Introduction to Databases & SQL

T5 Bootcamp by SDAIA

Introduction to Databases & SQL

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

Agenda

Program layout

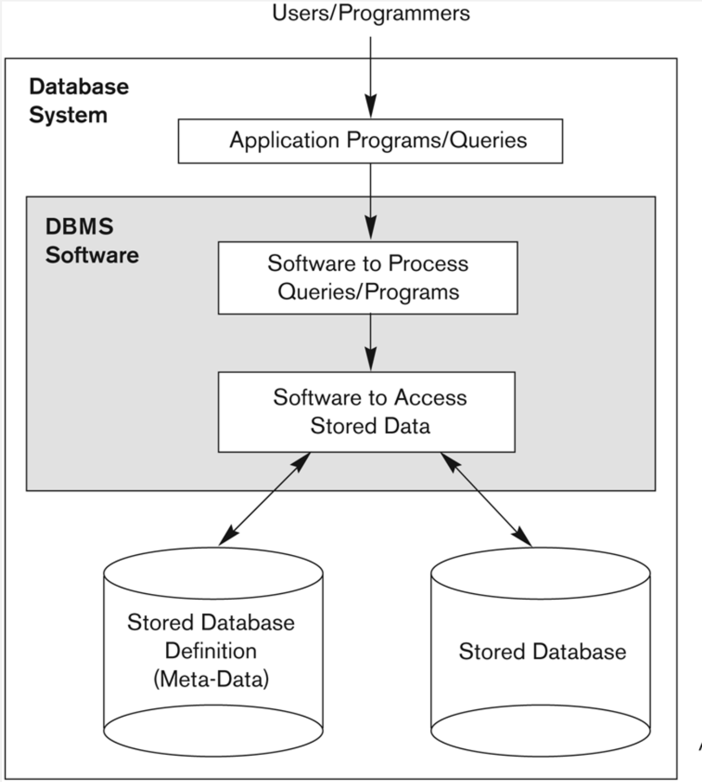
- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY/Company)
- Main Characteristics of the Database Approach
- Advantages of Using the Database Approach
- When Not to Use Databases
- Structured Query Language

Environment

- Grading system
- Module Architecture
  - Slides
  - Hands-on Practice
  - Project
  - Task
  - Exam
- Oracle Live SQL

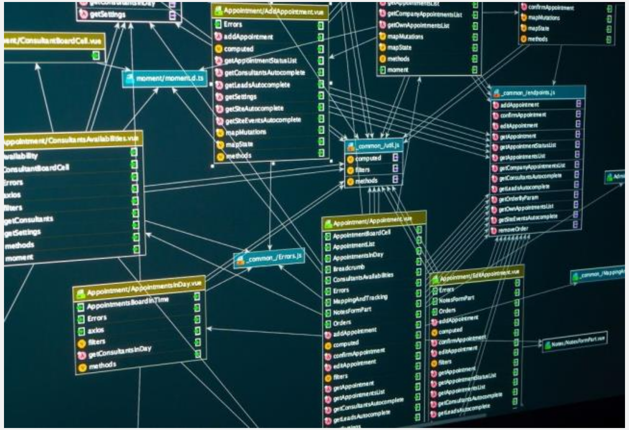
Basic Definitions

- **Database**: collection of related data.
- **Data**: Known facts that can be recorded and have an implicit meaning.
- **Database Management System (DBMS)**: A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System**: The DBMS software together with the data itself. Sometimes, the applications are also included.



Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- Other Databases:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications
  - NoSql Databases



DBMS Functionality

- Specify the characteristics of a specific database, including its data types, structures, and constraints.
- Populate or Create the initial contents of the database on a secondary storage device.
- Interacting with the database:
  - Retrieval: Performing queries and generating reports.
  - Modification: Making insertions, deletions, and updates to its data.
  - Accessing the database via web applications.
- Managing and distributing data among multiple users and application programs concurrently while ensuring data integrity and consistency.

Example of University Database

- Mini-world: University
- Some mini-world entities:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

- **Relationships:**
  - SECTIONs are of specific COURSEs
  - STUDENTs take SECTIONs
  - COURSEs have prerequisite COURSEs
  - INSTRUCTORs teach SECTIONs
  - COURSEs are offered by DEPARTMENTs
  - STUDENTs major in DEPARTMENTs

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Main Characteristics of the Database Approach

- The self-descriptive aspect of a database system: DBMS catalog and metadata.
- Separation between programs and data (program-data independence).
- Data Abstraction: Utilization of a data model.
- Provision for multiple perspectives of the data.
- Facilitation of data sharing and multi-user transaction processing (support for concurrent users).

Advantages of Using the Database Approach

- Managing data redundancy effectively.
- Enabling data sharing among multiple users.
- Implementing measures to prevent unauthorized access to data.
- Ensuring persistent storage for object-oriented program objects.
- Establishing storage structures to optimize query processing efficiency.
- Offering backup and recovery functionalities.
- Offering various interfaces tailored to different user groups.
- Modeling intricate relationships within the data.
- Enforcing integrity constraints on the database.
- Deriving insights and executing actions based on stored data using deductive and active rules.

When not to use a DBMS?

- Main challenges (costs) associated with using a DBMS:
  - High initial investment and potential need for additional hardware.
  - Overhead related to providing generality, security, concurrency control, recovery, and integrity functions.
- Instances where a DBMS might be unnecessary:
  - In cases where the database and applications are straightforward, well-defined, and unlikely to change.
  - When stringent real-time requirements cannot be met due to DBMS overhead.
  - If there is no requirement for multiple users to access the data concurrently.
- Instances where no DBMS may suffice:
  - When the complexity of the data exceeds the capabilities of the database system due to modeling limitations.
  - If database users require special operations that are not supported by the DBMS.

Data Modeling

Data Models

- Data Model:
  - A set of concepts to describe database structure, the operations for manipulating these structures, and certain database enforced constraints.
- Data Model Structure and Constraints:
  - Constructs are used to define the database structure
  - Constructs typically include elements (and their data types) as well as groups of elements (e.g. entity, record, table), and relationships among such groups
  - Constraints specify some restrictions on valid data; these constraints must be enforced at all times

Schemas versus Instances

- Database Schema:
  - The **description** of a database structure, data types, and the constraints on the database.
- Schema Diagram:
  - **Illustrative** display of a database schema.
- Schema Construct:
  - A **component** of the schema or an object within the schema, e.g., STUDENT, COURSE.

