Introduction to RDBMS & Oracle SQL

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 Oracle
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.
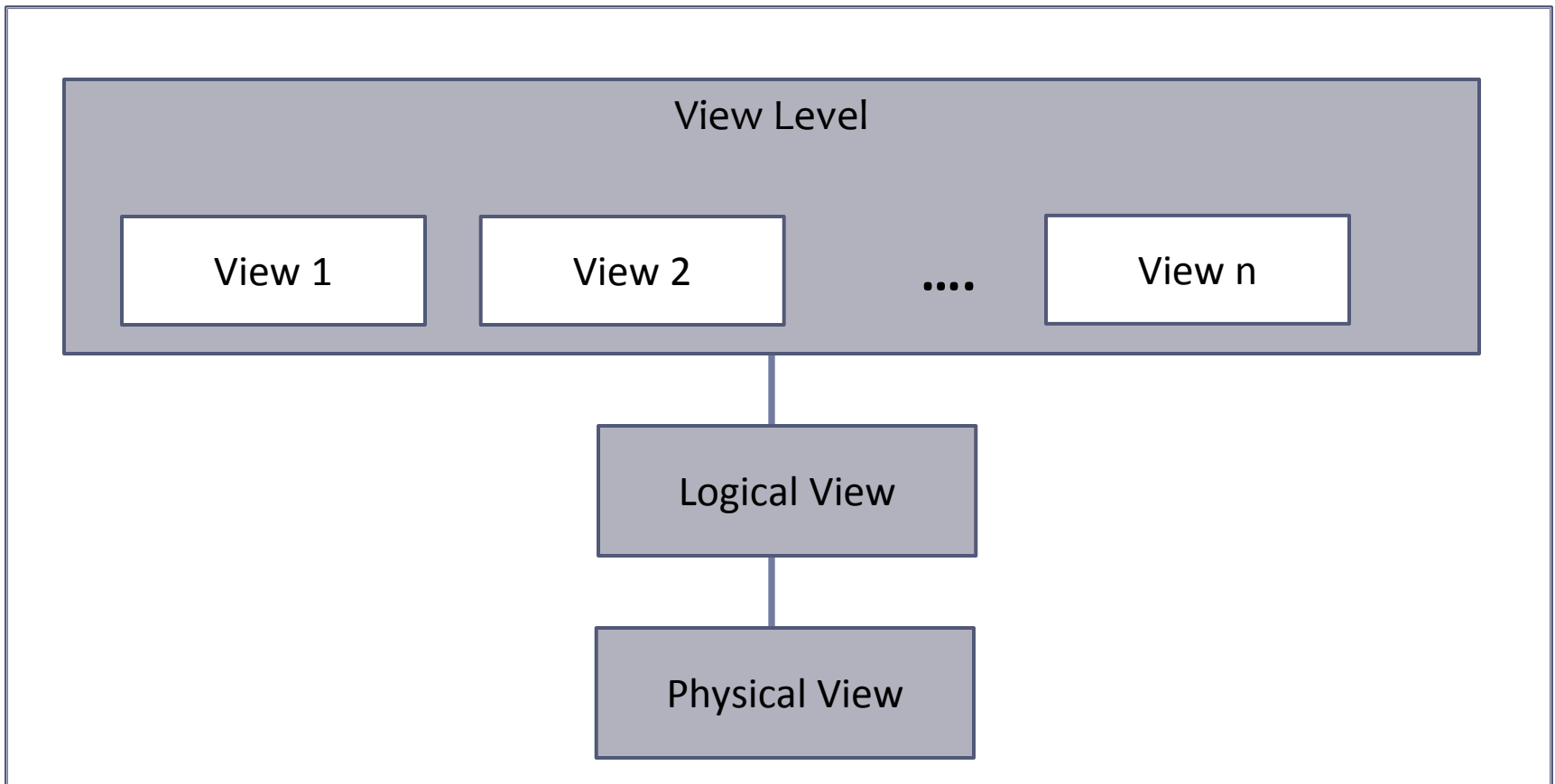
  type *customer* = **record**

      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

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.

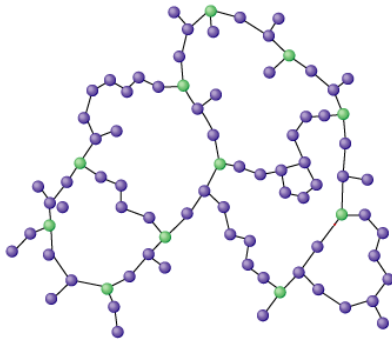- The purpose of a data model is to represent data and to make the data understandable.

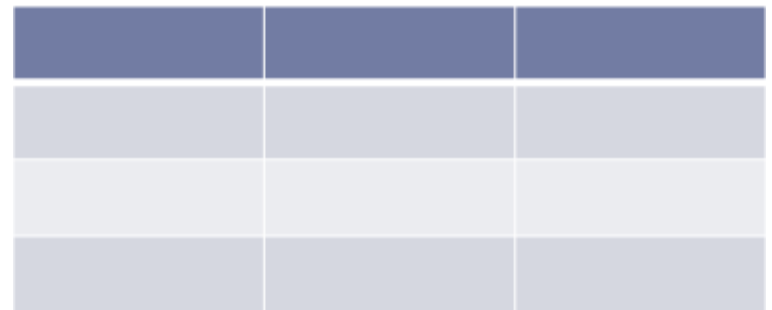# Data Models Continue…

▸ Data Models are categorized into:

Hierarchical Model →
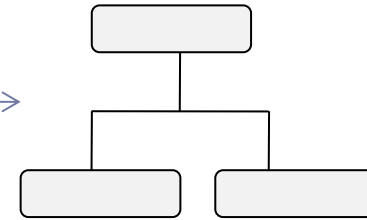
Network Model ←

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
				Attribute
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

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 association 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.

Data Modeling

▸ **Data modeling** in software engineering is the process of creating a **data model** for an information system by applying formal data modeling techniques.

▸ The goal of the data model is to make sure that the all data objects required by the database are completely and accurately represented.

Modeling Methodology

▶ Data models represent information areas of interest. While there are many ways to create data models:

 ▶ Bottom-up models or View Integration models

  ▶ The methodology is often the result of reengineering effort.

 ▶ Top-down logical data models

  ▶ This methodology on other hand, are created in an abstract way by getting information from people.

