Introduction to RDBMS & MySQL

By

Anand Kulkarni

Table of Content

Module	Topic
Module 1:	DBMS & RDBMS Concepts
Module 2:	Building a Logical Database Model (E-R diagrams)
Module 3:	Database Normalization
Module 4:	Getting Started with MySQL
Module 5:	Data Retrieval and Ordering the Output
Module 6:	Creating Table Structures
Module 7:	Inserting, Modifying and Deleting Data
Module 8:	Modifying Table Structure
Module 9:	Integrity Constraints
Module 10:	Built-in Functions
Module 11:	Joins & Sub Queries

Module 1: DBMS & RDBMS Concepts

- Overview
  - File System
  - Disadvantage of File System
  - Database Management System (DBMS)
  - Instances and Schemas
  - Data Models
  - DML & DDL
  - Introduction to RDBMS
  - Relationship
  - Keys

File System

- Flat Files is a file of data that **does not contain links** to other files or is a non-relational database
- Example

Bob |123 street | California | $200.00
Nathan | 800 Street | Utah | $10.00

Disadvantage of File System

- In the early days, database applications were built directly on top of file systems
- Drawbacks of using file systems to store data:
  - Potential duplication
  - Non-unique records
  - Harder to update
  - Inherently inefficient
  - Harder to change data format
  - Poor at complex queries
  - Security issues

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use
- Database Applications:
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives

DBMS Level of Abstraction

- **Physical level**: describes how a record (e.g., customer) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

  type *customer* = **record**

       customer_id : string;
       customer_name : string;
       customer_street : string;
       customer_city : string;

       end;

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

- An architecture for a database system

View Level

| View 1 | View 2 | .... | View n |

Logical View

Physical View
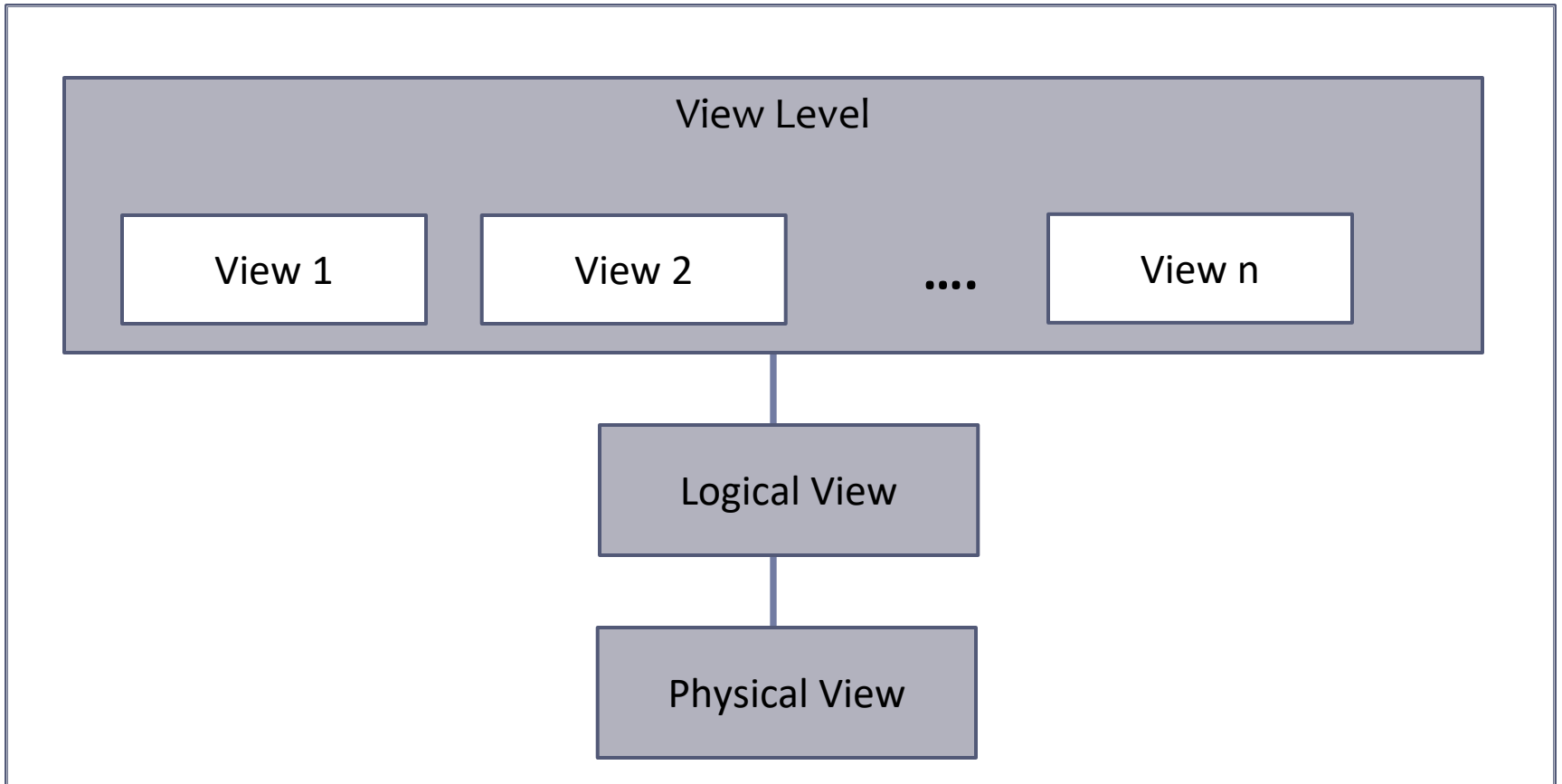
Instances and Schemas

- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them)
- **Instance** – the actual content of the database at a particular point in time

Data Models

- A model is a representation of reality, '**real world**' objects and events, and their associations. It is an abstraction that concentrates on the essential, inherent aspects of an organization and ignore the accidental properties.

- The purpose of a data model is to represent data and to make the data understandable.
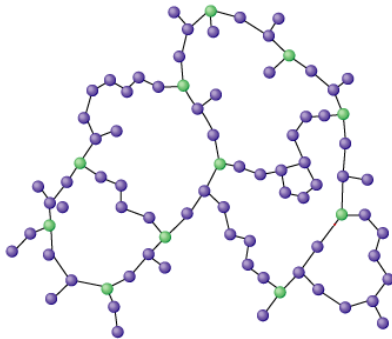
Data Models Continue…

- Data Models are categorized into:

Hierarchical Model →
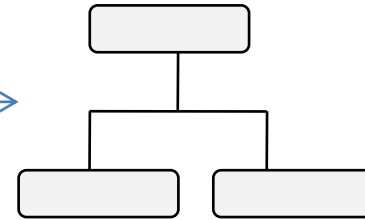
Network Model ←

Relational Model →

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Procedural** – user specifies what data is required and how to get those data
  - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language

Data Definition Language (DDL)

- Specification notation for defining the database schema
  - Example: create table account (
                        account_number   char(10),
                        branch_name   char(10),
                        balance   integer)
- DDL compiler generates a set of tables stored in a data dictionary
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Data storage and definition language
    - Specifies the storage structure and access methods used
  - Integrity constraints
    - Domain constraints
    - Referential integrity (e.g. branch_name must correspond to a valid branch in the branch table)
- Authorization

Introduction to RDBMS

- RDBMS stands for **R**elational **D**ata**b**ase **M**anagement **S**ystem

- A DBMS in which **data** is stored in **tables** and the **relationships** among the data are also stored in **tables**.

- 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.

Relational Database Model

customer_id	customer_name	customer_street	customer_city	account_number
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201

Attribute

Tuple

Sample Relational Database

customer_id	customer_name	customer_street	customer_city
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table
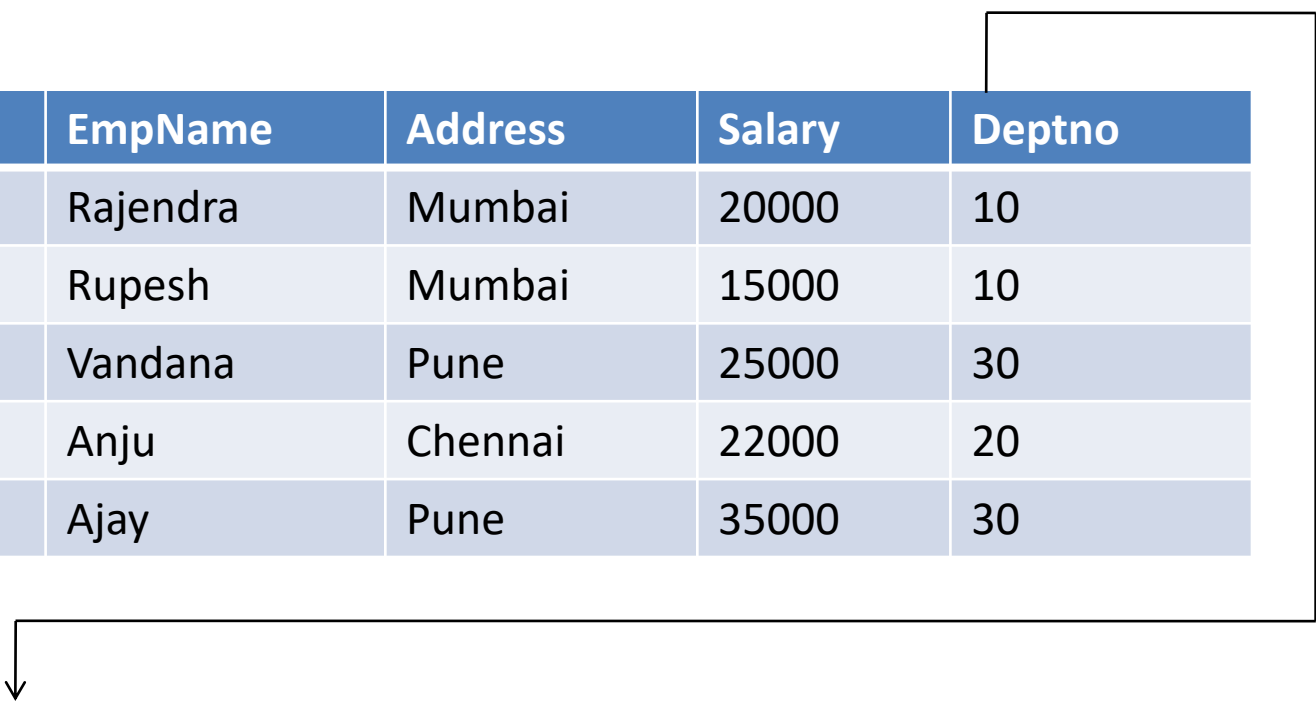
customer_id	account_number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Relationship

