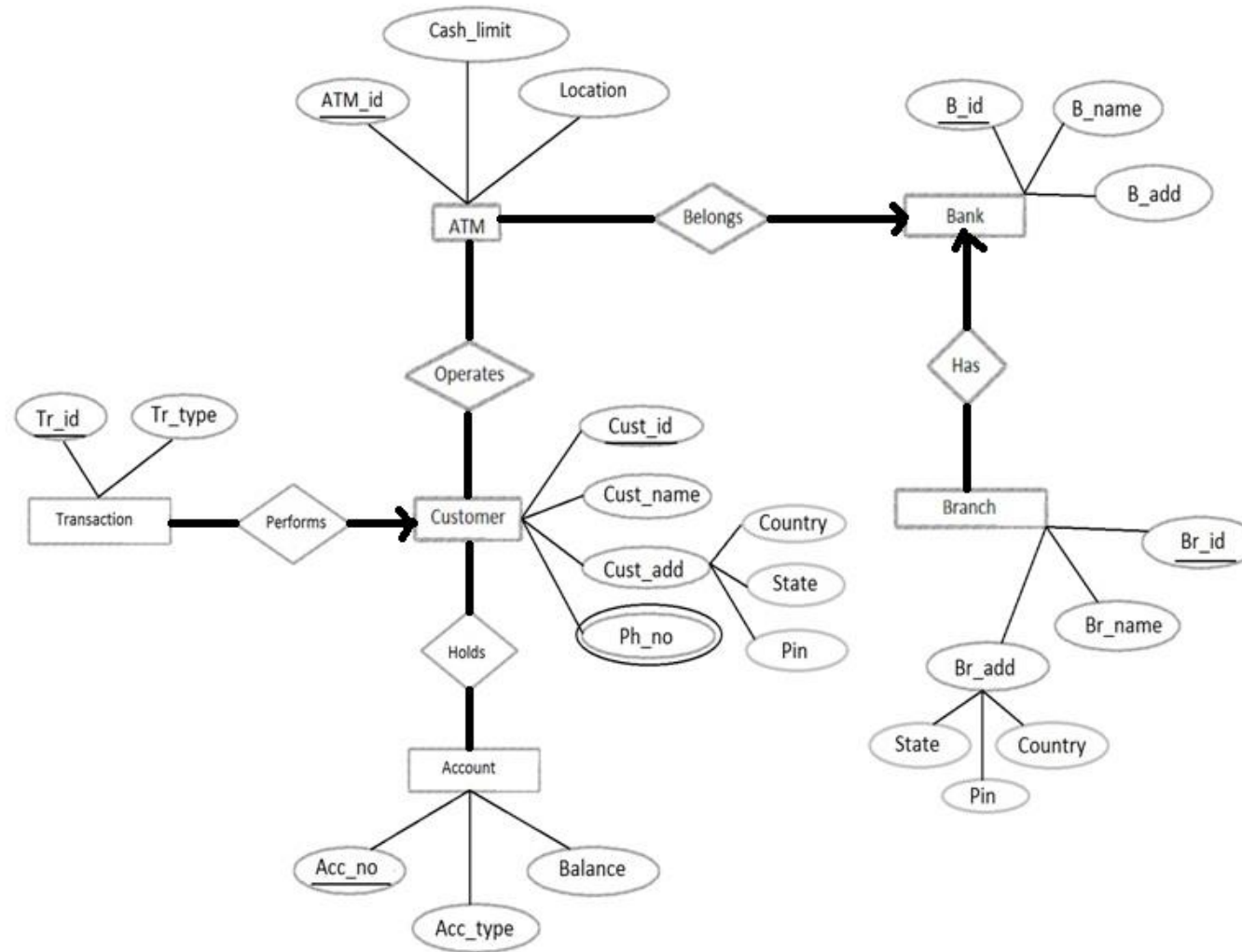
# The Flight Database

The ER diagram derived from our requirements





# Example: Self study



# Module 3

## Basic SELECT operations

# SQL Commands :

- **Data Query Language(DQL)**
  - SELECT
- **Data Definition Language (DDL)**
  - CREATE, ALTER, DROP, TRUNCATE, RENAME, COMMENT
- **Data Manipulation Language(DML)**
  - INSERT, UPDATE, DELETE, (MERGE)
- **Data Control Language(DCL)**
  - GRANT, REVOKE
- **Transaction control Language(TCL)**
  - COMMIT, ROLLBACK, SAVEPOINT