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.
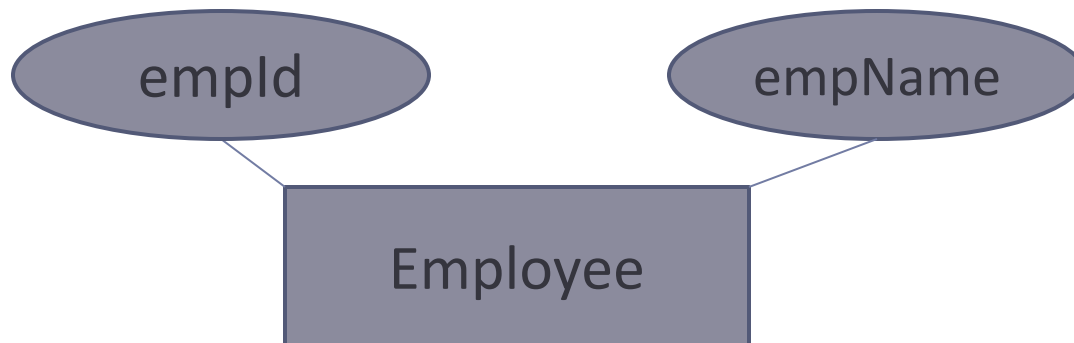
Entity

▸ 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.
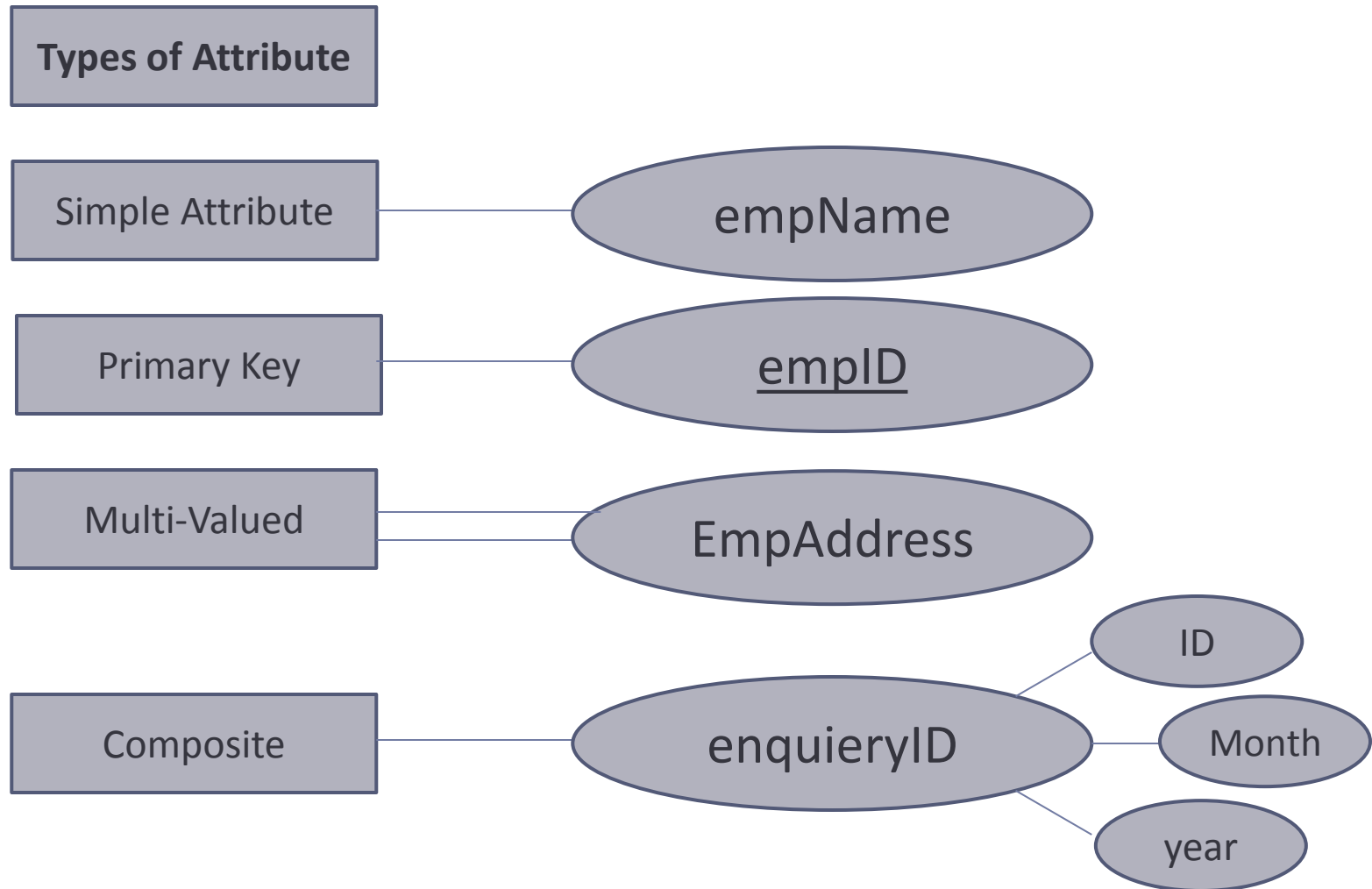
▸ Example

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 | <u>empID</u> |

| Multi-Valued | EmpAddress |

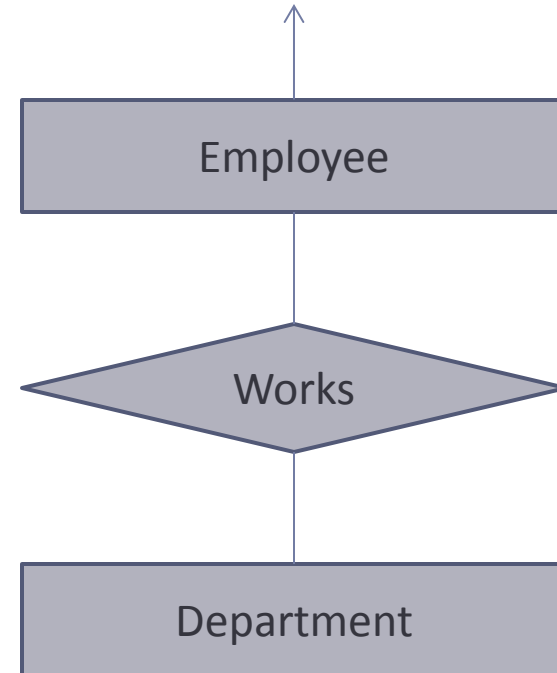| Composite | enquieryID — ID, Month, year |

Relationship Degree

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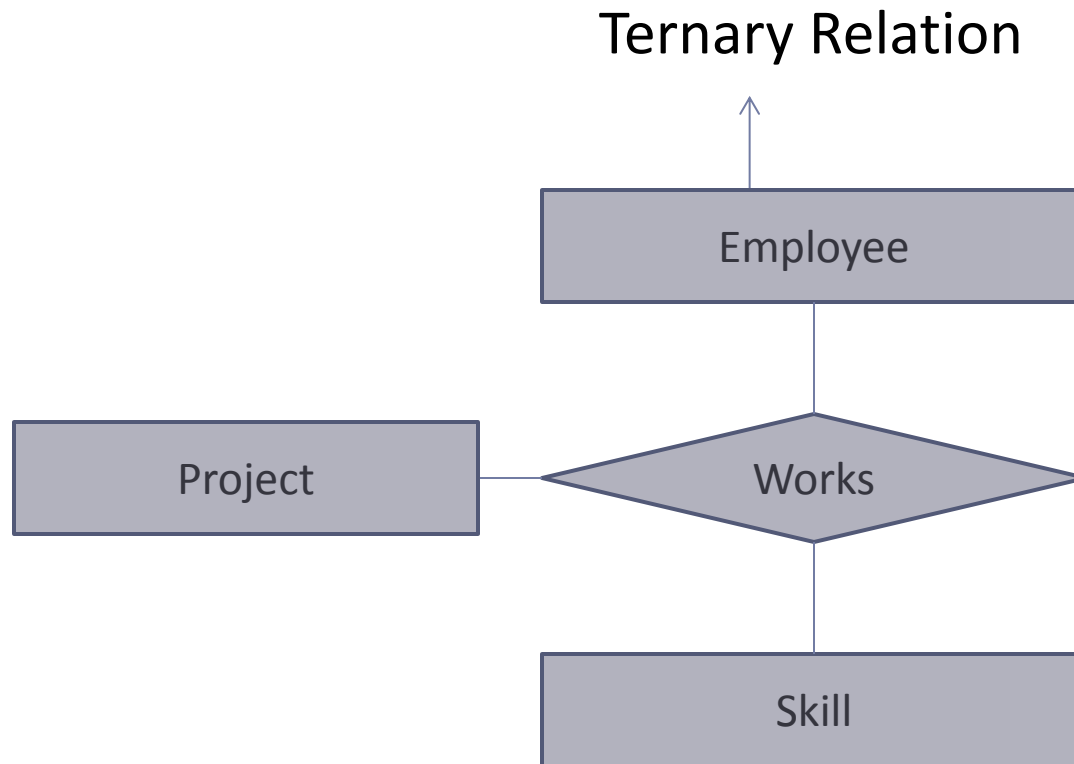