EmpCode	EmpName	Address	Salary	Deptno
E001	Rajendra	Mumbai	20000	10
E002	Rupesh	Mumbai	15000	10
E003	Vandana	Pune	25000	30
E004	Anju	Chennai	22000	20
E005	Ajay	Pune	35000	30

Deptno	Dname	Location
10	Admin	Mumbai
20	IT	Pune
30	Sales	Chennai

A relationship is an <u>association</u> between two or more tables.

Keys

- Keys are fundamental part of relational database because they enable tables in the database to be related with each other.
- Types of keys are:
  - **Primary Key**

    Primary key is a key that uniquely identify each record in a table. It cannot be NULL.
  - **Composite Key**

    Key that consist of two or more columns that uniquely identify an entity occurrence is called Composite key.
  - **Foreign Key**

    A foreign key is column or set of columns in a table whose values match a primary key in another table.

Keys continue…

- **Candidate Key**

  Candidate keys are defined as the set of fields from which primary key can be selected.

- **Alternate Key**

  The candidate keys that are not selected as primary key are known as alternate keys.

- **Super Key**

  Super key is a column or combination of columns that identifies a record within the table.

Keys continue…

Id	F_Name	L_Name	Phone	Email	Salary	Address_id
101	John	Obama	11111	a@a.in	10000	505
102	Tom	Bush	22222	b@a.in	20000	302
103	Ivan	Kerry	33333	c@a.in	30000	211

- **Primary Key:** {Id}
- **Composite Key:** {Id, companyId}
- **Foreign Key:** {Address_id}
- **Candidate Key:** {Id}, {F_Name,L_Name}, {Phone}, {Email}
- **Alternate Key:** {F_Name, L_Name}, {Phone}, {Email}
- **Super Key:** {Id}, {F_Name, L_Name}, {Phone}, {Email}, {Id, Salary}, {Id, Phone}, {Id, Email}, {Phone, Email}

Module 2: Building a Logical Database

- Overview
  - Database Design
  - Modeling Methodology
  - Introduction to E-R Diagram
    - Entity
    - Attribute
    - Tuple
    - Chain Notation
    - Relationship degree
    - Relationship Participation
  - Integrity Constraints

Database Design

- To design database following are the steps:
  - Requirement Analysis
  - Diagrammatic representation
  - Translating Diagrams to tables
  - Refining the tables based on fixed set of rules.

Entity relationship diagrams

- An entity-relationship model (ERM) is an abstract conceptual representation of structured data.

- Entity-relationship modeling is a **relational schema database modeling method**, used in software engineering to produce a type of **conceptual data model** (or semantic data model) of a system, often a relational database, and its requirements in a **top-down** fashion.
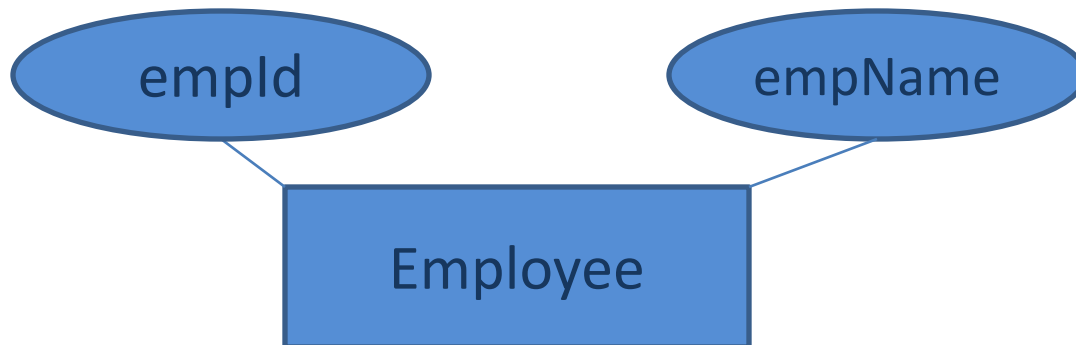
Entity

- In relation to a database , an entity is a single person, place, or thing about which data can be stored.

- In data modeling (a first step in the creation of a database), an entity is some unit of data that can be classified and have stated relationships to other entities.
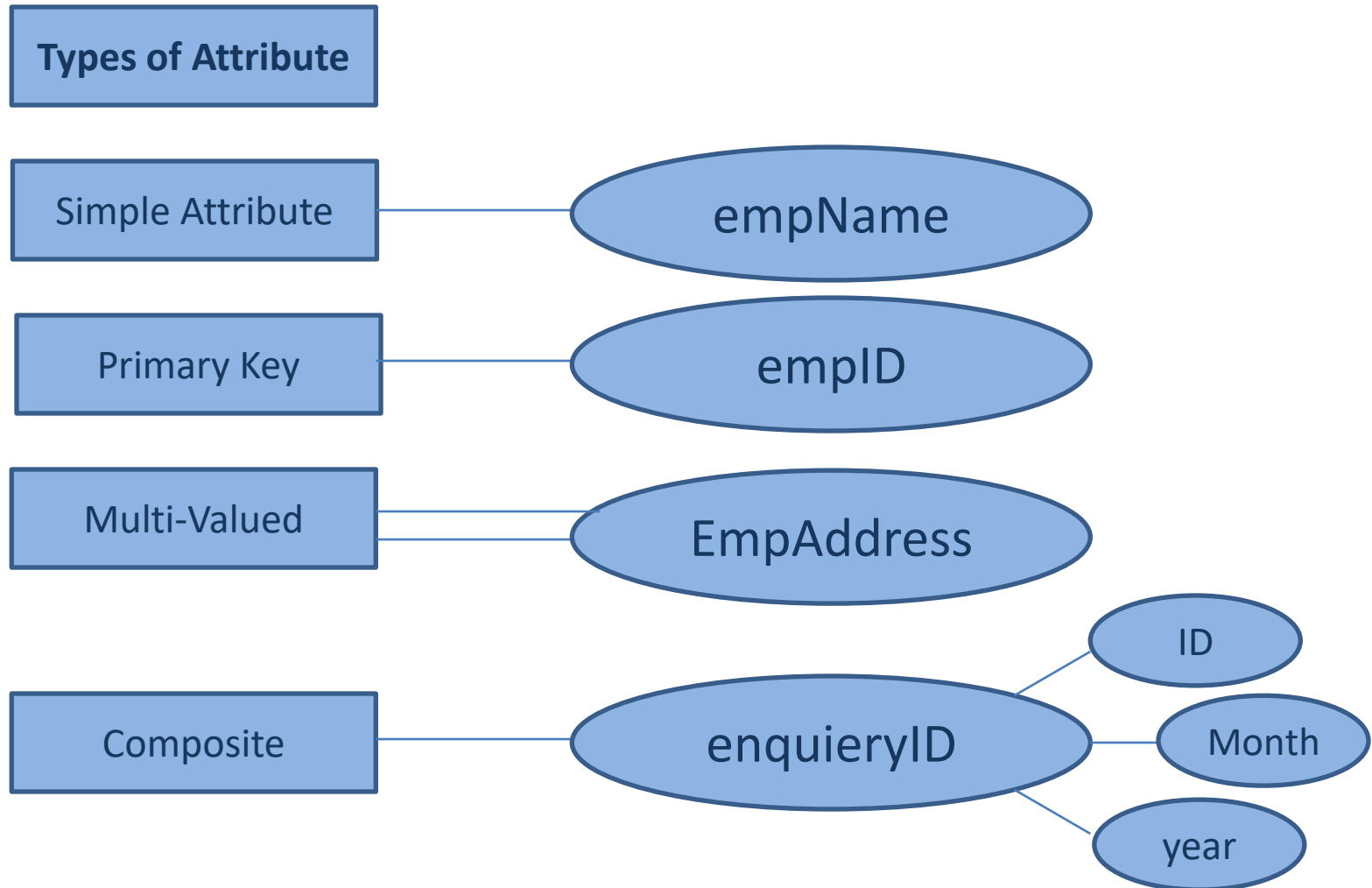
- Example

Employee

Attributes

- **Characteristics/properties** of an entity, that provide descriptive details of it.

- every attribute must be given a **name** that is **unique** across the entity.