# Capabilities of SQL `SELECT` Statements

## Projection

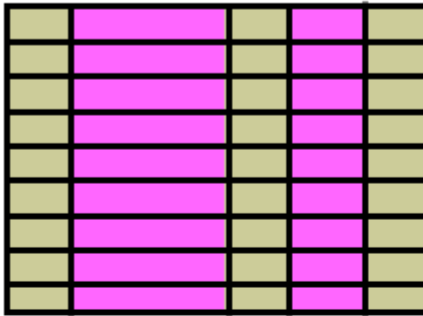



Table 1

## Selection

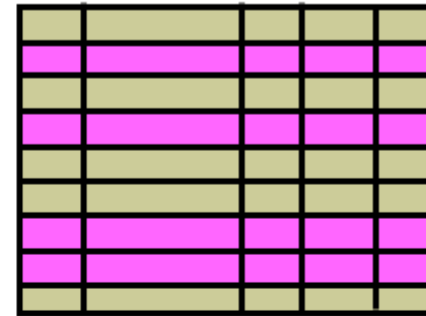



Table 1

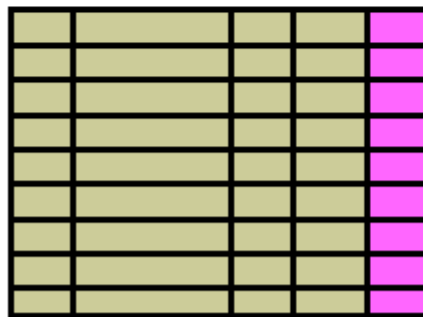



Table 1

Join

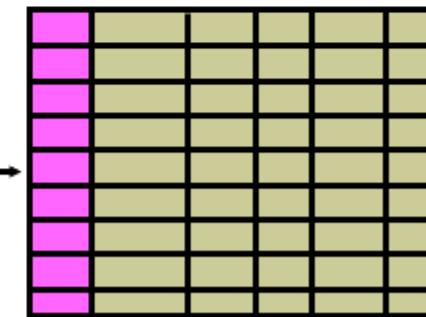



Table 2

# Basic SELECT Command

SELECT is used for retrieving data from database tables

- SELECT identifies the columns to be displayed.
- FROM identifies the table containing those columns

## SELECT

[DISTINCT | ALL] <column-list>

FROM <table-names>

[WHERE <condition>]

[ORDER BY <column-list>]

[GROUP BY <column-list>]

[HAVING <condition>]

- ([ ] - optional, | - or)

# Selecting All Columns

- Selecting All Columns
  - `SELECT * FROM employees;`
- Selecting specific Columns
- The columns to be selected from a table are specified after the keyword select. This operation is also called **projection**.
  - `SELECT first_name, last_name, hire_date FROM employees;`

## WHERE clause- Filtering Data

- For selection of tuples that satisfy certain conditions, the where clause is used.
- In a where clause simple conditions based on comparison operators can be combined using the logical connectives and, or, and not to form complex conditions.
- Conditions may also include pattern matching operations and even subqueries
- `SELECT * FROM employees WHERE rownum <11;`

## ORDER BY clause

- Order by clause has one or more attributes listed in the select clause as parameter.
- desc specifies a descending order
- **asc** specifies an ascending order (default order).
  - `SELECT first_name, department_id as , hire_date from employees order by department_id ;`
- **Sorting by Multiple Columns**
  - `SELECT first_name, department_id as , hire_date from employees order by department_id asc, hire_date desc;`



# Distinct clause

- If a table has duplicate data, you can use the DISTINCT clause to eliminate the duplicate values
- `select DISTINCT job_id from employees`
- Distinct on multiple columns
- `select DISTINCT job_id,department_id from employees ORDER BY job_id`

# Comparison Operators

- Comparison operators compare a certain column value with several other column values.
- These are the main comparison operators:
  - **BETWEEN:** Tests whether a value is between a pair of values
  - **IN:** Tests whether a value is in a list of values
  - **LIKE:** Tests whether a value follows a certain pattern, as shown here:

# Comparison operators

Operator	Description
=	Equal To
<>	Not Equal To
!=	Not Equal To
>	Greater Than
<	Less than
>=	Greater or equal
<=	Lesser or equal

```
SELECT employee_id FROM employees WHERE salary = 50000;  
SELECT employee_id FROM employees WHERE salary < 50000;  
SELECT employee_id FROM employees WHERE salary >= 50000;  
SELECT employee_id FROM employees WHERE salary != 50000;
```