Unary Relation
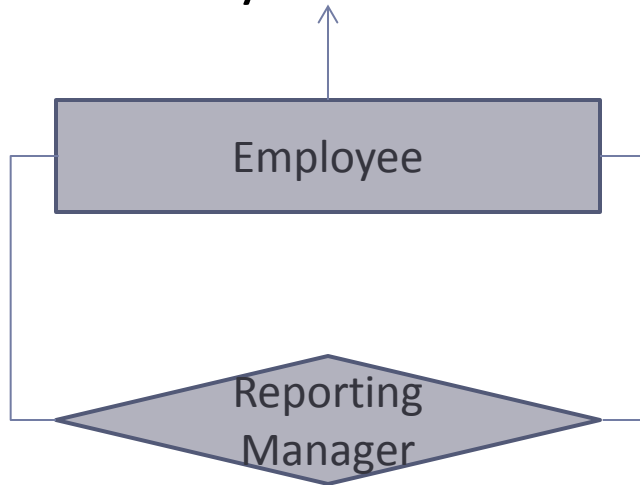
Binary Relation

Relationship Degree Continue…

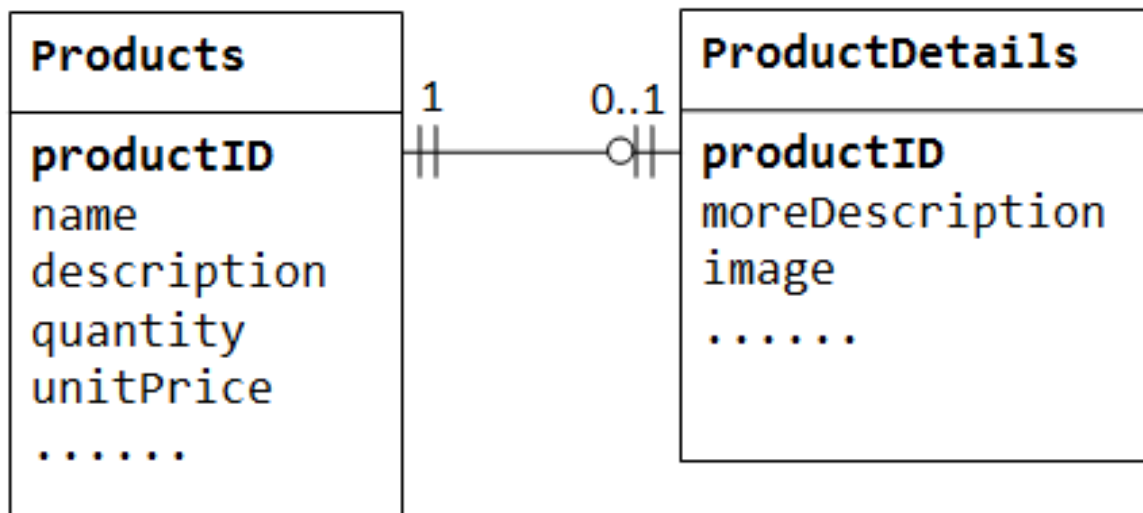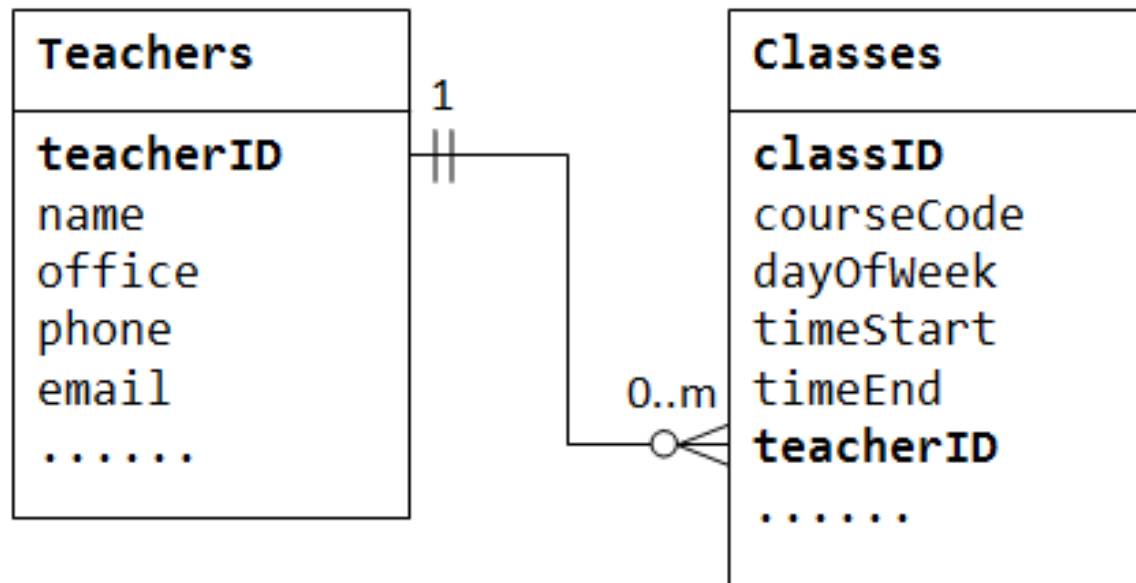Ternary Relation