- Attributes are represented by **ovals** and are connected to the entity with a line.
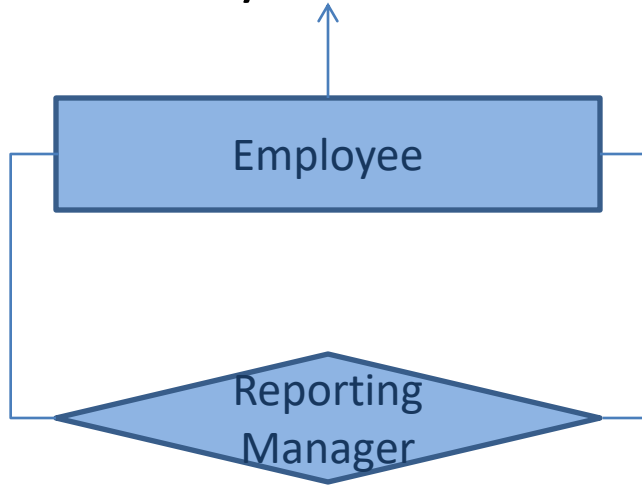

empId   empName

Employee

Chain Notation Attribute

Types of Attribute

Simple Attribute —— empName

Primary Key —— empID

Multi-Valued —— EmpAddress
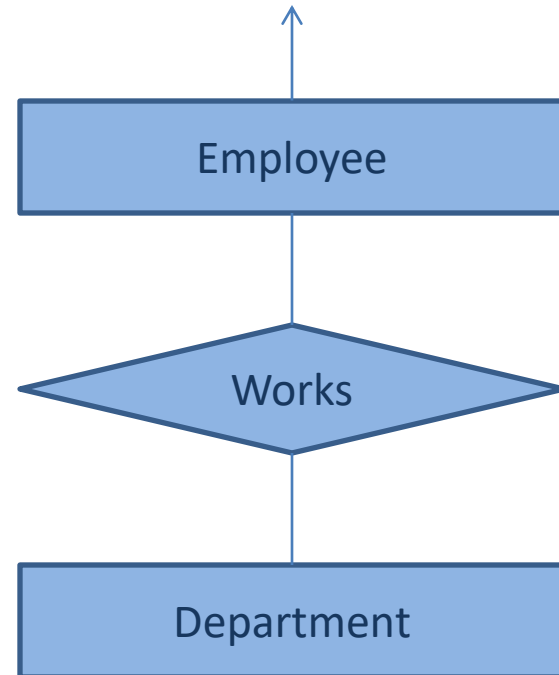
Composite —— enquieryID
- ID
- Month
- year

Relationship Degree

- Relationship degree indicates the number of entities involved in the relationship

Unary Relation

Binary Relation
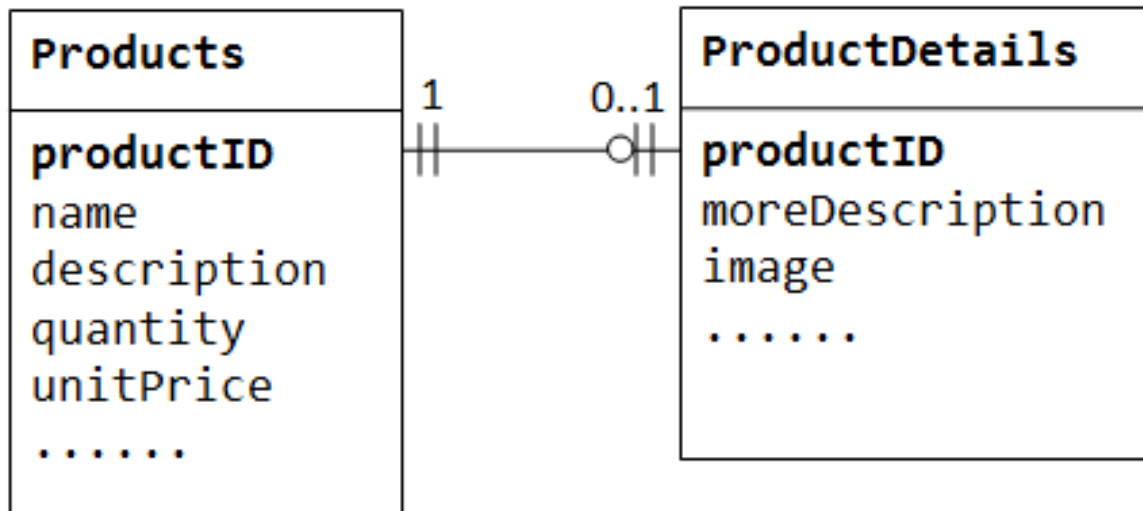

27

Relationship Degree Continue…

Ternary Relation
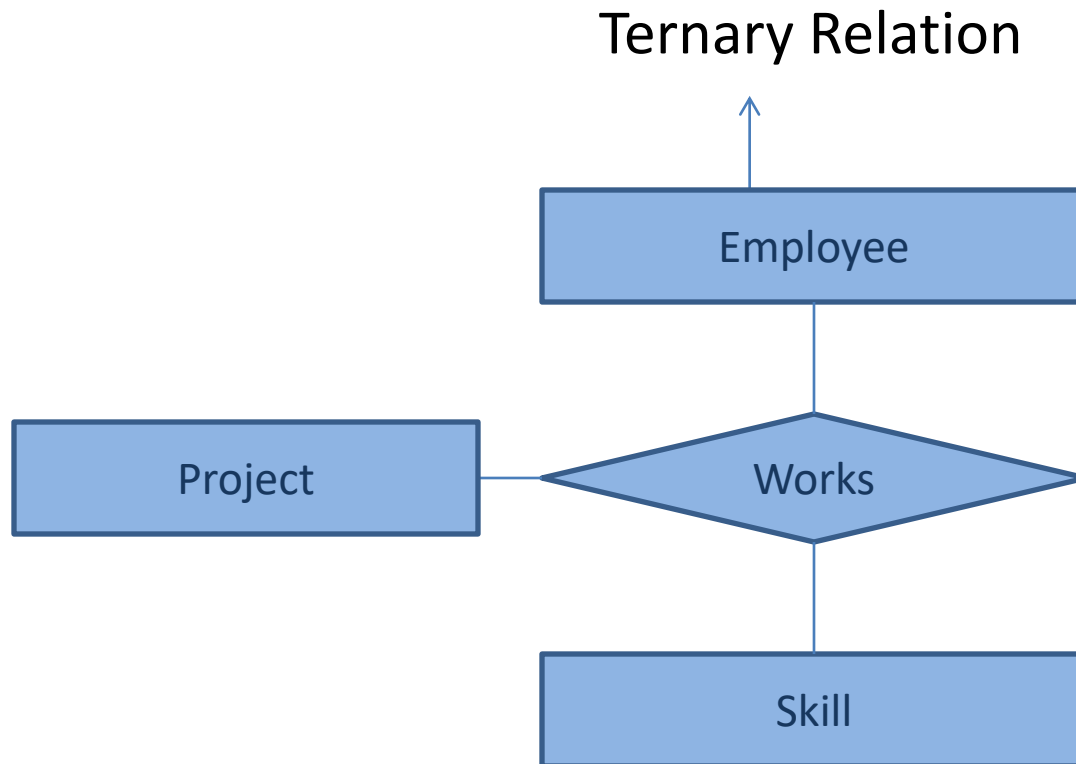
Relationships among Tables

- A database consisting of independent and unrelated tables serves little purpose. The power of relational database lies in the relationship that can be defined between tables.

- The types of relationship include
  - One-to-one
  - One-to-many
  - Many-to-many