STUDENT

Name	Student_number	Class	Major

Schema
databa

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

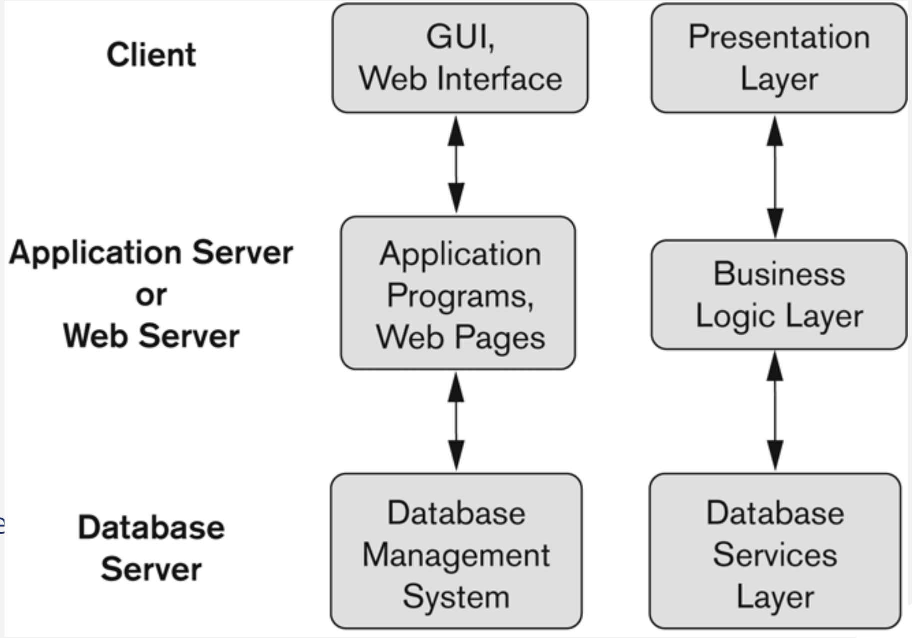
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

DBMS Languages - SQL

- **Data Definition Language (DDL)**
  - Used for defining and managing the structure of database objects.It includes commands for creating, altering, and dropping database objects such as tables, indexes, views, and constraints.
  - Examples of DDL commands include CREATE, ALTER, DROP, and TRUNCATE.
- **Data Manipulation Language (DML)**
  - Used for managing data within a database. It includes commands for inserting, updating, deleting, and retrieving data from database tables.
  - Examples of DML commands include INSERT, UPDATE, DELETE, and SELECT.
- **Data Control Language (DCL)**
  - Used for controlling access to data within a database. It includes commands for granting and revoking permissions and privileges to database objects.
  - Examples of DCL commands include GRANT and REVOKE.

DBMS Server

- Furnishes database inquiry and transaction utilities to the users.
- Relational Database Management System (RDBMS) servers are commonly referred to as SQL servers, query servers, or transaction servers.
- Client applications leverage an Application Programming Interface (API) to connect to server databases through a standard interface, such as:
  - ODBC: Open Database Connectivity standard
  - JDBC: Java Database Connectivity for Java programming access.
- Both the client and server necessitate installation of the suitable client module and server module software for ODBC or JDBC.



Classification of DBMSs

- Based on the data model used
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- Other classifications
  - Single-user (personal computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database)
    vs. distributed (uses multiple computers, multiple databases)

Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase Systems

Data Modeling Using the Entity-Relationship (ER) Model

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

Agenda

- Overview of Database Design Process
- Example Database Application (COMPANY)
- ER Model Concepts
  - Entities and Attributes
  - Entity Types, Value Sets, and Key Attributes
  - Relationships and Relationship Types
  - Weak Entity Types
- Roles and Attributes in Relationship Types
- ER Diagrams - Notation
- ER Diagram for COMPANY Schema

Use Case: COMPANY Database

We need to create a database schema design based on the following requirements of the COMPANY Database:

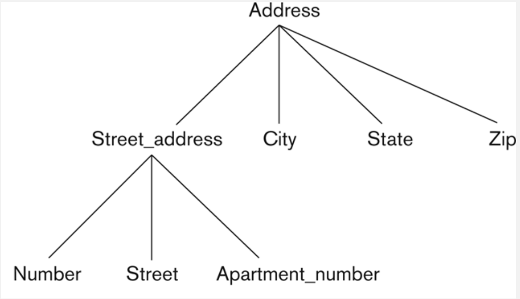
- The company is organized into DEPARTMENTs. Each department has a name, number and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.
- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
  - Each employee works for one department but may work on several projects.
  - We keep track of the number of hours per week that an employee currently works on each project.
  - We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs.
  - 1For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Entities & Attributes

- Entities refer to distinct objects or entities within the real-world domain, which are represented within the database.
  - For instance, an EMPLOYEE named John Smith, the Research DEPARTMENT, or the ProductX PROJECT.
- Attributes serve as characteristics utilized to describe an entity.
  - For instance, an EMPLOYEE entity may possess attributes such as Name, SSN, Address, Sex, and BirthDate.
- Each entity possesses a unique set of values corresponding to its attributes.
  - For instance, a specific employee entity may have attributes with values such as Name='John Smith', SSN='123456789', Address='731 Fondren, Houston, TX', Sex='M', and BirthDate='09-JAN-55'.
- Every attribute is associated with a specific value set or data type, such as integer, string, subrange, enumerated type, etc.

Types of Attributes

- Simple
  - Single atomic value for the attribute. SSN or Sex.
- Composite
  - Attribute may be composed of several components. For example:
  - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
  - Name(FirstName, MiddleName, LastName).
- Multi-valued
  - An attribute may have multiple values. For example, Color of a CAR or PreviousDegrees of a STUDENT.



Entity Types and Key Attributes

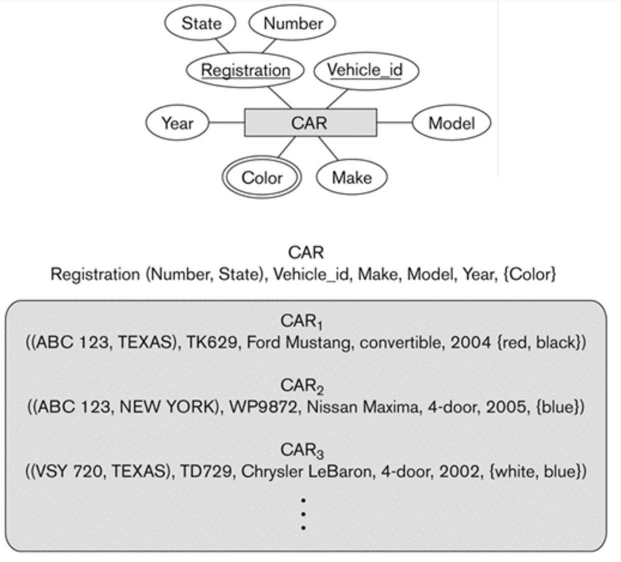
- Entities with the same basic attributes are grouped or typed into an **entity type**. Ex: entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a **key attribute** of the entity type. Ex: SSN of EMPLOYEE.