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
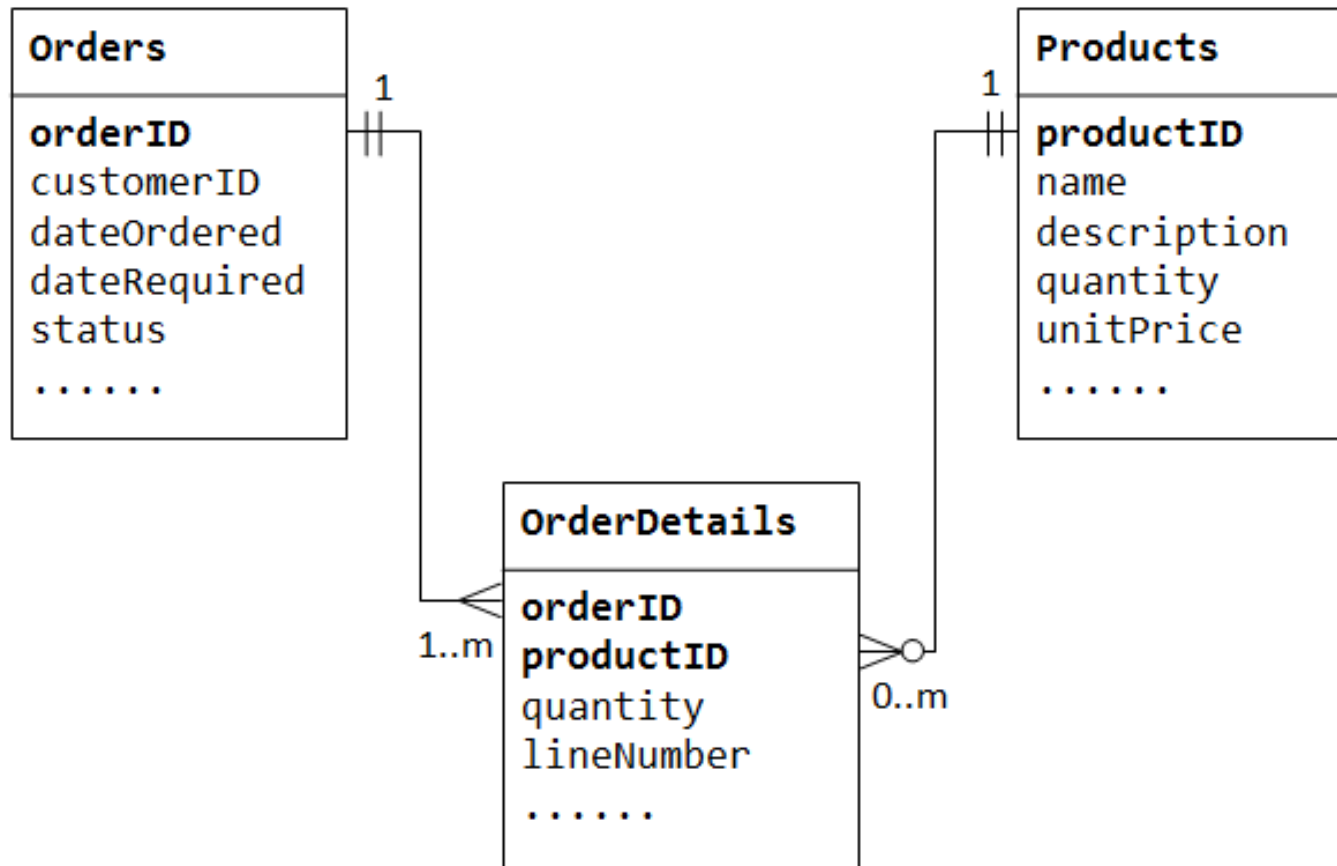
Relationship Participation

▶ **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
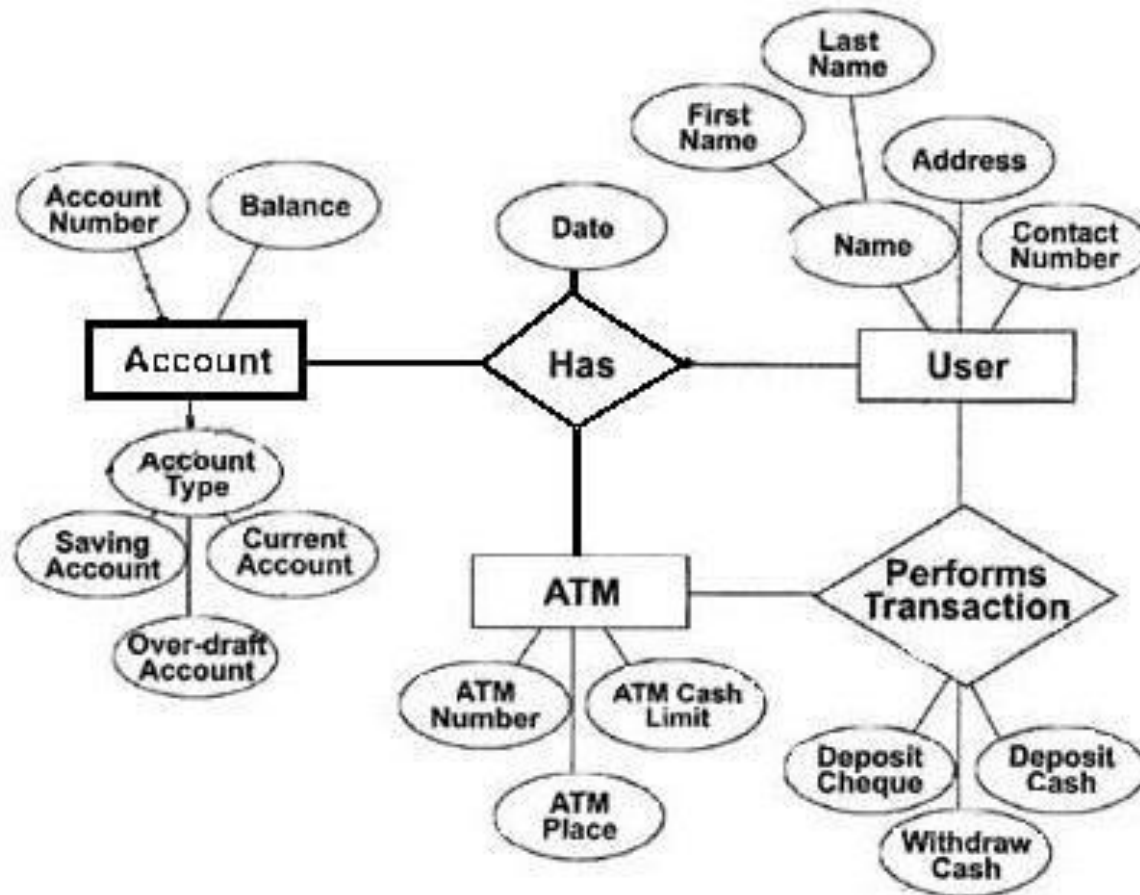
ER Diagram of Banking System