One to One

One to Many
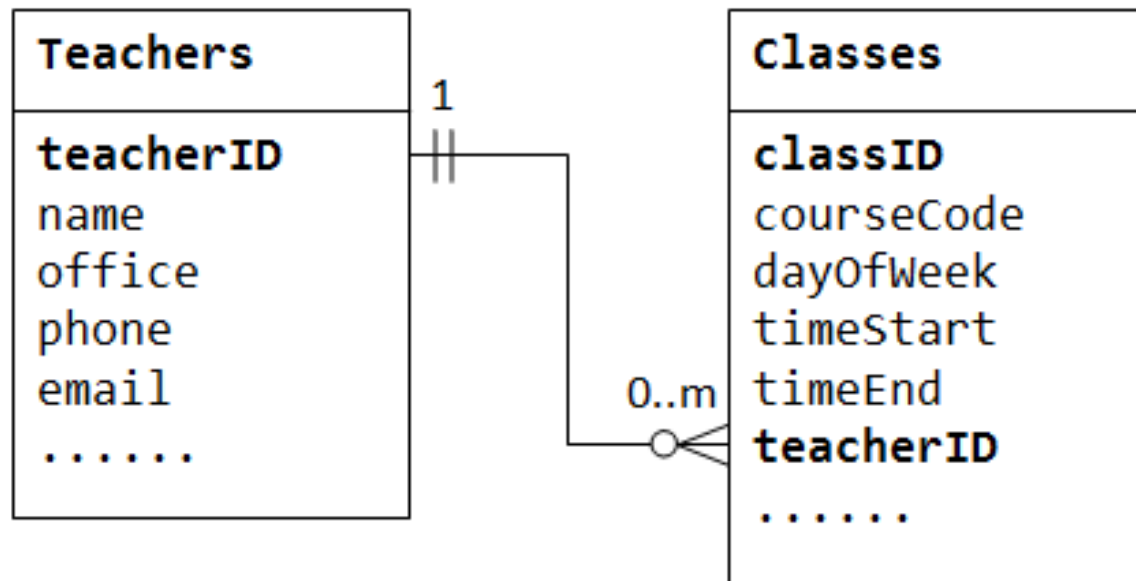
Many to Many
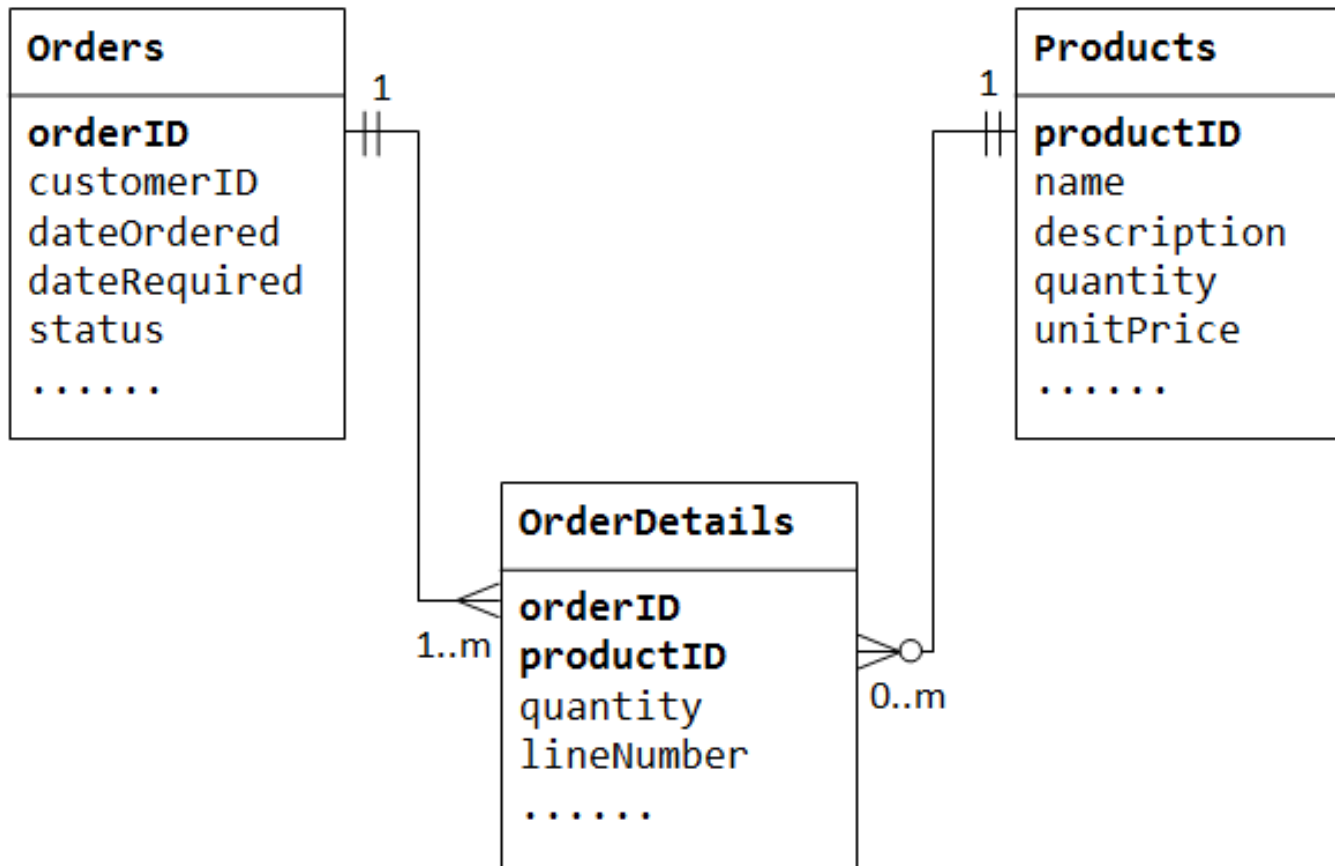
Relationship Participation

- **Fundamental/Strong entity**
  - an entity that is capable of its own existence - i.e. an entity whose instances exist not with standing the existence of other entities.
- **Weak Entities**
  - an entity that is not capable of its own existence.
- **Associative Entities**
  - Associative entity is an entity that is used to resolve a many: many relationship.

E-R Diagram Example
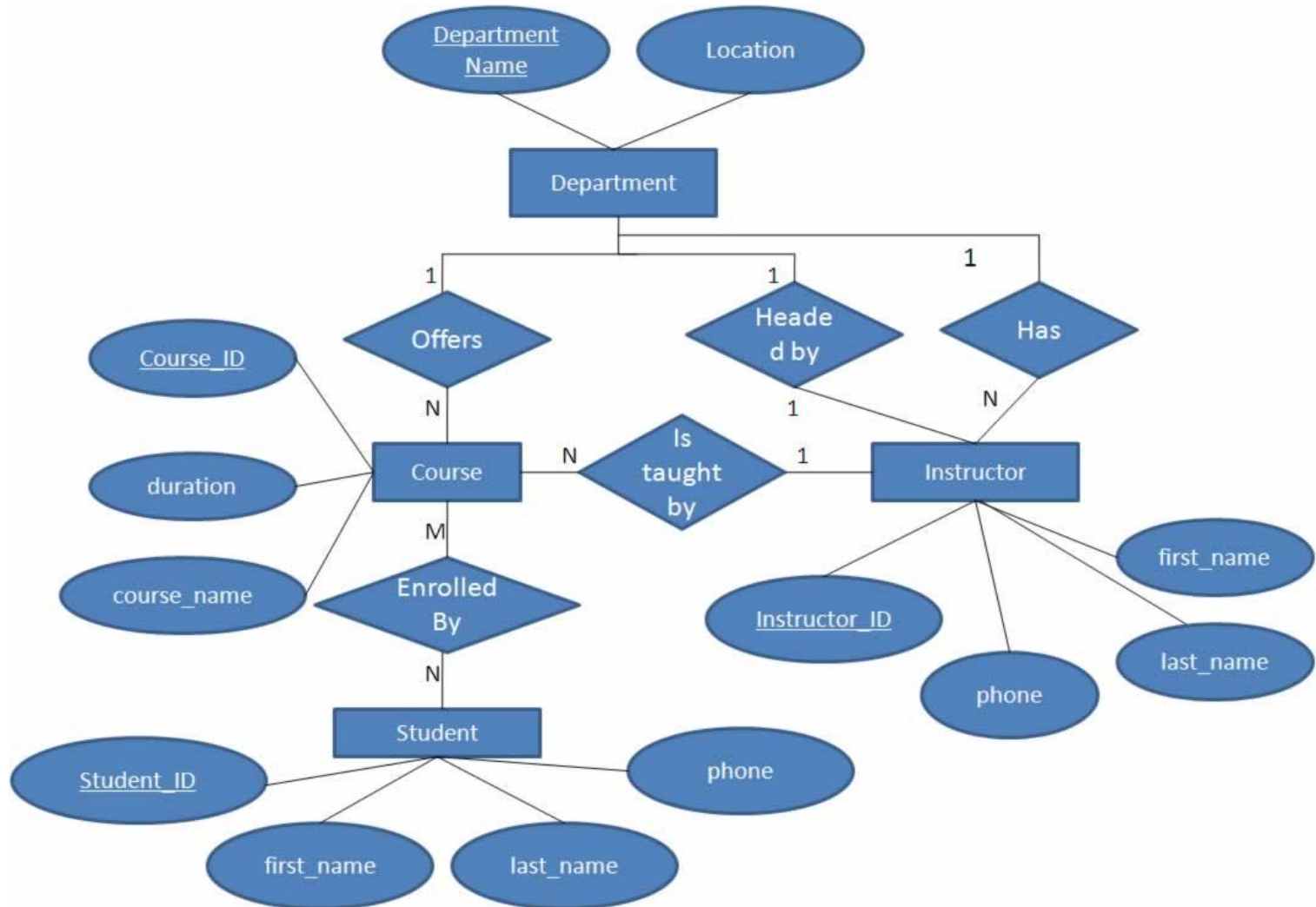
Bank ERD exercise

The person opens an Account in a Bank and gets a account number and ATM card. The person can make transactions in ATM centers.