# IN /NOT IN and Between Operator

- If any item in the list following a NOT IN operation is null, all rows evaluate to UNKNOWN (and no rows are returned).
- You use the BETWEEN operator to select rows whose column value is inclusive within a specified range.
- The range specified with the BETWEEN operator is inclusive.

```
select * from dept where deptno IN (20,30);  
select * from dept where deptno NOT IN (20,30);  
select * from emp where SAL between 1500 and 2500;  
select * from Emp where HIRE_DATE between '02-01-03' and '02-01-04';
```

# The LIKE Operator

The LIKE condition uses pattern matching to restrict rows in a SELECT statement.

```
SELECT employee_id, last_name FROM employees WHERE last_name LIKE 'Fa%';
```

- The pattern should be enclosed in single quotes (' ').
- The percent sign (%) acts as a wildcard for one or more characters
- The asterisk (\*) acts as many or all characters .
- The single underscore character (\_) acts as a wildcard for one and only one character.

# NULL value

**NULL is a special value** that means:

- unavailable
- unassigned
- unknown
- Not Applicable

NULL **value is not equivalent** to

- zero
- blank space

The operations **= null** and **!= null** are **not defined!**

```
select employee_id,manager_id from employees where manager_id IS NOT NULL;
```

# Logical Operators

- The logical operators, also called **Boolean operators**, logically compare two or more values.
- The main logical operators are **AND, OR, NOT**
- When there are multiple operators within a single statement, you need rules of precedence.
- The following is the order of precedence of operators in Oracle, with the most important first:
- =, !=, <, >, <=, >= IS NULL, LIKE, BETWEEN, IN, EXISTS NOT AND OR
  - `SELECT * FROM employees WHERE JOB_ID= 'AC_MGR' AND department_id = 110 ;`
  - `SELECT * FROM employees WHERE JOB_ID= 'AC_MGR' OR department_id = 110 ;`

# A COLUMN ALIAS

- Renames a column heading
- Is useful with calculations
- Immediately follows the column name (There can also be the optional AS keyword between the column name and alias.)
- Requires double quotation marks (" ") if it contains spaces or special characters, or if it is case-sensitive

```
SELECT last_name AS Surname, commission_pct commission FROM employees  
WHERE commission_pct IS NOT NULL;
```



# Oracle Built-in Datatypes

**CHAR** (*size*)

fixed-length char array

**VARCHAR2** (*size*)

Variable-length char string

**NUMBER** (*precision, scale*)

any numeric

**DATE**

date

**TIMESTAMP**

date+time

**CLOB**

char large object

**BLOB**

binary large object

**BINARY\_FLOAT**

32 bit floating point

**BINARY\_DOUBLE**

64 bit floating point

... + *some others*



# Conversion Functions

- The most common of these functions are:
  - **TO\_CHAR** function converts a floating number to a string.
  - **TO\_NUMBER** function converts a floating number or a string to a number.
  - **TO\_DATE** function converts character data to a DATE data type.
- 
- `SELECT TO_CHAR(TO_DATE('20-JUL-08', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;`
  - `SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY') FROM DUAL;`

## BUILT-IN Oracle functions

- **Built-In Oracle functions** help you perform many transformations quickly, without your having to do any coding.

Single-row functions,

Aggregate functions,

Number functions

Date functions,

General and Conditional functions,

Analytical functions.

# Single-Row Functions

Single-row functions are typically used to perform tasks such as converting a lowercase word to uppercase or vice versa, or replacing a portion of text in a row.

# Character Functions :

Function	Description
CONCAT	Allows you to concatenate two strings together
CONVERT	Converts a string from <u>one character</u> set to another
INITCAP	Sets the first character in each word to uppercase and the rest to lowercase
INSTR	Returns the location of a substring in a string
LENGTH	Returns the length of the specified string
LOWER	Converts all letters in the specified string to lowercase
LPAD	Pads the left-side of a string with a specific set of characters
LTRIM	Removes all specified characters from the left-hand side of a string
REGEXP_INSTR	Returns the location of a regular expression pattern in a string
REGEXP_REPLACE	Allows you to replace a sequence of characters in a string with another set of characters using regular expression pattern matching
REGEXP_SUBSTR	Allows you to extract a substring from a string using regular expression pattern matching