Company ERD exercise

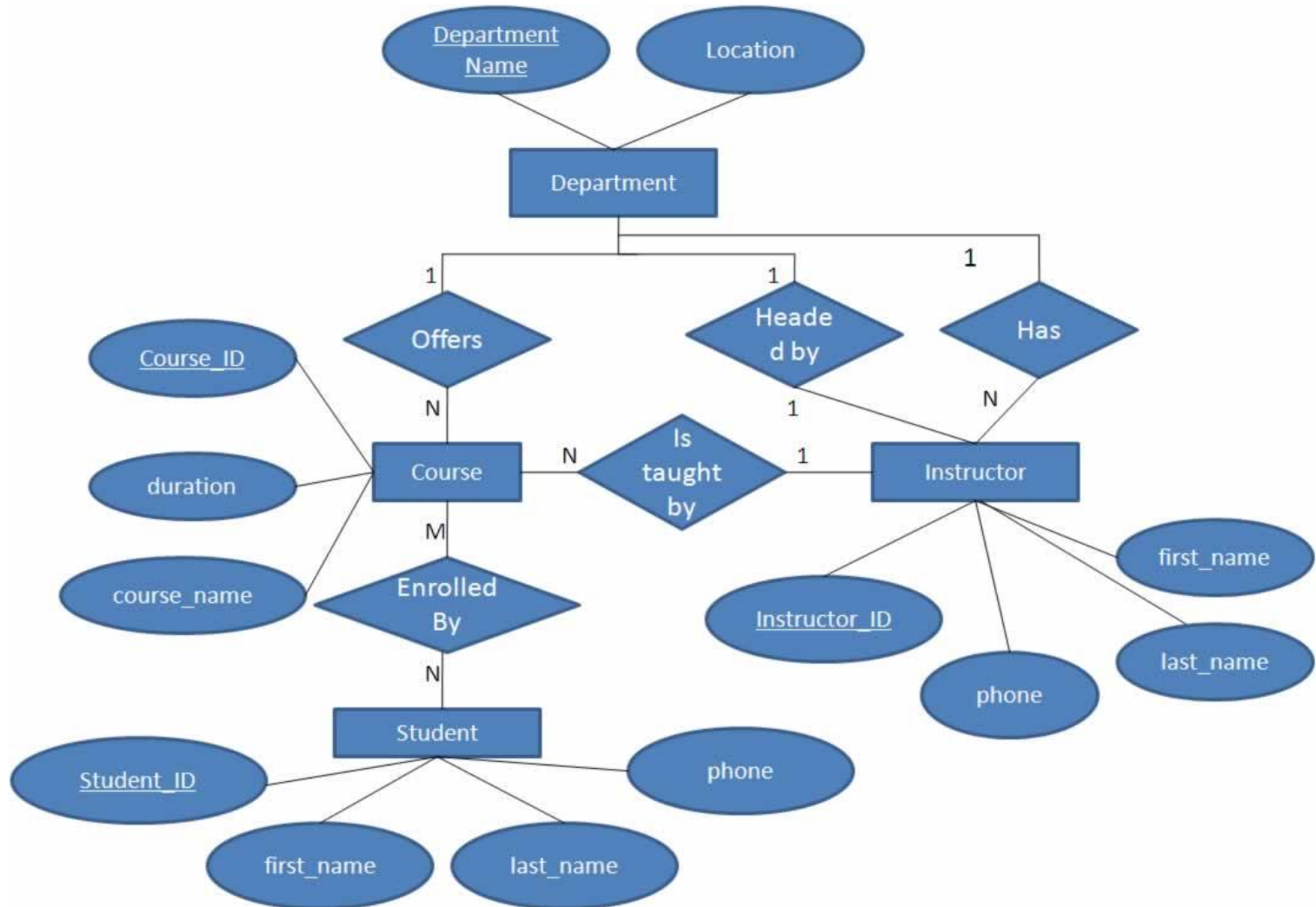A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important 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
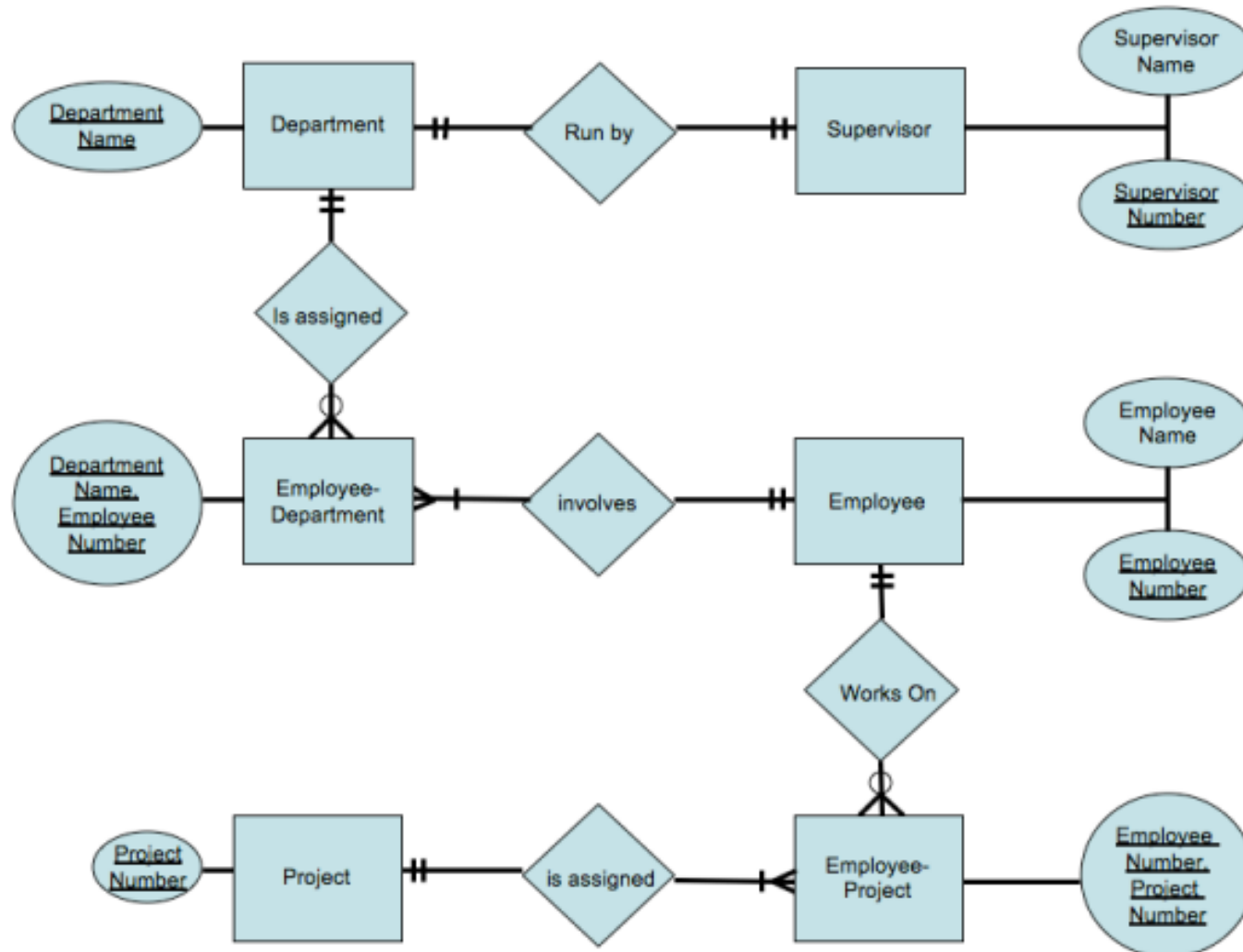
Integrity Constraints

▶ 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

➤ Delete anomaly

Take an example of following PRODUCT table:

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

College	Student	Age	Subject
Fergusson	Adam	15	Biology
MIT	Alex	14	Maths
BMCC	Stuart	17	Maths
Fergusson	Adam	15	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…
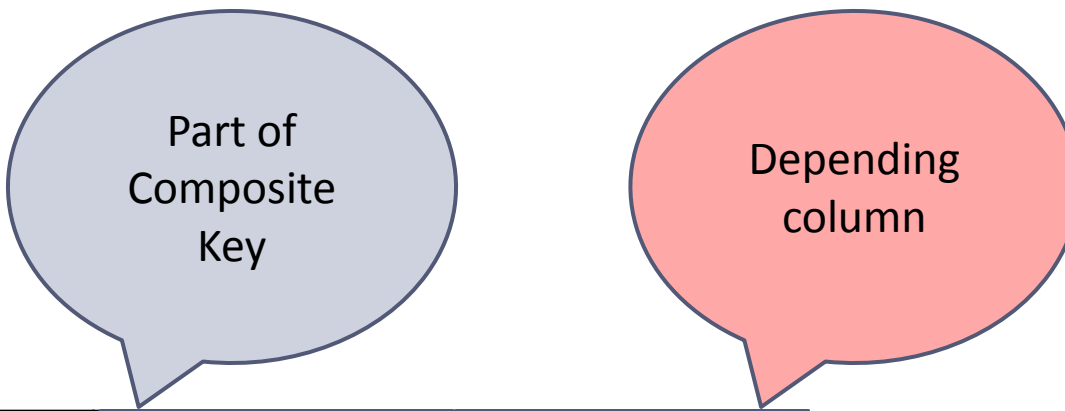
Composite Primary Key

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

▶ Xoriant Soultions Pvt. Ltd.

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

Transitive Dependency

street, city and state depends upon Zip

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 Oracle

▶ Overview

  ▶ Introduction to Databases

  ▶ Introducing SQL

  ▶ SQL Developer with Oracle

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
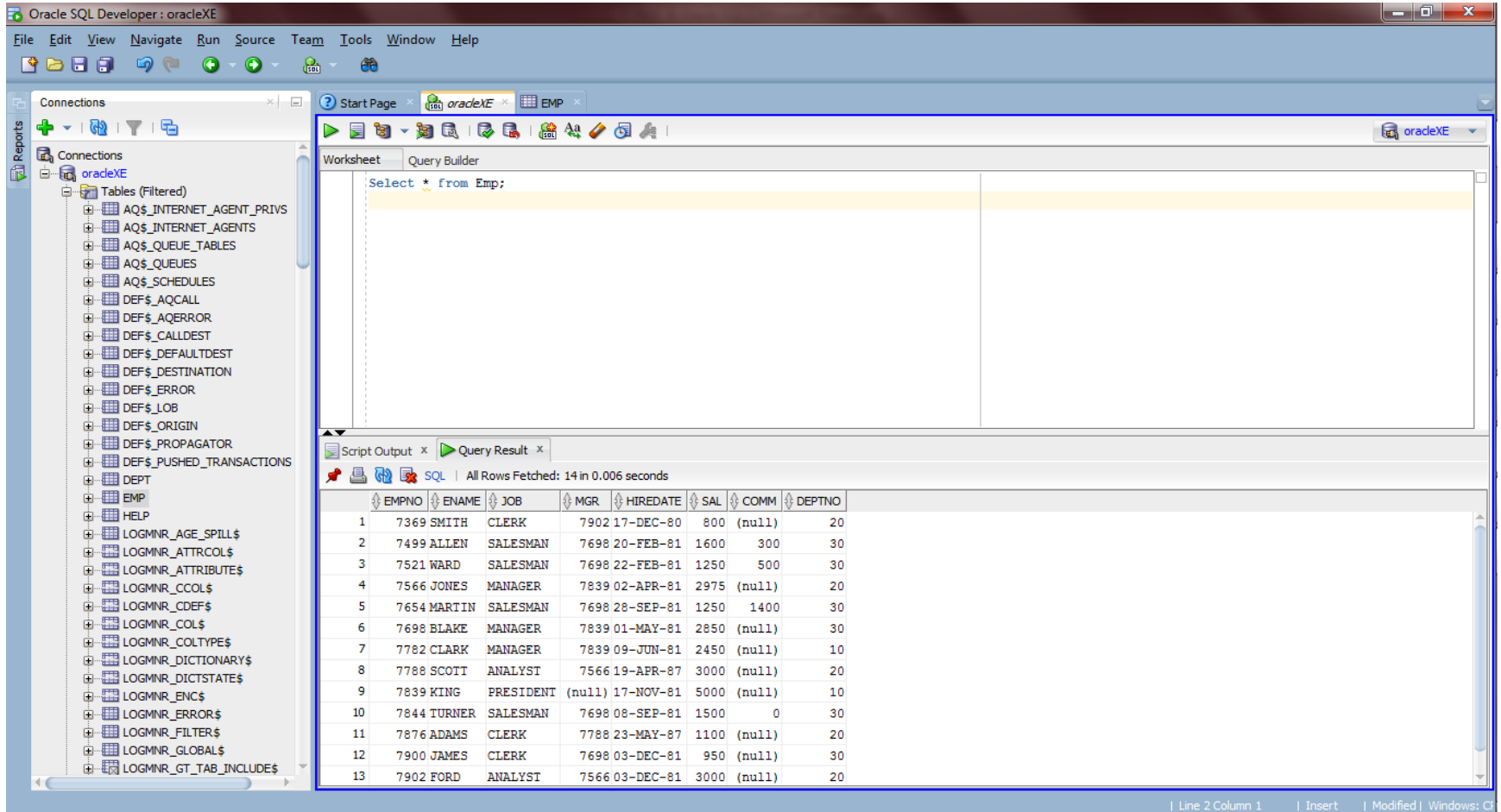
SQL statement entered

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

SQL Developer with Oracle

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	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
BUDGET		NUMBER

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 (sal) FROM emp;

▸ SELECT AVG (sal) FROM emp;

▸ SELECT MAX (sal) FROM emp;

▸ SELECT MIN (sal) FROM emp;

▸ SELECT * FROM employee WHERE salary = (SELECT MIN (salary) FROM employee);

The GROUP BY clause

SELECT deptno, sum (sal)

FROM emp

GROUP BY deptno;

The HAVING Clause

SELECT deptno, sum (sal) FROM emp

GROUP BY deptno

HAVING sum (sal) > 7000;

SELECT deptno, sum (sal)

FROM emp

WHERE deptno in (10, 30)

GROUP BY deptno

HAVING sum (sal) > 8000

ORDER BY sum (sal) desc;

Module 6: Creating Tables

▶ Overview

  ▶ Creating a Table

  ▶ Data Types

Creating Tables

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

CREATE TABLE dept (
      deptno varchar2 (4),
      dname varchar2 (20),
      loc varchar2(20)
);

Data Types

Data Type	Description	Example
char (n)	Fixed-length character data. Max 2000 bytes	deptno char (4)
varchar (n)	Variable-length character data. Max 4000 bytes	deptno varchar2 (4)
varchar2 (n)	Variable-length character data. Max size is 4000 bytes	deptno varchar (4)
number (p, s)	Numeric data, 'p' is the total length and "s" is the number of decimal places.	reading number (5, 2). Maximum value: 99.99
Date	Date and time. Range is 01/01/4712 BC to 31/12/4712 AD.	hiredate date

Data Types

Data Type	Description	Example
long	Variable-length character data. Max 2 GB	remarks long
raw (n)	Binary format data. Max size 2000 bytes	esc_seqraw (15)
Long raw	Same as raw, but maximum size is 2 GB.	picture long raw
BLOB	Stores binary large objects up to 4GB	
CLOB	Stores character large objects up to 4GB.	
BFILE	Enables access to binary file LOBs that are stored in the file system outside the Oracle database. Maximum file size up to 4GB.	