Bank ERD
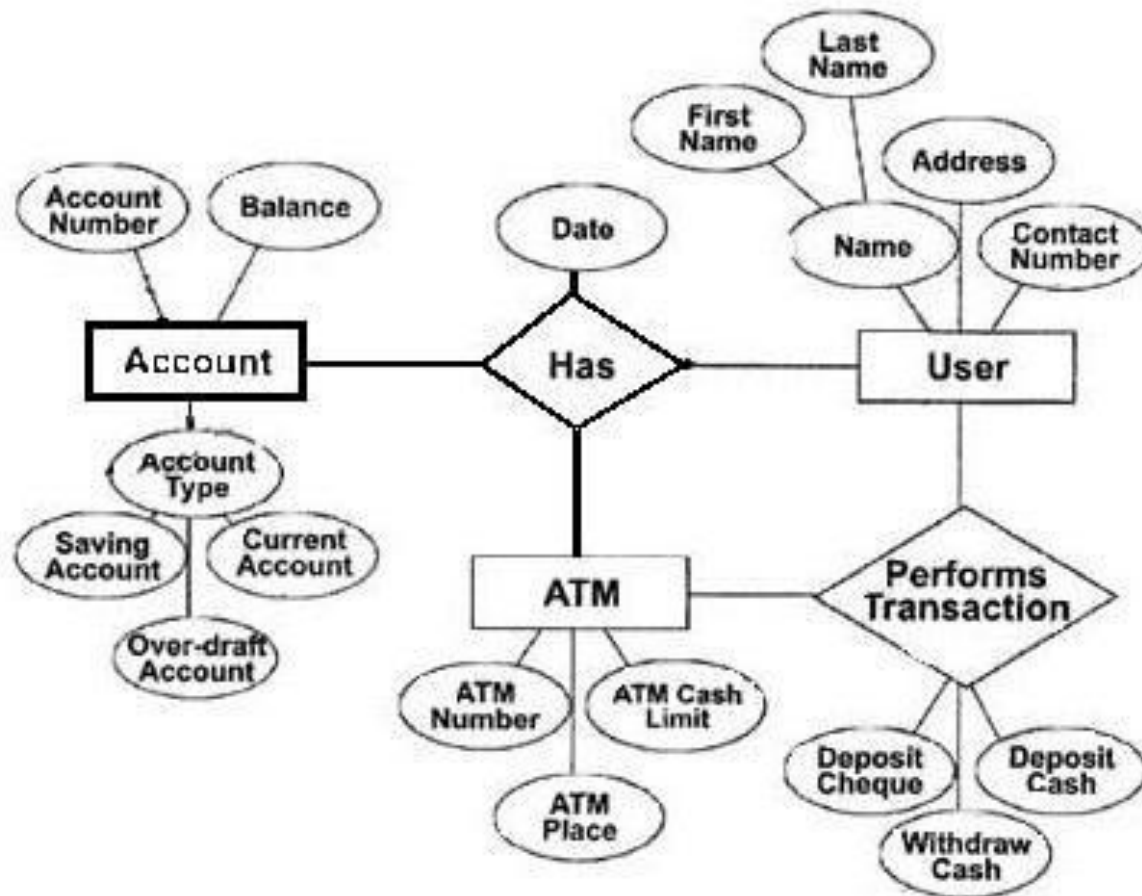
ER Diagram of Banking System

Company ERD exercise

A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.

Note:

1) Supervisor is not an employee.

2) One employee can work on many projects.

Company ERD
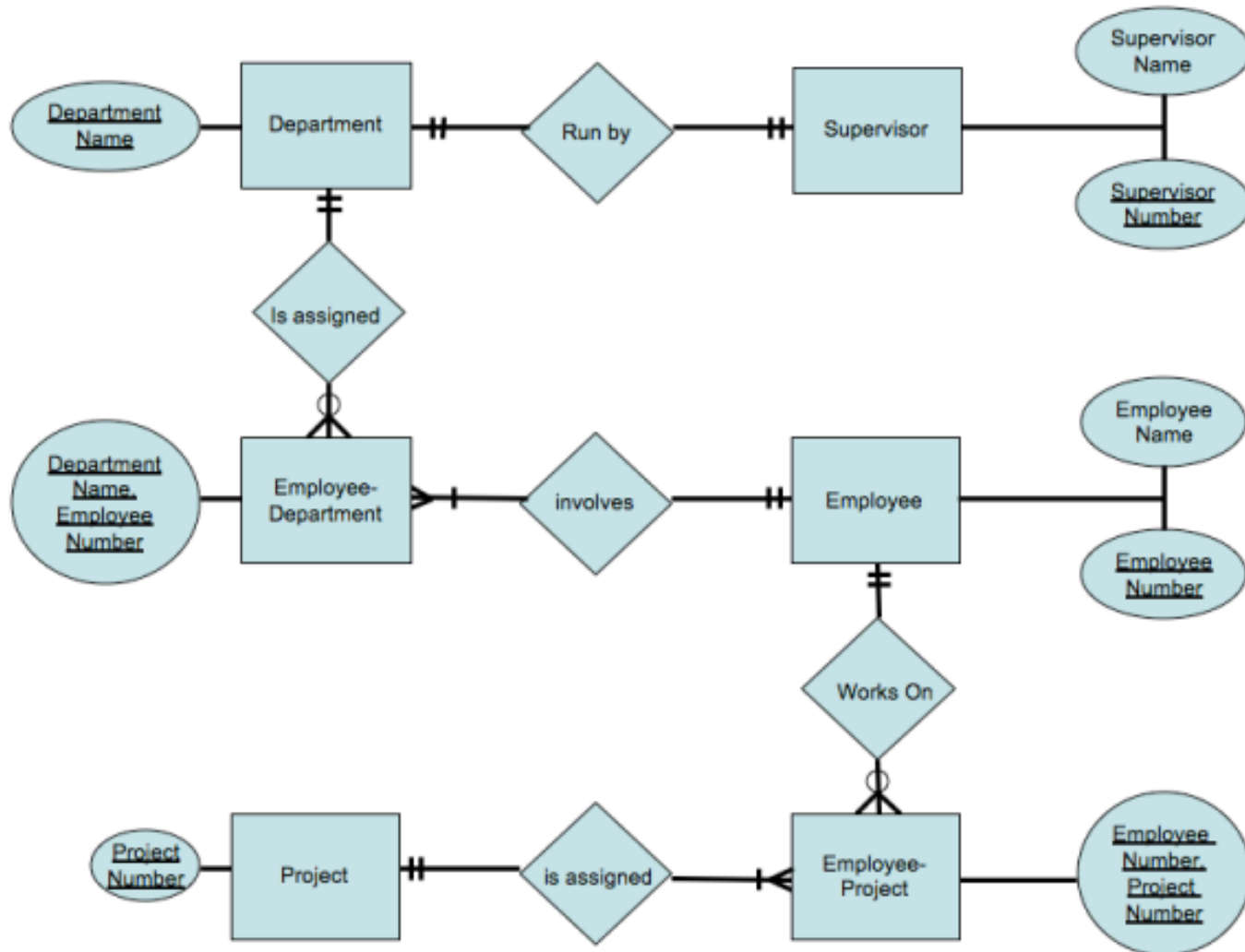
Integrity Constraints

- Integrity means something like 'be right' and consistent. The data in a database must be right and in good condition.
  - Domain Integrity
  - Entity Integrity Constraint
  - Referential Integrity Constraint
  - Foreign Key Integrity Constraint

Domain Integrity

- Domain integrity means the definition of a valid set of values for an attribute. You define
  - data type,
  - length or size
  - is null value allowed
  - is the value unique or not
  - for an attribute.
  - You may also define the default value, the range (values in between) and/or specific values for the attribute.

Integrity Constraints continue…

- **Entity Integrity Constraint**
  – The entity integrity constraint states that primary keys can't be null. There must be a proper value in the primary key field.

– **Referential Integrity Constraint**

  - The referential integrity constraint is specified between two tables and it is used to maintain the consistency among rows between the two tables.

– **Foreign Key Integrity Constraint**

  - There are two foreign key integrity constraints: cascade update related fields and cascade delete related rows.

Module 3: Database Normalization

- Overview
  - Introduction to Normalization
  - Normalization Rule

Introduction to Normalization

- Database Normalization **is a technique** of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion.

Why Normalization?

Database without normalization face the following deviation from standards also called anomalies:

➤ Insert anomaly

➤ Update anomaly

➤

Ta

Id	Product_name	Product_cost	Product_location
1001	Solar Cooker	3000	Pune
1002	Solar AC	25000	Mumbai
1003	Solar Lamp	2000	Kolkata
1004	Solar Cooker	3000	Indore

Why Normalization continued…

➢ **Insert anomaly:**

Insert Anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes. For example we cannot insert new product information into PRODUCT table unless we know the location where we wish to launch it.

➢ **Update anomaly:**

Update Anomaly exists when one or more instances of duplicated data is updated, but not all. Suppose we want to update the price of 'Solar Cooker' in PRODUCT table.

Why Normalization continued…

➢**Delete anomaly:**

Delete Anomaly exists when certain attributes are lost because of the deletion of other attributes. Suppose we want to delete 'Mumbai' manufacturing location then information about 'Solar AC' will also be lost.

Thus, in order to overcome the anomalies, we need database normalization.

Normalization Rule

- Normalization rule are divided into following normal form.
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)

First Normal Form (1NF) continued…

Table will be in 1st normal form if

➢ There are no duplicated rows in the table.

➢ Each cell is single-valued (i.e., there are no repeating groups or arrays).

➢ Entries in a column (attribute, field) are of the same kind.

First Normal Form (1NF) continued…

Consider the following table:

College	Student	Age	Subject
Fergusson	Adam	15	Biology, Maths
MIT	Alex	14	Maths
BMCC	Stuart	17	Maths

First Normal Form (1NF) continued…

Table after 1st Normal Form:

College	Student	Age
Fergusson	Adam	15
MIT	Alex	14
BMCC	Stuart	17

College	Student	Subject
Fergusson	Adam	Biology
Fergusson	Adam	Maths
MIT	Alex	Maths
BMCC	Stuart	Maths

Second Normal Form (2NF)

- Table should be in 1st Normal Form
- As per the 2NF there must not be **any partial dependency of any column on primary key**
- It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

Second Normal Form (2NF) Continue…

Composite Primary Key
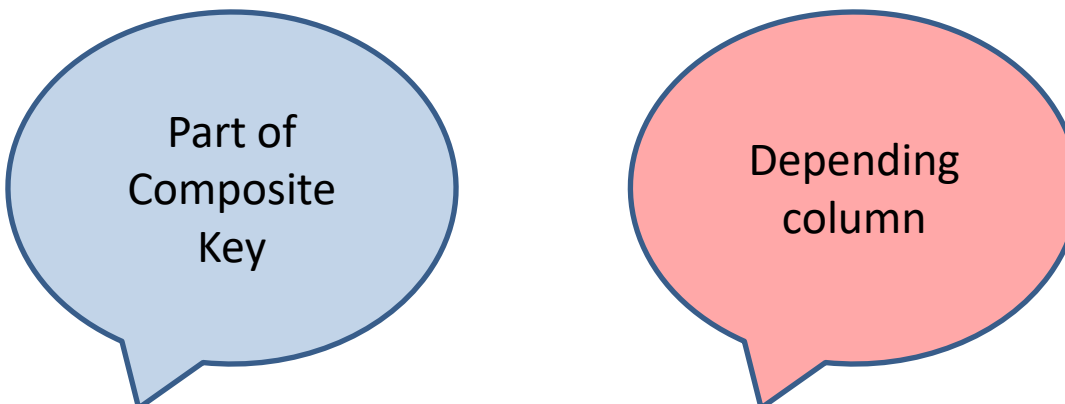
Column should dependent on Primary Key

College	Student	Age
Fergusson	Adam	15
Fergusson	Adam	15
MIT	Alex	14
BMCC	Stuart	17

Student Table in 1st Normal Form

Second Normal Form (2NF) Continue...