Entity Types and Key Attributes

- A key attribute may be composite.
  - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), aka license plate number.
- Each key is underlined

Displaying an Entity in ER Diagram

- Entity type is displayed in a rectangular box
- Attributes are displayed in ovals
  - Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is underlined
  - Multivalued attributes displayed in double ovals



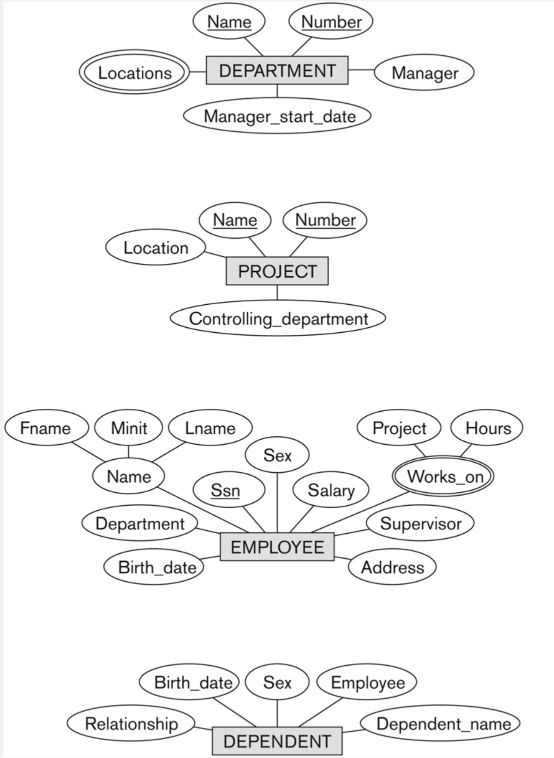
Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

Based on the requirements, we can identify four initial entity types in the COMPANY database:

- DEPARTMENT
- PROJECT
- EMPLOYEE
- DEPENDENT

Their initial design is shown on the following slide

The initial attributes shown are derived from the requirements description



Introducing relationships

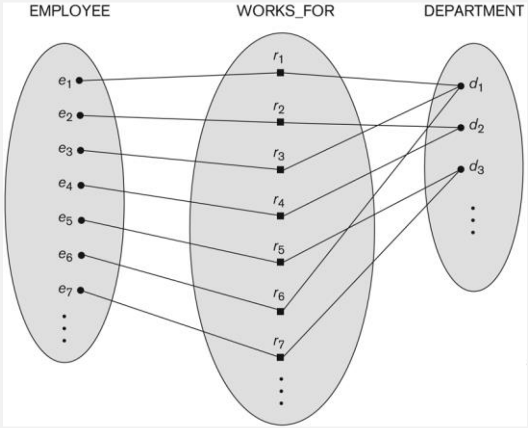
ER model three main concepts: Entities, Attributes, Relationships

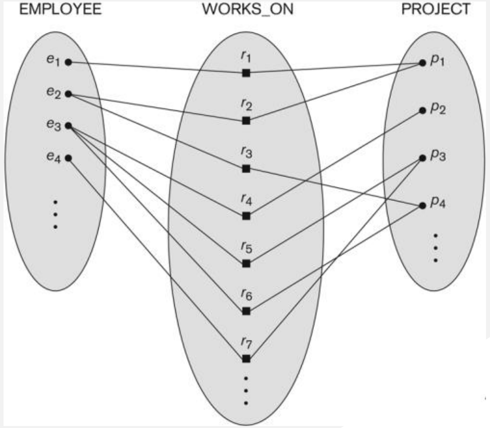
Relationship relates two or more distinct entities with a specific meaning.

- EMPLOYEE John Smith works on the ProductX PROJECT
- EMPLOYEE Franklin Wong manages the Research DEPARTMENT.

Relationship degree is the number of participating entity types.

- Ex: MANAGES and WORKS_ON are binary relationships.

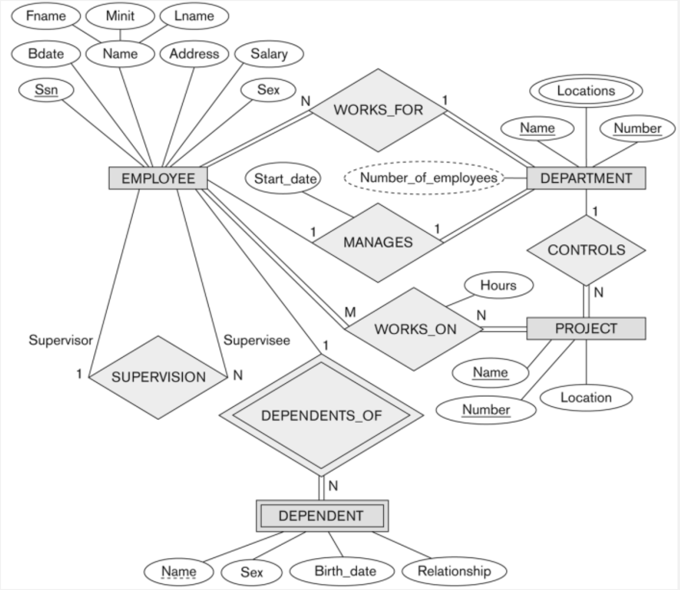




Refining the COMPANY database schema by introducing relationships

Six relationship types are identified

- WORKS_FOR (between EMPLOYEE, DEPARTMENT)
- MANAGES (between EMPLOYEE, DEPARTMENT)
- CONTROLS (between DEPARTMENT, PROJECT)
- WORKS_ON (between EMPLOYEE, PROJECT)
- SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
- DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)



Recursive Relationship Type (Self Join)

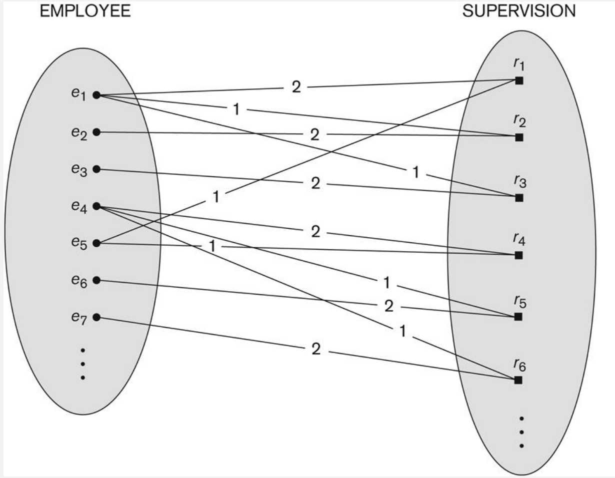
Relationship with the same participating entity type in distinct roles

Example: the SUPERVISION relationship

EMPLOYEE participates twice in two distinct roles:

- supervisor (or boss) role
- supervisee (or subordinate) role

Can you give another example?