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	VARCHAR2(4)
DNAME	NOT NULL	VARCHAR2(20)
LOC	NOT NULL	VARCHAR2(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, sal, ename) VALUES(1051, 5000, 'Sunil');

INSERT INTO RESEARCH_EMP

SELECT * FROM employee WHERE deptno=20;

Create table RESEARCH_EMP

As

SELECT * FROM employee WHERE deptno=20;

Modifying and Deleting Data

▸ UPDATE emp SET sal = sal + 100;

▸ UPDATE emp SET sal = sal + 200 WHERE deptno= 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 number (2));

ALTER table emp

MODIFY (age number (3));

ALTER table emp DROP column age;

ALTER table employee DROP (comm, 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 number (5) **NOT NULL**,

        ename varchar2 (25) **NOT NULL**,

        deptno varchar2 (4));

UNIQUE Constraints

CREATE TABLE emp(

      empno number (5) CONSTRAINT emp_uq UNIQUE,

      ename varchar2 (25) CONSTRAINT emp_null NOT NULL

);

SELECT constraint_name FROM USER_CONSTRAINTS

      WHERE table_name = 'EMP';

Primary key Constraint

```
CREATE TABLE supplier (
        supp_code number (4) CONSTRAINT code_pk PRIMARY KEY,
        supp_name varchar2 (30)  CONSTRAINT name_uq UNIQUE
);


CREATE TABLE supplier (
        supp_code number (4)
        CONSTRAINT  code_pk PRIMARY KEY,
        supp_name varchar2 (30)
        CONSTRAINT  name_uq UNIQUE
        CONSTRAINT  name_null NOT NULL
);
```

The CHECK Constraint

```
CREATE TABLE employee (
        empno number (5) PRIMARY KEY,
        ename varchar2 (25) NOT NULL,
        deptno varchar2 (4)
        CONSTRAINT deptno_check
        CHECK (deptno > max(deptno) )
);
```

The REFERENCES Constraint

```
CREATE TABLE employee (
        empno number (5)
        CONSTRAINT empno_pk PRIMARY KEY,
        ename varchar2 (25)
        CONSTRAINT ename_null NOT NULL,
        deptno varchar2 (4)
        CONSTRAINT deptno_ref
        REFERENCES dept (deptno)
);
```

The REFERENCES Constraint

```
CREATE TABLE emp (
        empno number (5) primary key,
        ename varchar2 (25) not null,
        deptno varchar2 (4)
        CONSTRAINT deptno_ref
        REFERENCES dept(deptno)
        ON DELETE CASCADE
);
```

Table Constraints

CREATE TABLE emp(

       empno number (4) not null,

       ename varchar2 (40) not null,

       deptno varchar2 (4),

       CONSTRAINT emp_uq

       UNIQUE (empno,ename)

);

Adding Constraints to Columns of an existing Table

ALTER TABLE emp

MODIFY (

        hiredate constraint emp_hiredate not null

);

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

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

Enabling and Disabling Constraints

ALTER TABLE dept

DISABLE CONSTRAINT deptno_pk;

ALTER TABLE dept

ENABLE CONSTRAINT deptno_pk;

ALTER TABLE dept

DISABLE CONSTRAINT deptno_pk

CASCADE CONSTRAINTS;

Dropping a Constraint

ALTER TABLE emp

      DROP CONSTRAINT hiredate;

ALTER TABLE employee

      DROP CONSTRAINT hiredate CASCADE;

ALTER TABLE dept

      DROP COLUMN depno

       CASCADE CONSTRAINTS;

Sequences

- Is a database object which is used to generate automatic unique integer values.

CREATE SEQUENCE sequence_name

  [INCREMENT BY n1]

  [START WITH n2]

  [MAXVALUE n3]

  [MINVALUE n4]

  [CYCLE | NOCYCLE];

Sequences

CREATE SEQUENCE EMP_NUMBER
Increment By 2
Start With 3;

ALTER SEQUENCE emp_number
MAXVALUE 250;

DROP SEQUENCE emp_number;

Using Sequence in SQL

INSERT INTO employee (empno, ename)
VALUES (EMP_NUMBER.NEXTVAL, 'Satish');

SELECT EMP_NUMBER.CURRVAL FROM dual;

Good to know about Sequence

Sequence is supported in most of the databases like oracle, SQL Server, DB2, Postgre etc. However, in MySQL you cannot create the sequence as explained earlier. You need to use AUTO_NUMBER attribute to a column for this.

Module 10: Built-In Functions

▶ Overview

  ▶ Numeric functions

  ▶ Character functions

  ▶ Date functions

  ▶ Special formats with Date data types

  ▶ Conversion functions

Functions on Numeric data types

Function	Returns	Example	Result
ceil (n)	Nearest whole integer greater than or equal to n.	SELECT ceil (9.86) FROM dual;	10
floor (n)	Largest integer equal to or less than n.	SELECT floor (9.86) FROM dual;	9
mod (m, n)	Remainder of m divided by n. If n = 0, then m is returned.	SELECT mod (11, 4) FROM dual;	3
power (m, n)	Number m raised to the power of n.	SELECT power (5, 2) FROM dual;	25
round (n, m)	Number n rounded off to m decimal places.	SELECT round (9.86, 1) FROM dual;	9.9
sign (n)	If n = 0, returns 0. If n > 0, returns 1. If n < 0, returns -1.	SELECT sign (9.86) FROM dual;	1
sqrt (n)	Square root of n.	SELECT sqrt (25) FROM dual;	5

▶ Xoriant Soultions Pvt. Ltd.