# Character Functions :

REPLACE	Replaces a sequence of characters in a string with another set of characters
RPAD	Pads the right-side of a string with a specific set of characters
RTRIM	Removes all specified characters from the right-hand side of a string
SUBSTR	Allows you to extract a substring from a string
TRANSLATE	Replaces a sequence of characters in a string with another set of characters
TRIM	Removes all specified characters either from the beginning or the end of a string
UPPER	Converts all letters in the specified string to uppercase
VSIZE	Returns the number of bytes in the internal representation of an expression

# Number Functions

Function	Description
COUNT	Returns the count of an expression
EXP	Returns $e$ raised to the power of number
GREATEST	Returns the greatest value in a list of expressions
LEAST	Returns the smallest value in a list of expressions
MOD	Returns the remainder of $n$ divided by $m$
POWER	Returns $m$ raised to the $n$ th power
ROUND	Returns a number rounded to a certain number of decimal places
SQRT	Returns the square root of a number
SUM	Returns the summed value of an expression
TRUNC	Returns a number truncated to a certain number of decimal places

# Date Functions

Function	Description
ADD_MONTHS	Returns a date with a specified number of months added
CURRENT_DATE	Returns the current date in the time zone of the current SQL session as set by the ALTER SESSION command
ROUND	Returns a date rounded to a specific unit of measure
LAST_DAY	Returns the last day of the month based on a date value
MONTHS_BETWEEN	Returns the number of months between date1 and date2
NEXT_DAY	Returns the first weekday that is greater than a date
SYSDATE	Returns the current system date and time on your local database
SYSTIMESTAMP	Returns the current system date and time (including fractional seconds and time zone) on your local database
TRUNC	Returns a date truncated to a specific unit of measure



# Aggregate Functions

- To compute things such as averages and totals of a selected column in a query.
  - **MIN:** The MIN function returns the smallest value.
  - **MAX:** The MAX function returns the largest value.
  - **AVG:** The AVG function computes the average value of a column.
  - **SUM:** The SUM function computes the sum of a column:
  - **COUNT:** The COUNT function returns the total number of columns.
  - **COUNT(\*):** The COUNT(\*) function returns the number of rows in a table.

```
SELECT MIN(hire_date) , max(hire_date ) FROM employees;
```

# General and Conditional Functions

- The conditional functions help you decide among several choices.
- **NVL:** The NVL function replaces the value in a table column with the value after the comma if the column is null. Thus, the NVL function takes care of column values if the column values are null and converts them to non-null values:
  - `SELECT last_name,salary , salary * NVL (commission_pct,0)/100 as COMM FROM employees;`
  - `SELECT SALARY,commission_pct, SALARY+NVL(commission_pct,0)`
  - `FROM EMPLOYEES where SALARY>10000;`
  - `SELECT SALARY,commission_pct,NVL2(COMMISSION_PCT,SALARY+(SALARY*COMMISSION_PCT), SALARY)`  
`TOTAL`
  - `FROM EMPLOYEES where SALARY>10000;`
- **COALESCE:** This function is similar to NVL, but it returns the first non-null value in the list:
  - `SELECT last_name, COALESCE(commission_pct, salary, 10) comm FROM employees ORDER BY commission_pct;`

# General and Conditional Functions

- **DECODE:** To incorporate basic if-then functionality into SQL code.

```
SELECT last_name, job_id, salary, DECODE(job_id, 'IT_PROG', 1.10*salary, 'ST_CLERK',  
1.15*salary, 'SA_REP', 1.20*salary,salary) REVISED_SALARY FROM employees;
```

## CASE:

```
SELECT last_name, commission_pct,  
(CASE commission_pct  
WHEN 0.1 THEN 'Low'  
WHEN 0.15 THEN 'Average'  
WHEN 0.2 THEN 'High'  
ELSE 'N/A' END ) Commission  
FROM  
employees  
ORDER BY  
last_name;
```

# Analytical Functions

- **Ranking functions:** These enable you to rank items in a data set according to some criteria.
- Oracle has several types of ranking functions, including RANK, DENSE\_RANK, CUME\_DIST, PERCENT\_RANK, and NTILE.
- **rank() over( [partition by column] order by column )**

```
select first_name, salary,  
       RANK() OVER (PARTITION BY department_id ORDER BY salary)  
from employees  
where department_id = 80;
```

Thank You