Weak Entity Types

Entity without a key attribute and participate in an identifying relationship type with an owner or identifying entity

Entities are identified by the combination of:

- A partial key of the weak entity type
- The particular entity they are related to in the identifying entity type

Example:

- A DEPENDENT entity is identified by the dependent's first name, and the specific EMPLOYEE with whom the dependent is related
- Name of DEPENDENT is the partial key
- DEPENDENT is a weak entity type
- EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Relationships Constraints

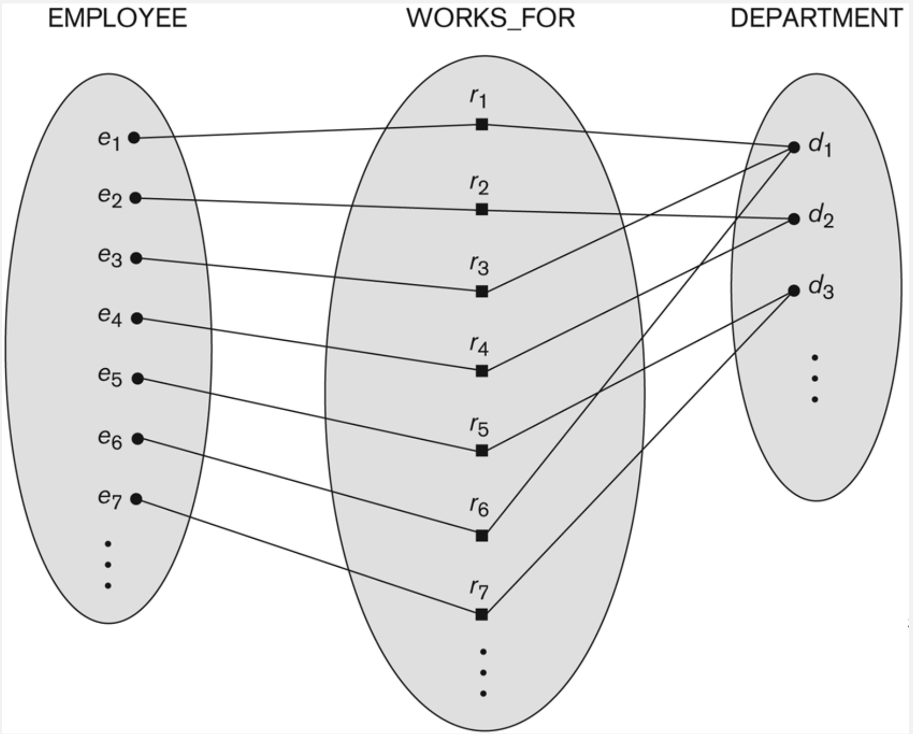
Cardinality Ratio (specifies maximum participation)

- One-to-one (1:1)
- One-to-many (1:N) or Many-to-one (N:1)
- Many-to-many (M:N)

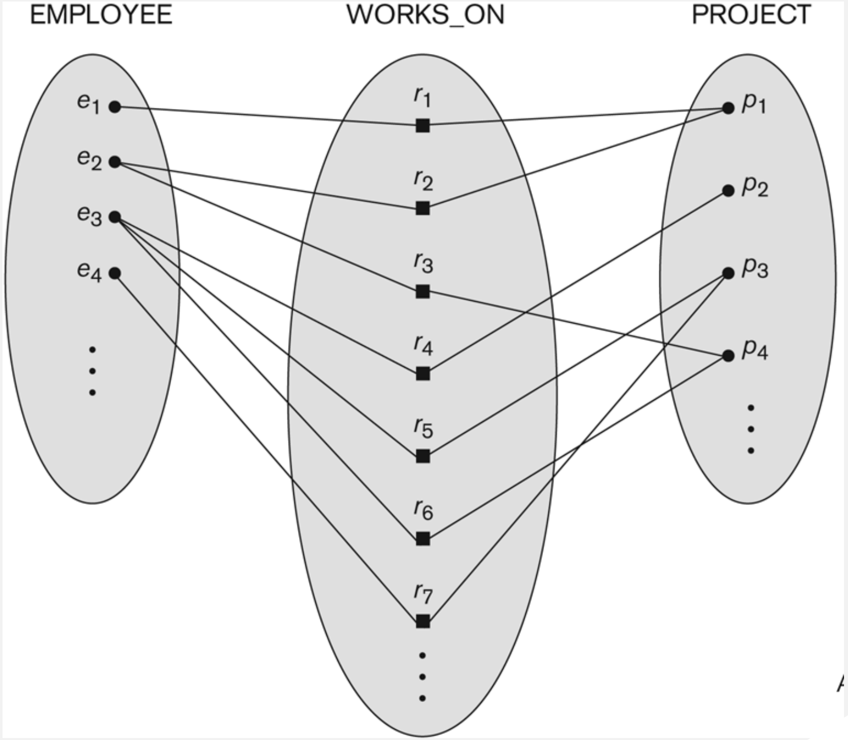
Existence Dependency Constraint (specifies minimum participation)

- zero (optional participation, not existence-dependent)
- one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship



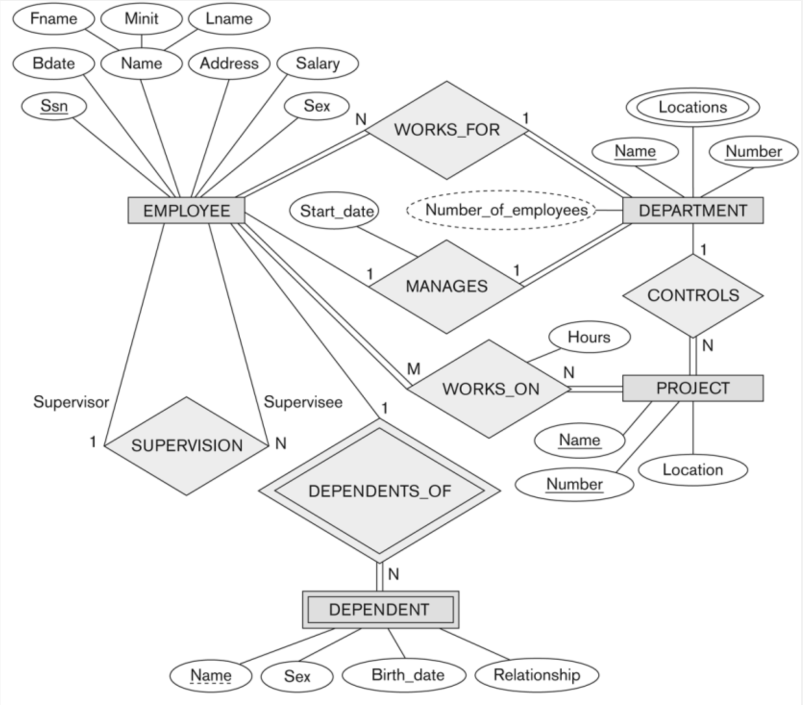
Many-to-many (M:N) Relationship



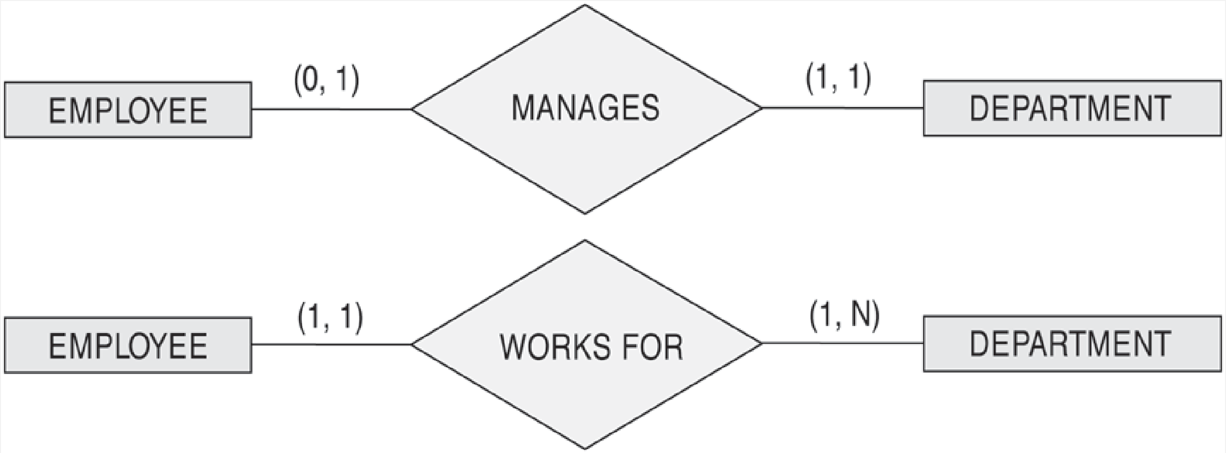
Relationship Attributes

A relationship type can have attributes:

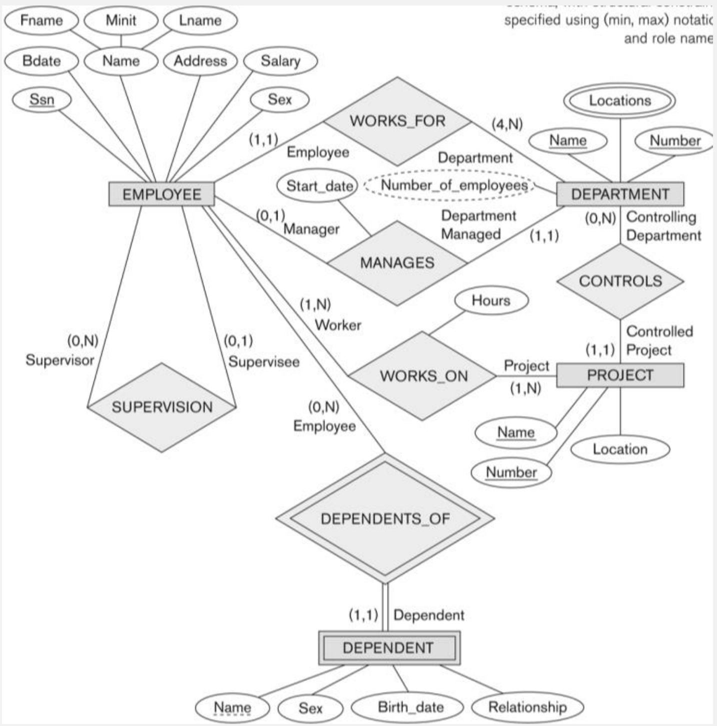
- Ex: HoursPerWeek of WORKS_ON
- Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
- Mostly used with M:N relationships

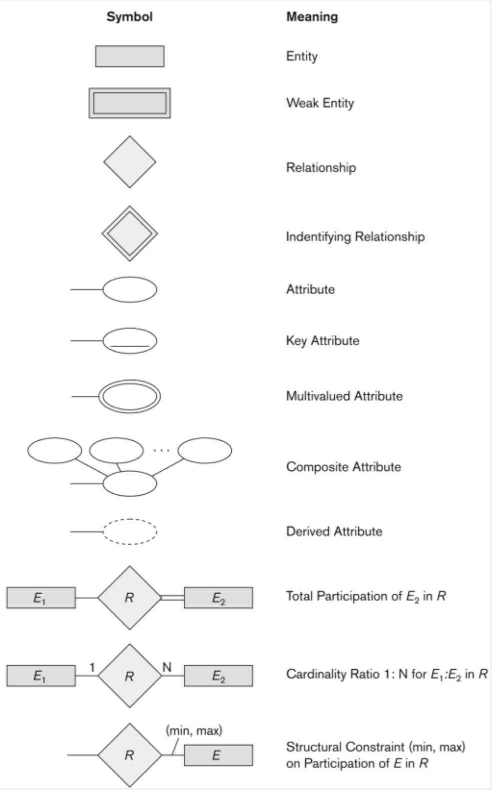


(min,max) notation



COMPANY ER Schema Diagram with (min, max) notation



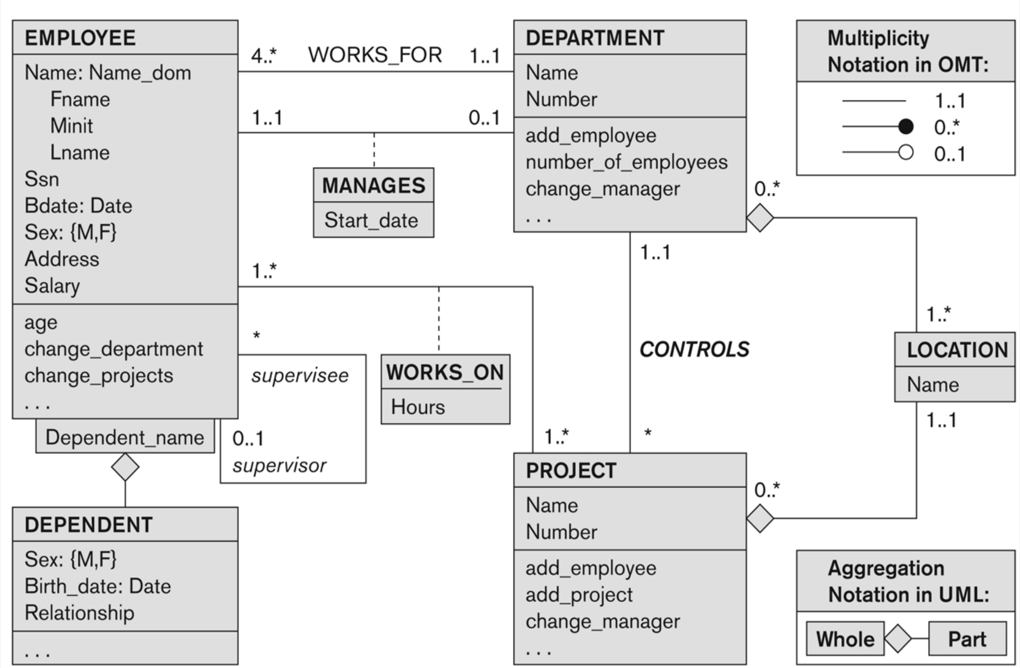


UML class diagrams

Represent classes (entity types) as large rounded boxes

Three sections:

- Entity type (class) name
- Attributes,
- Class operations



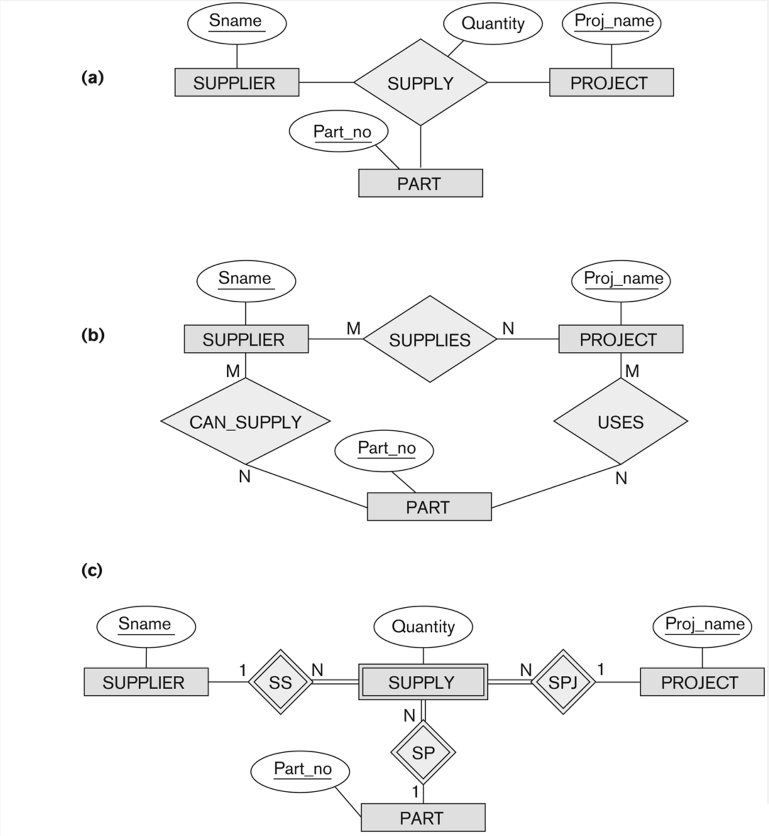
Higher Degree Relationships

Binary Relationship: degree 2

Ternary Relationship: degree 3

N-ary: degree N

Degree specifies number of entities participating in a relationship

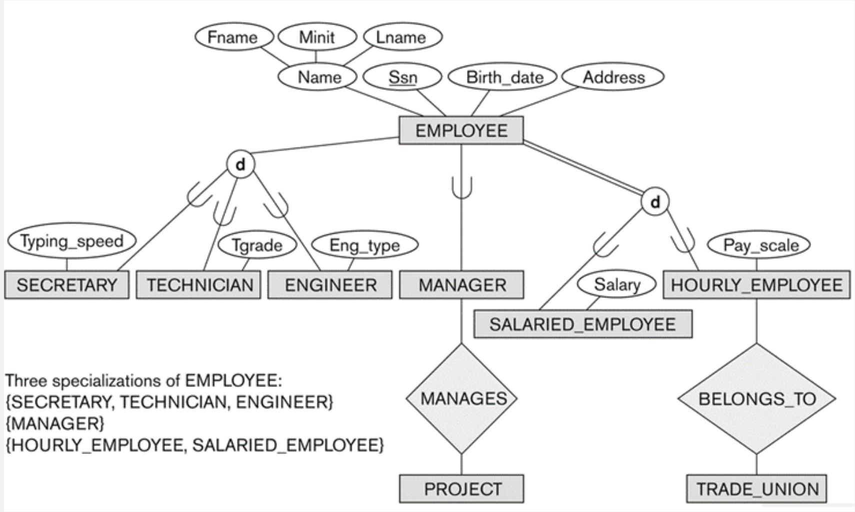


Subclasses and Superclasses

Subgroupings entities

Disjointness Constraint:

- Disjoint (d): an entity can be a member of at most one of the subclasses
- Overlapping (o): that is the same entity may be a member of more than one subclasses



Relational Model Concepts

- The relational Model of Data is based on the concept of a Relation
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in ACM paper

Informal Definitions

Relation is a table of values.

Relation contains a set of rows.

Data elements in each row corresponds to a real-world entity or relationship (tuples)

Column indicates data items meaning in that column (attribute)

Relation Name

Attributes

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Tuples

Relational Integrity Constraints

Constraints are conditions that must hold on all valid relation states.

There are three main types of constraints in the relational model:

- **Key** constraints
- **Entity integrity** constraints
- **Referential integrity** constraints
- **Domain** constraint

Every value in a tuple must be from the domain of its attribute (or it could be null, if allowed for that attribute)

- Several candidate keys, choose one to be your **primary key**.
- Primary key attributes (**underlined**)

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

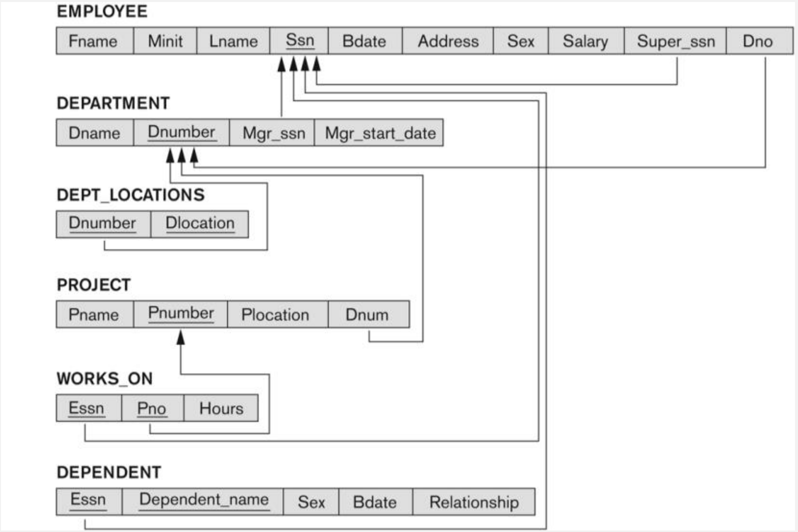
Entity Integrity vs. Referential Integrity

Entity Integrity:

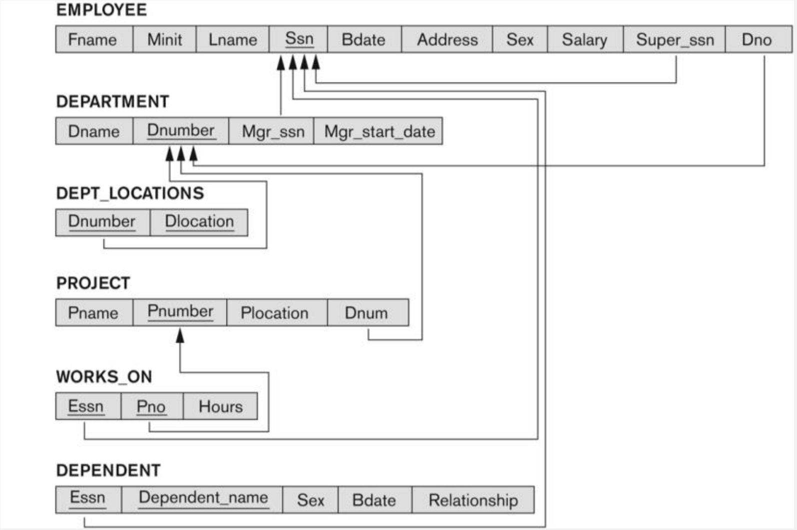
- Primary key cannot have null values

Referential Integrity:

- Specifies a relationship among tuples in two relations



Populated database state for COMPANY



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Update Operations on Relations (Tables)

INSERT a tuple (row).
DELETE a tuple (row).
MODIFY a tuple (row).
Meanwhile:

- Integrity constraints are not violated.
- You can group several update operations together.
- Updates are propagated automatically (integrity constraints).

- Integrity violation:
  - Operation is cancelled (RESTRICT or REJECT option)
  - Inform the user and perform the operation.
  - Additional updates are triggered to correct violations (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine

Possible violations:

- INSERT new tuple:
  - Domain constraint:
    - New tuple value is not of the specified attribute domain
  - Key constraint:
    - Key value in the new tuple already
  - Referential integrity:
    - Foreign key value does not exist in the referenced relation
  - Entity integrity:
    - Primary key value is null

- DELETE :
  - Primary key value is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL RESTRICT option: reject the deletion
  - Above options must be specified during foreign key constraint design

Possible violations

- UPDATE:
  Domain constraint and NOT NULL constraint
  Other constraints may also be violated:
  - Updating primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Specify similar options to DELETE
  - Updating foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute:
    - Can only violate domain constraints

In-Class Exercise

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema.

SQL

Structured Query Language

CREATE TABLE

- Creates a new base table with a name, attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))

Specify NOT NULL constraint

```
CREATE TABLE DEPARTMENT (
DNAME    VARCHAR(10) NOT NULL,
DNUMBER INTEGER      NOT NULL,
MGRSSN  CHAR(9),
MGRSTARTDATE CHAR(9)
      );
```

- Primary key attributes, secondary keys, and referential integrity constraints

```
CREATE TABLE DEPT (
DNAME VARCHAR(10)    NOT NULL,
DNUMBER INTEGER      NOT NULL,
MGRSSN  CHAR(9),
MGRSTARTDATE CHAR(9),
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES
EMP
    );
```

DROP TABLE

```
DROP TABLE DEPENDENT;
```

ALTER TABLE

- Adds an attribute:
  - NULLs in all the previous existing (NOT NULL constraint not allowed)

```
ALTER TABLE EMPLOYEE
ADD JOB VARCHAR(12);
```

REFERENTIAL INTEGRITY OPTIONS

- Specify RESTRICT, CASCADE, SET NULL or SET DEFAULT

```
CREATE TABLE DEPT (
    DNAME               VARCHAR(10)         NOT NULL,
    DNUMBER             INTEGER             NOT NULL,
    MGRSSN              CHAR(9),
    MGRSTARTDATE        CHAR(9),
    PRIMARY KEY (DNUMBER),
    UNIQUE (DNAME),
    FOREIGN KEY (MGRSSN) REFERENCES EMP
    ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

REFERENTIAL INTEGRITY OPTIONS

```
CREATE TABLE EMP(
   ENAME           VARCHAR(30)  NOT NULL,
   ESSN            CHAR(9),
   BDATE           DATE,
   DNO             INTEGER  DEFAULT 1,
   SUPERSSN        CHAR(9),
   PRIMARY KEY (ESSN),
   FOREIGN KEY (DNO) REFERENCES DEPT
   ON DELETE SET DEFAULT ON UPDATE  CASCADE,
   FOREIGN KEY (SUPERSSN) REFERENCES EMP ON DELETE
   SET NULL ON UPDATE CASCADE);
```

Additional Data Types in SQL2 and SQL-99

DATE, TIME, and TIMESTAMP data types

DATE:
- format yyyy-mm-dd

TIME:
- format hh:mm:ss

TIME(i):
- Specifies fractions of a second
- format is hh:mm:ss:ii...i

- **TIMESTAMP:**
  - DATE and TIME components
- **INTERVAL:**
  - Relative value (not absolute value)
  - DAY/TIME intervals
  - YEAR/MONTH intervals
  - Positive or negative
  - Added to or subtracted from an absolute value, the result is an absolute value

Retrieval Queries in SQL

Basic statement : SELECT statement

Basic form of the SQL SELECT statement is called a mapping or a SELECT-FROM-WHERE block

```
SELECT      [DISTINCT ]  <attribute list>
FROM        <table list>
WHERE       <condition>
```

- <attribute list> : attribute of interest to be retrieved
- <table list> : relation names
- <condition> : (Boolean) expression identifying tuples of interest

Relational Database Schema

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Simple SQL Queries

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
SELECT    BDATE, ADDRESS
FROM      EMPLOYEE
WHERE     FNAME='John' AND
MINIT='B' AND LNAME='Smith'
```

- Results may contain duplicate tuples

Joining multiple tables

- Retrieve the name and address of all employees who work for the 'Research' department.

SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE, DEPARTMENT
WHERE
DNAME='Research'
AND DNUMBER = DNO

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE ***DNUM=DNUMBER***
AND MGRSSN=SSN
AND PLOCATION='Stafford'

Aliases, * and DISTINCT, Empty WHERE-clause

- You can use the same attribute name in different relations
- Prefixing the relation name to the attribute name
- Example: EMPLOYEE.LNAME, DEPARTMENT.DNAME

- ALIASES
  - Sometimes you need to refer same table more than once twice
  - Ex : For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
SELECT    E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM      EMPLOYEE E, EMPLOYEE S
WHERE     E.SUPERSSN=S.SSN
```

WHERE-clause

- Missing WHERE-clause = no condition = all tuples
- Ex: Retrieve the SSN values for all employees.