Functions on Character data type

Function	Returns	Example	Result
initcap (x)	Changes the first character of each word to capital letters.	SELECT initcap ('king') FROM dual;	King
lower (x)	Converts the entire string to lowercase.	SELECT lower (King') FROM dual;	king
upper (x)	Converts the entire string to uppercase.	SELECT upper ('King') FROM dual;	KING
replace (char, str1, str2)	Every occurrence of str1 in char is replaced with str2.	SELECT replace('Cap' , 'C', 'M') FROM dual;	Map
soundex (x)	Every word that has a similar phonetic sound, even if it is spelled differently.	SELECT ename FROM emp WHERE soundex (ename) ='onkar'	Omkar Onkar Korgaonkar
substr (char, m, n)	Part of char, starting FROM position m and taking characters.	SELECT substr ('Computer', 1, 4) FROM dual;	Comp
length (char)	Length of char.	SELECT length ('Oracle') FROM dual;	6

Functions on Date data types

Function	Returns	Example	Result
sysdate	Current date and time.	SELECT sysdate FROM dual;	25-NOV-97
last_day (date)	Last day of the month for the given date.	SELECT last_day (sysdate) FROM dual;	30-NOV-97
add_months (date, n)	Adds n months to the given date.	SELECT add_months (sysdate, 2) FROM dual;	25-JAN-98
months_between (date1, date2)	Difference in months between date1 and date2.	SELECT months_between (sysdate, '01-JAN-99') FROM dual;	-13.20232
next_day (date, day)	Date is the specified day of the week after the given date.	SELECT next_day (sysdate, 'sunday') FROM dual;	30-NOV-97

Conversion Functions

▶ **The conversion functions are:**

- ▶ to_char()
- ▶ to_number()
- ▶ to_date()

Formats with Date data types

Format	Returns	Example	Result
Y	Last digit of the year.	SELECT to_char (sysdate, 'Y') FROM dual;	4
YY	Last 2 digits of the year.	SELECT to_char (sysdate, 'YY') FROM dual;	14
YYY	Last 3 digits of the year	SELECT to_char (sysdate, 'YYY') FROM dual;	014
YYYY	All 4 digits of the year	SELECT to_char (sysdate, 'YYYY') FROM dual;	2014
year	Year spelled out.	SELECT to_char (sysdate, 'year') FROM dual;	Two thousand fourteen
Q	Quarter of the year (Jan through Feb is 1).	SELECT to_char (sysdate, 'q') FROM dual;	4
MM	Month of the year (01-12).	SELECT to_char (sysdate, 'mm') FROM dual;	11
RM	Roman numeral for month.	SELECT to_char (sysdate, 'rm') FROM dual;	XI

Formats with Date data types

Format	Returns	Example	Result
month	Name of the month as a nine-character long string.	SELECT to_char (sysdate, 'month') FROM dual;	november
WW	Week of the year	SELECT to_char (sysdate, 'ww') FROM dual;	48
W	Week of the month	SELECT to_char (sysdate, 'w') FROM dual;	4
DDD	Day of the year; January 01 is 001; December 31 is 365 or 366.	SELECT to_char (sysdate, 'ddd') FROM dual;	329
DD	Day of the month.	SELECT to_char (sysdate, 'dd') FROM dual;	25
D	Day of the week. Sunday = 1; Saturday = 7.	SELECT to_char (sysdate, 'd') FROM dual;	3
DY	Abbreviated name of the day.	SELECT to_char (sysdate, 'dy') FROM dual;	tue

Format with Date data types

Format	Returns	Example	Result
HH or HH12	Hour of the day (01-12).	SELECT to_char (sysdate, 'hh') FROM dual;	04
HH24	Hour of the day in 24-hour clock.	SELECT to_char (sysdate, 'hh24') FROM dual;	16
MI	Minutes (00-59)	SELECT to_char (sysdate, 'mi') FROM dual;	20
SS	Seconds (00-59)	SELECT to_char (sysdate, 'ss') FROM dual;	22

TO_NUMBER() function

Example	Result	Comment
TO_NUMBER('100.12')	100.12	Converts char data into number format.
TO_NUMBER('$100.12', '$999D99')	100.12	Removes dollar sign & returns the actual number
TO_NUMBER('$100,12', '$999,99')	10012	Removes comma & returns actual number

TO_DATE() function

Example	Result	Comment
to_date('29-10-2009', 'DD-MM-YYYY')	29-OCT-09	
to_date('October.29.2010', 'Month.DD.YYYY')	29-OCT-10	
to_date('January 15, 1989, 11:00 A.M.', 'Month dd, YYYY, HH:MI A.M.')	15-JAN-89	

Reference

http://docs.oracle.com/cd/B28359_01/server.111/b28286/functions001.htm

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, table_name2
WHERE 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…
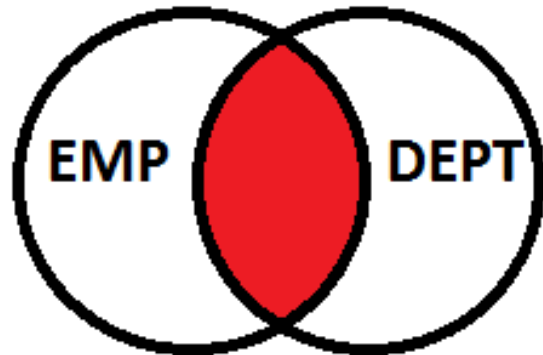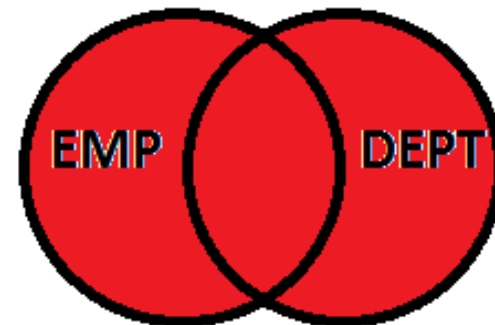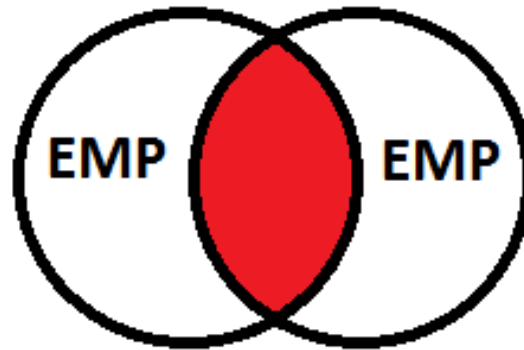

Inner Join
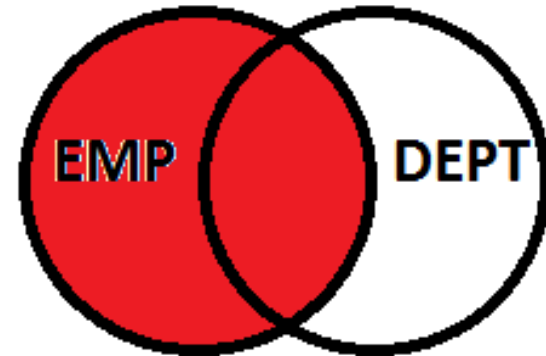
Left Outer Join
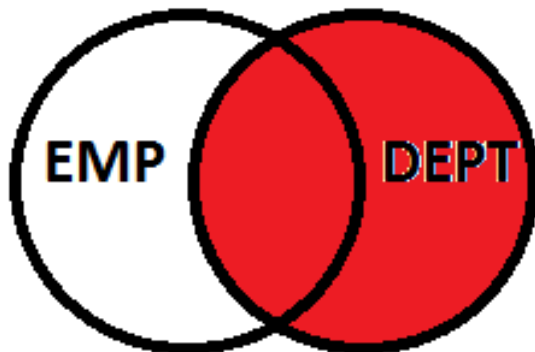
Right Outer Join

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.empno, a.deptno, b.dname

     FROM emp a FULL OUTER JOIN dept b

     ON (a.deptno=b.deptno);

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);