Part of Composite Key

Depending column

College	Student	Age
Fergusson	Adam	15
Fergusson	Adam	15
MIT	Alex	14
BMCC	Stuart	17

Student Table in 1st Normal Form

Second Normal Form (2NF) Continue...

Student Table in 1st Normal Form

College	Student	Age
Fergusson	Adam	15
MIT	Alex	14
BMCC	Stuart	17

New Student Table following 2NF will be :

Student	Age
Adam	15
Alex	14
Stuart	17

New Subject Table following 2NF will be :
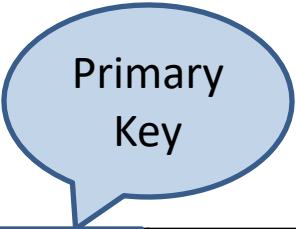
College	Student	Subject
Fergusson	Adam	Biology
Fergusson	Adam	Maths
MIT	Alex	Maths
BMCC	Stuart	Maths

Third Normal Form (3NF)

- **Third Normal form** applies that every non-prime attribute of table must be dependent on primary key. The *transitive functional dependency* should be removed from the table. The table must be in **Second Normal form**. For example, consider a table with following fields.

Example

- Student_Detail Table

Primary Key

Std_id	Std_name	DOB	Street	city	State	Zip
1088	Rahul	01-Jan-1987	G.G. Road	Thane	MH	400601
1092	Kiran	08-Aug-1989	Karve Road	Dombivali	MH	4210202
2010	Amit	19-Sep-1984	G.G. Road	Thane	MH	400601
2211	Geeta	01-Feb-2000	Karve Road	Dombivali	MH	4210202

Example

- Student_Detail Table

street, city and state depends upon Zip

Transitive Dependency

Std_id	Std_name	DOB	Street	city	State	Zip
1088	Rahul	01-Jan-1987	G.G. Road	Thane	MH	400601
1092	Kiran	08-Aug-1989	Karve Road	Dombivali	MH	4210202
2010	Amit	19-Sep-1984	G.G. Road	Thane	MH	400601
2211	Geeta	01-Feb-2000	Kare Road	Dombivali	MH	4210202

Example

▸ New Student_Detail Table :

Student_id	Student_name	DOB	Zip
1088	Rahul	01-Jan-1987	400601
1092	Kiran	08-Aug-1989	4210202
2010	Amit	19-Sep-1984	400601
2211	Geeta	01-Feb-2000	4210202

• Address Table :

Zip	Street	city	State
400601	G.G. Road	Thane	MH
4210202	Karve Road	Dombivali	MH

The advantage of removing transitive dependency is,

▸ Amount of data duplication is reduced.

▸ Data integrity achieved.

Case study

Project Management Report

Project Code: PC010   **Project Manager:** M Philips
Project Title: Pensions System   **Project Budget:** $24500

Employee No	Employee Name	Department No.	Department Name	Hourly Rate
S10001	A Smith	L004	IT	$22
S10030	L Jones	L023	Pensions	$18
S21010	P Lewis	L004	IT	$21
S00232	R Smith	L003	Programming	$26

Total Staff: 4   **Average Hourly Rate: $21.88**

Case study continued…

Table in unnormalized form (UNF)

Project Code	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50
PC045	Salaries System	H Martin	17400	S13210	W Richards	L008	Salary	17.00
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Database	23.25
PC064	HR System	K Lewis	12250	S21010	P Lewis	L004	IT	17.50
PC064	HR System	K Lewis	12250	S10034	B James	L009	HR	16.50

Case study continued…

Table after 1st Normal Form (1NF)   Repeating Attributes Removed

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

Project Code	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	S10001	A Smith	L004	IT	22.00
PC010	S10030	L Jones	L023	Pensions	18.50
PC010	S21010	P Lewis	L004	IT	21.00
PC045	S10010	B Jones	L004	IT	21.75
PC045	S10001	A Smith	L004	IT	18.00
PC045	S31002	T Gilbert	L028	Database	25.50
PC045	S13210	W Richards	L008	Salary	17.00
PC064	S31002	T Gilbert	L028	Database	23.25
PC064	S21010	P Lewis	L004	IT	17.50
PC064	S10034	B James	L009	HR	16.50

Case study continued…

Table after 2nd Normal Form (2NF) Partial Key Dependencies Removed

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

Project Code	Employee No.	Hourly Rate
PC010	S10001	22.00
PC010	S10030	18.50
PC010	S21010	21.00
PC045	S10010	21.75
PC045	S10001	18.00
PC045	S31002	25.50
PC045	S13210	17.00
PC064	S31002	23.25
PC064	S21010	17.50
PC064	S10034	16.50

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

Case study continued…

Table after 3rd Normal Form (3NF) Removed transitive dependency

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

Project Code	Employee No.	Hourly Rate
PC010	S10001	22.00
PC010	S10030	18.50
PC010	S21010	21.00
PC045	S10010	21.75
PC045	S10001	18.00
PC045	S31002	25.50
PC045	S13210	17.00
064	S31002	23.25
PC064	S21010	17.50
PC064	S10034	16.50

Department No.	Department Name
L004	IT
L023	Pensions
L028	Database
L008	Salary
L009	HR

Employee No.	Employee Name	Department No. *
S10001	A Smith	L004
S10030	L Jones	L023
S21010	P Lewis	L004
S10010	B Jones	L004
S31002	T Gilbert	L023
S13210	W Richards	L008
S10034	B James	L0009

Module 4: Getting Started with MySQL

- Overview
  - Introduction to Databases
  - Introducing SQL

Introduction to Databases

Computerized record-keeping system

EMPNO	ENAME	JOB	MANAGER	HIREDATE	SALARY	COMMISSION	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-1980	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-1981	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-1981	1250	500	30
7566	JONES	MANAGER	7839	02-APR-1981	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-1981	2850		30
7782	CLARK	MANAGER	7839	09-JUN-1981	2450	0	10
7788	SCOTT	ANALYST	7566	19-APR-1987	3000		20

Introducing SQL

SQL statement entered

Select * from Emp

Database

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10
7788	SCOTT	ANALYST	7566	19-APR-87	3000	(null)	20
7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100	(null)	20

Module 5: Data Retrieval & Ordering Output

- **Overview**
  - Simple Data Retrieval
  - Describing Table Structure
  - Conditional Retrieval using Arithmetic, Relational, Logical   and Special Operators
  - The ORDER BY clause.
  - Aggregate functions
  - The GROUP BY and HAVING clause

Describing table

DESC dept

Name	Null	Type
DEPTNO	NOT NULL	INT(2)
DNAME		VARCHAR(14)
LOC		VARCHAR(13)
BUDGET		INT

Data Retrieval

SELECT * FROM emp;

SELECT empno, ename FROM emp;

SELECT distinct deptno FROM emp;

Conditional Retrieval

SELECT * FROM emp WHERE sal > 3500;

SELECT   empno, ename FROM emp WHERE deptno = 20;

Relational Operators

- =   equal to
- !=   not equal to
- ^=   not equal to
- <>   not equal to
- >   greater than
- <   less than
- >=   greater than or equal to
- <=   less than or equal to

SELECT * FROM emp WHERE sal> 3000;

SELECT ename FROM emp WHERE deptno**!=** 10;

SELECT * FROM employee WHERE ename **=** 'ALLEN';

Logical Operators

- **The AND Operator**

  SELECT * FROM emp WHERE deptno = 10 AND job = 'SALESMAN';

  SELECT * FROM emp WHERE sal >= 3000 AND sal <= 4000;

- **The OR Operator**

  SELECT * FROM emp WHERE deptno = 20 OR deptno = 10;

- **The NOT Operator**

  SELECT * FROM employee WHERE NOT deptno = 10;

Range & List Operators

- **The BETWEEN operator**

  SELECT * FROM emp WHERE sal BETWEEN 3000 and 4000;

  SELECT * FROM emp WHERE hiredate BETWEEN '01-JAN-80' and '31-DEC-89'

- **The IN operator**

  – SELECT * FROM emp WHERE deptno IN (10,20);

  – SELECT * FROM emp WHERE deptno NOT IN (10,20);

Wildcard & is Null Operators

- The LIKE operator

  SELECT * FROM emp WHERE ename LIKE 'J%';

  SELECT * FROM emp WHERE ename LIKE '%AD';

  SELECT * FROM emp WHERE ename LIKE '%AD%';

  SELECT * FROM emp WHERE grade LIKE 'A_____';

- The IS NULL operator

  SELECT * FROM emp WHERE comm IS NULL;

  SELECT * FROM emp WHERE comm IS NOT NULL;

Arithmetic Operators

- + addition
- - subtraction
- * multiplication
- / division

SELECT * FROM emp WHERE sal + com > 3000 and comm is not null;

SELECT ename, sal + comm FROM emp WHERE comm is not null;

SELECT ename, sal+ comm "Total Earning" FROM emp WHERE comm is not null;

Sorting Output

- **Ordering on single column**

  SELECT * FROM emp ORDER BY empno;

  SELECT * FROM emp WHERE deptno= 10 ORDER BY ename;

  SELECT * FROM emp ORDER BY sal DESC;

- **Ordering on multiple columns**

  SELECT * FROM emp ORDER BY depno, ename;

  SELECT * FROM emp ORDER BY deptno, job DESC;

Aggregate Functions

- SELECT COUNT(*) FROM emp;
- SELECT SUM(salary) FROM emp;
- SELECT AVG(salary) FROM emp;
- SELECT MAX(salary) FROM emp;
- SELECT MIN(salary) FROM emp;
- SELECT * FROM emp WHERE salary = (SELECT MIN(salary) FROM emp);

The GROUP BY clause

SELECT deptno, sum (sal)

FROM emp

GROUP BY deptno;

The HAVING Clause

SELECT dept_no, sum(salary) FROM emp

GROUP BY dept_no

HAVING sum(salary) > 7000

SELECT dept_no, sum(salary)

FROM emp

WHERE dept_no in(10, 30)

GROUP BY dept_no

HAVING sum(salary) > 8000

Module 6: Creating Tables

- Overview
  - Creating a Table
  - Data Types

Creating Tables

CREATE TABLE tablename (
        column-name data-type [other clauses]...);

CREATE TABLE dept (
        dept_no varchar(4),
        dname varchar(20),
        loc varchar(20)
);

Numeric Data Types

Data Type	Description	Example
INT	A normal-sized integer that can be signed or unsigned. You can specify a width of up to 11 digits.	dept_no INT(4)
TINYINT	A very small integer that can be signed or unsigned. You can specify a width of up to 4 digits.	dept_no TINYINT(4)
SMALLINT	A small integer that can be signed or unsigned. You can specify a width of up to 5 digits.	dept_no SMALLINT(4)
MEDIUMINT	A medium-sized integer that can be signed or unsigned. You can specify a width of up to 9 digits.	dept_no MEDIUMINT(4)
BIGINT	A large integer that can be signed or unsigned. You can specify a width of up to 20 digits.	dept_no BIGINT(4)

Numeric Data Types continue...

Data Type	Description	Example
FLOAT	A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). Decimal precision can go to 24 places for a FLOAT.	salary FLOAT(7, 2)
DOUBLE	A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). Decimal precision can go to 53 places for a DOUBLE.	salary DOUBLE(7, 2)
DECIMAL	An unpacked floating-point number that cannot be unsigned.	salary DECIMAL(7, 2)

Date & Time Data Types

Data Type	Description	Example
DATE	A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31.	hired_date DATE
DATETIME	A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59.	hired_date DATETIME
TIMESTAMP	A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; For example 19731230153000 (YYYYMMDDHHMMSS)	hired_date TIMESTAMP
TIME	Stores the time in HH:MM:SS format.	meeting_at TIME

String Data Types

Data Type	Description	Example
CHAR	A fixed-length string between 1 and 255 characters in length.	gender CHAR(1)
VARCHAR	A variable-length string between 1 and 255 characters in length.	username VARCHAR(15)
BLOB or TEXT	Stores large amount of data. The difference between BLOB & TEXT is sorting and comparing on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields.	emp_photo BLOB
ENUM	An enumeration, to store possible values of a column.	genders ENUM('Male', 'Female')

Module 7: Inserting, Modifying & Deleting Data

- Overview
  - Inserting Data into a Table
  - Inserting Data into a Table using Sub query
  - Modifying Data in a Table
  - Deleting Data from a Table

Inserting Data into a Table

desc dept;

Name   Null?   Type

------------   --------------------   ---------------
--

DEPTNO   NOT NULL   VARCHAR(4)

DNAME   NOT NULL   VARCHAR(20)

LOC   NOT NULL   VARCHAR(20)

INSERT INTO table-name VALUES (value1, value2, ...);

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');

INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');

INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');

INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

Customized Insertion

INSERT INTO emp (empno, salary, ename)
VALUES(1051, 5000, 'Sunil');

INSERT INTO RESEARCH_EMP

SELECT * FROM emp WHERE dept_no=20;

Create table RESEARCH_EMP

As

SELECT * FROM emp WHERE dept_no=20;

Modifying and Deleting Data

- UPDATE emp SET salary = salary + 100;
- UPDATE emp SET salary = salary + 200 WHERE dept_no= 10;

- DELETE FROM emp;
- DELETE FROM emp WHERE deptno = 30;

Module 8: Modifying Table Structure

- Overview
  - Altering Table structure
  - Dropping Column from a Table
  - Dropping a Table

Modifying a Table Structure

ALTER table emp

ADD (age int(2));

ALTER table emp

MODIFY age int(3);

ALTER table emp DROP column age;

ALTER table emp DROP commission, DROP age;

Dropping a Table

DROP TABLE table-name;

- DROP table dept;

Module 9: Integrity Constraints

▸ Overview
  ▸ Understanding Table and Column Constraints
  ▸ Creating, Modifying and Dropping Column level constraints
  ▸ Creating, Modifying and Dropping Table level constraints
  ▸ Adding Constraints to Columns of an existing table
  ▸ Enabling and Disabling Constraints
  ▸ Dropping Columns and Tables having constraints
  ▸ Creating, modifying & Dropping Sequence

Integrity Constraints

- Not Null
- Unique
- Primary Key
- Check
- Foreign Key

Column Constraints

CREATE TABLE employee (

empno int(5) **NOT NULL**,

ename varchar (25) **NOT NULL**,

deptno varchar(4));

UNIQUE Constraints

CREATE TABLE emp(

       empno int(5),

       ename varchar(25) NOT NULL

       CONSTRAINT emp_eq UNIQUE(empno));

SELECT constraint_name FROM information_schema.TABLE_CONSTRAINTS   WHERE table_name = 'EMP';

Primary key Constraint

CREATE TABLE supplier (

       supp_code int(4), supp_name varchar(30),

 CONSTRAINT code_pk PRIMARY KEY(supp_code),

 CONSTRAINT name_uq UNIQUE(supp_name)

);

The REFERENCES Constraint

CREATE TABLE emp (

    employeeID int(5) primary key,

    name varchar(80),

    job varchar(30),

    departmentID int not null ,

    CONSTRAINT fk_department FOREIGN KEY (departmentID) references dept(ID)

)

The REFERENCES Constraint

CREATE TABLE emp (

    employeeID int(5) primary key,

    name varchar(80),

    job varchar(30),

    departmentID int not null ,

    CONSTRAINT fk_department FOREIGN KEY (departmentID) references dept(ID)

    ON DELETE CASCADE

);

Adding Constraints to Columns of an existing Table

ALTER TABLE dept
ADD
CONSTRAINT cd_pk PRIMARY KEY (deptno);

ALTER TABLE emp
ADD
      CONSTRAINT cd_fk
      FOREIGN KEY(dept_no) REFERENCES dept (deptno);

Dropping a Constraint

ALTER TABLE emp

DROP FOREIGN KEY fk_department;

Module 10: Built-In Functions

- Overview
  - Numeric functions
  - Control functions
  - Character functions
  - Date functions
  - Special formats with Date data types
  - Conversion functions

Functions on Numeric data types

Function	Returns	Example	Result
CEIL (n)	Raises a number containing decimals to the highest whole number	SELECT CEIL(4.84)	5
FLOOR(n)	This reduces any number containing decimals to the lowest whole number.	SELECT FLOOR(4.84)	4
ROUND(number, decimal places)	It rounds the figures up or down to the nearest whole number (or to a specified number of decimal places).	SELECT ROUND(4.88, 1)	4.9
TRUNCATE(number, places)	This function, rather than rounding, simply shortens the number to a required decimal place.	SELECT TRUNCATE(4.88, 1)	4.8

Functions on Numeric data types continue...

Function	Returns	Example	Result
COUNT(field)	This counts the number of times a row (or field) is returned.	select COUNT(*) from Employee	12
AVG(field)	Returns the average of different field values.	select AVG(salary) from Employee	25000
MIN(field)	Returns minimum value of a field	select MIN(salary) from Employee	12000
MAX(field)	Returns maximum value of a field	select MAX(salary) from Employee	87000
SUM(field)	Returns sum of all values of a field	select SUM(salary) from Employee	970000

Control functions

Function	Returns	Example	Result
IF(condition, true value, false value)	Returns true value if condition fulfils; otherwise false value.	SELECT IF(74 > 40, 'PASS', 'FAILED')	PASS
CASE()	It is similar to switch case	SELECT CASE 2 WHEN 1 THEN 'One' WHEN 2 THEN 'Two' WHEN 3 THEN 'Three' END;	Two
IFNULL(original value, new value)	Returns new value if original value is null; otherwise returns original value itself	SELECT IFNULL(NULL, 'The value is NULL')	This value is NULL

Character data type functions

Function	Returns	Example	Result
CONCAT(str1, str2,…)	It is used to concatenate multiple strings. It returns concatenated string.	select CONCAT('AB', 'CD', 'EF')	ABCDEF
REPLACE(whole_string, to_be_replaced, replacement)	It is used to replace a specific portion of a string into new one.	select REPLACE('I like to talk', 'talk', 'walk')	I like to walk
INSERT(string,start_position,length,newstring)	It overwrites any text in the string from a start point for a certain length.	select INSERT('I can talk',7,0,'walk and ')	I can walk and talk
LEFT(string, length)	Extracts the specific portion of a string from left.	select LEFT('I am a human being', 4)	I am
RIGHT(string, length)	Extracts the specific portion of a string from right.	select RIGHT('I am a human being', 5)	being
MID(string, start position, length)	Extracts specific portion of a string within another string.	select MID('I am a human being', 3, 10)	am a human

Character data type functions continue...

Function	Returns	Example	Result
SUBSTRING(string, position)	Returns subset of a string.	select SUBSTRING('I am a human being', 8)	human being
LOCATE(substring, string)	It finds a substring within a string	select LOCATE('human', 'I am a human being')	8
LENGTH(string)	Returns the length of a string.	select LENGTH('MySQL')	5
UCASE(string)	Returns uppercase string	select UCASE('MySql')	MYSQL
LCASE(string)	Returns lowercase string	select LCASE('MySql')	mysql
REVERSE(string)	Returns the reversed string	select REVERSE('MySql')	lqSyM

Date data type functions

Function	Returns	Example	Result
NOW()	Returns the current date & time.	select NOW()	2017-02-12 17:21:40
CURTIME()	Returns the current time.	select CURTIME()	17:21:40
CURDATE()	Returns the current date.	select CURDATE()	2017-02-12
DATE_ADD(date,INTERVAL expr type)	Adds time to a date.	DATE_ADD(now(), INTERVAL 5 DAY)	2017-03-08 12:12:54
DATE_SUB(date,INTERVAL expr type)	Subtracts time from a date.	DATE_SUB(now(), INTERVAL 5 DAY)	2017-02-26 12:12:54

Date data type functions continue…

Function	Returns	Example	Result
DATE_FORMAT()	Allows to format any date in customized fashion	select date_format(sysdate(), '%d-%m-%Y')	02-03-2017
DAYOFMONTH(date)	The numeric day of the month	DAYOFMONTH(sysdate())	3
DAYNAME(date)	The Name of the day	DAYNAME(sysdate())	Friday
MONTH(date)	The numeric month	MONTH(sysdate())	02
MONTHNAME(date)	The Month name	MONTHNAME(sysdate())	February

Date data type functions continue…

Function	Returns	Example	Result
YEAR(date)	Four digit year	YEAR(sysdate())	2017
HOUR(time	Hour (24 hour clock)	HOUR(sysdate())	20
MINUTE(time)	Minutes	MINUTE(sysdate())	34
SECOND(time)	Seconds	SECOND(sysdate())	55
DAYOFYEAR(date)	Numeric day of the year	DAYOFYEAR(sysdate())	265
STR_TO_DATE(string date, date format)	Converts string into date based upon supplied date format.	SELECT STR_TO_DATE('2013-02-11', '%Y-%m-%d');	2013-02-11

MySQL function reference

https://dev.mysql.com/doc/refman/5.7/en/functions.html

Module 11: Joins & Sub Queries

- Overview
  - Introduction to Join
  - Types of Joins
  - Introduction to Sub Queries

SQL Joins

- SQL Joins are used to **relate information in different tables**. A Join condition is a part of the sql query that **retrieves rows from two or more tables**.

- A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.

The Syntax for joining two tables is:

SELECT col1, col2, col3...
FROM table_name1 JOIN table_name2
ON table_name1.col2 = table_name2.col1;

Types of Joins

- SQL Inner Join

- SQL Outer Join

  - Left

  - Right
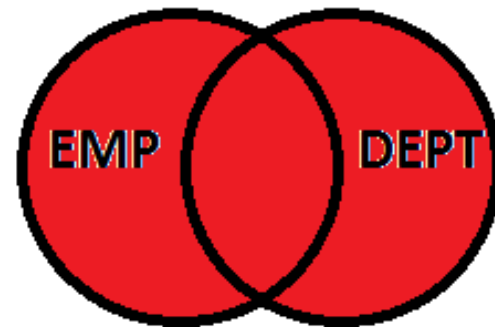
  - full

- ## SQL Self Join

Types of Joins continue...
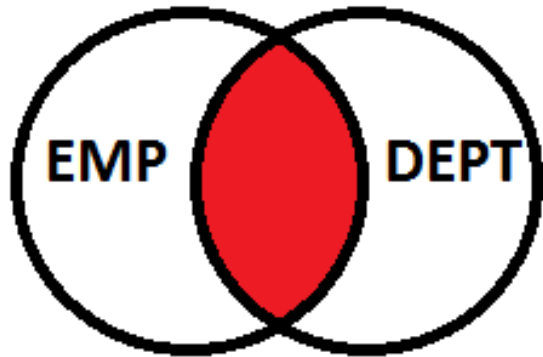
Inner Join
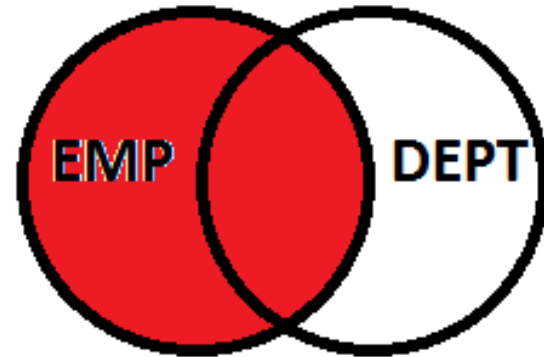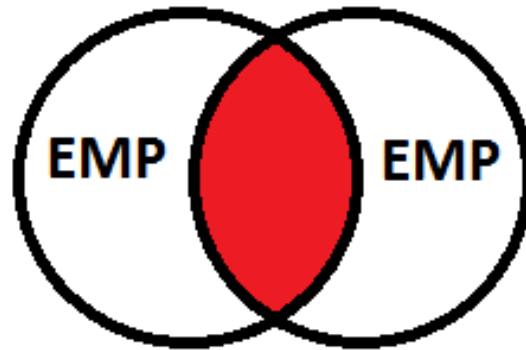
Left Outer Join
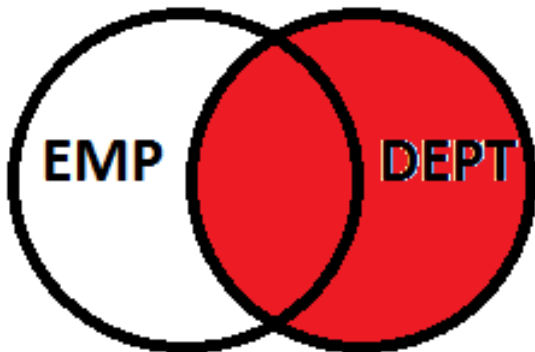
Right Outer Join

Full Join

Types of Joins continue…

Self Join

SQL Inner Join (Equi Join)

Show the employees with their department
name who are associated with any department.

SELECT ename, dname

     FROM EMP JOIN DEPT

     ON dept.dept_code = emp.dept_code;

Outer Join

Right Outer Join: Show the employees with their department name who are associated with any department along with departments with no employees associated.

SELECT a.empno, a.deptno, b.dname

   FROM emp a RIGHT OUTER JOIN dept b

   ON (a.deptno=b.deptno);

Left Outer Join: Show the employees with their department name whether or not they are associated with any department.

SELECT a.empno, a.deptno, b.dname

   FROM emp a LEFT OUTER JOIN dept b

   ON (a.deptno=b.deptno);

Outer Join continue…

Full Outer Join: Show the employees with their department name irrespective whether employees associated with departments or departments associated with employees.

SELECT a.id, a.dept_no, b.name

     FROM emp a LEFT OUTER JOIN dept b

     ON (a.dept_no=b.id)

 UNION

SELECT a.id, a.dept_no, b.name

     FROM emp a RIGHT OUTER JOIN dept b

     ON (a.dept_no=b.id)

Self Join

Show the employees with their manager's name.

SELECT employee.ename, manager.ename

     FROM emp employee join emp manager

     on emloyee.mgr = manager.empno;

SUBQUERIES

➢ Subquery is a query within a query.

➢ Subquery can be nested inside SELECT, INSERT, UPDATE, or DELETE statements.

➢ Subqueries must be enclosed within parentheses.

➢ There are three types of subqueries:

1. Single Row Sub Query

2. Multiple Row Sub Query

3. Correlated Sub Query

Subquery Types

➢ **Single Row Sub Query**

Single row subquery always returns a single row with single column only.

SELECT order_name, order_price FROM Orders where order_price = (SELECT MAX(order_price) FROM Orders)

➢ **Multiple Row Sub Query**

Multiple row sub query returns more than one row. Hence it is generally handled using IN comparison operator.

SELECT order_name order_price FROM Orders where item_id IN (SELECT itemId FROM Items where item_price > 1000)

Subquery Types

➢ Correlated Sub Query

In correlated subquery the inner query depends on values provided by the outer query.

SELECT EMPLOYEE_ID, salary, department_id

FROM   EMP E

WHERE salary > (SELECT AVG(salary)

       FROM   EMP T

       WHERE E.department_id = T.department_id)

SUBQUERIES

SELECT * FROM orders
      WHERE cust_code **IN** (
      SELECT cust_code FROM customer
      WHERE city_code = 'PUNE');

SELECT * FROM dept
      WHERE **EXISTS** (
      SELECT * FROM emp
      WHERE emp.deptno = dept.deptno);

SELECT * FROM dept
      WHERE **NOT EXISTS** (
      SELECT * FROM employee
      WHERE employee.dept_code = dept.dept_code);