```
SELECT    SSN
FROM      EMPLOYEE
```

- IMPORTANT: When more than one relation is specified in the FROM-clause this will cause CARTESIAN PRODUCT of rows

```
SELECT    SSN, DNAME
FROM      EMPLOYEE, DEPARTMENT
```

USE OF *

- * retrieves all the attribute values = all the attributes

  Examples:

```
Q1C:    SELECT   *
        FROM     EMPLOYEE
        WHERE    DNO=5

Q1D:    SELECT   *
        FROM     EMPLOYEE, DEPARTMENT
        WHERE    DNAME='Research' AND
        DNO=DNUMBER
```

USE OF DISTINCT

- **DISTINCT** eliminates duplicate
- Example: duplicate SALARY

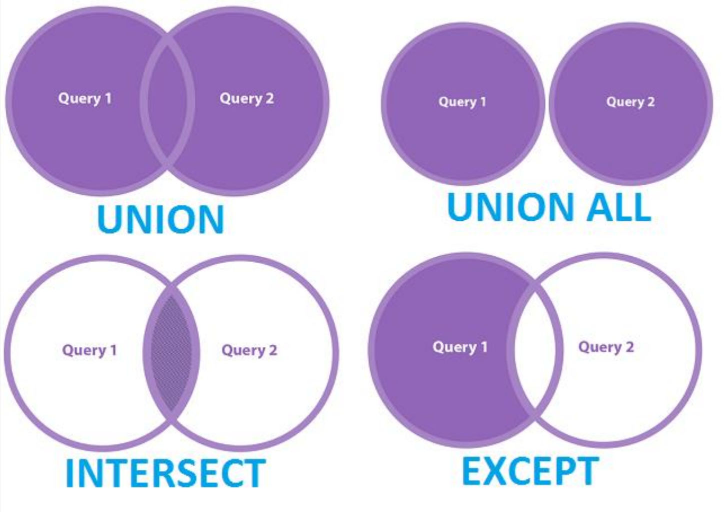
```
SELECT SALARY
FROM   EMPLOYEE

SELECT DISTINCT SALARY
FROM   EMPLOYEE
```

QUERY SET OPERATIONS

- Set operators: (UNION), (MINUS) and (INTERSECT)
- UNION vs UNION ALL (duplicate tuples)
- Queries compatibility :same attributes data type order
- Retrieve all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

```
(SELECT    PNAME
FROM       PROJECT, DEPARTMENT, EMPLOYEE
WHERE      DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')
UNION
(SELECT    PNAME
FROM       PROJECT, WORKS_ON, EMPLOYEE
WHERE      PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')
```



NESTING OF QUERIES

- Nested query is a complete select statement (inner) specified within another query (outer query)
- Ex: Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT    FNAME, LNAME, ADDRESS
FROM      EMPLOYEE
WHERE     DNO IN  (SELECT  DNUMBER
          FROM DEPARTMENT
          WHERE DNAME='Research` )
```

- Correlated nested queries:
  - WHERE-clause of a nested query references an attribute in the outer query
  - Ex: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
SELECT  E.FNAME, E.LNAME
FROM           EMPLOYEE AS E
WHERE E.SSN IN (SELECT ESSN
               FROM DEPENDENT
               WHERE ESSN=E.SSN AND
               E.FNAME=DEPENDENT_NAME)
```

THE EXISTS FUNCTION

- EXISTS checks emptiness of nested query is empty (no rows)
- Ex: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     EXISTS (SELECT *
          FROM      DEPENDENT
          WHERE     SSN=ESSN
          AND FNAME=DEPENDENT_NAME)
```

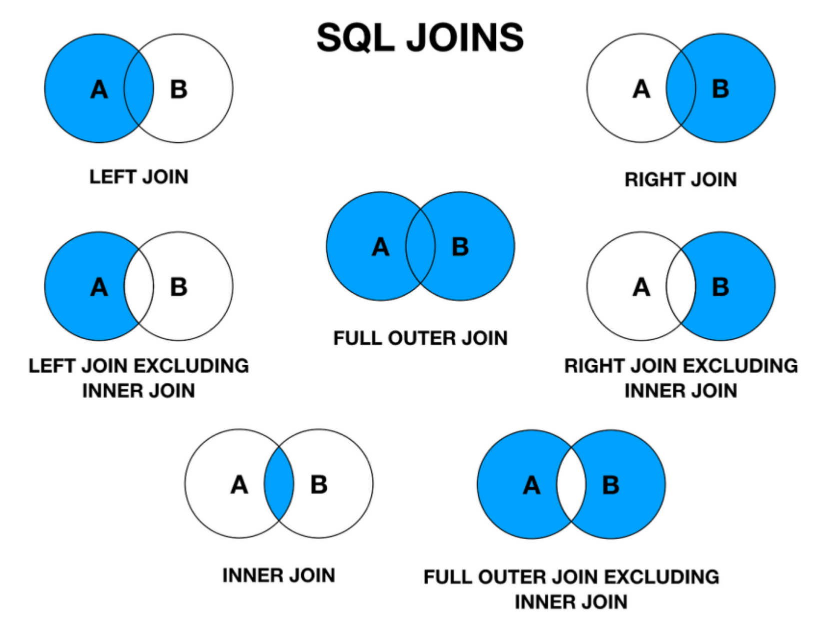
NULLS IN SQL QUERIES

- SQL uses IS or IS NOT to compare NULLs
- Ex: Retrieve the names of all employees who do not have supervisors.

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     SUPERSSN IS NULL
```

Joined Relations Feature

- Can specify a "joined relation" in the FROM-clause
- Regular "theta" JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN,



Joining Relations

- Examples:

```
SELECT    E.FNAME, E.LNAME,
S.FNAME, S.LNAME
FROM      EMPLOYEE E, EMPLOYEE S
WHERE     E.SUPERSSN=S.SSN
```

can be written as:

```
SELECT    E.FNAME, E.LNAME,
S.FNAME, S.LNAME
FROM      (EMPLOYEE E LEFT OUTER
JOIN EMPLOYEES ON
E.SUPERSSN=S.SSN)
```

- Examples:

```
SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND
DNUMBER=DNO
```

could be written as:

```
SELECT FNAME, LNAME, ADDRESS
FROM (EMPLOYEE JOIN DEPARTMENT
ON DNUMBER=DNO)
WHERE DNAME='Research'
```

Joining Relations

- Another Example:

```
SELECT    PNUMBER, DNUM, LNAME,  BDATE, ADDRESS
FROM      (PROJECT JOIN DEPARTMENT ON  DNUM=DNUMBER)
JOIN EMPLOYEE ON  MGRSSN=SSN) )
WHERE     PLOCATION='Stafford'
```

AGGREGATE FUNCTIONS

- Includes COUNT, SUM, MAX, MIN, and AVG
- Query: Find the maximum salary, the minimum salary, and the average salary among all employees.

```
SELECT    MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM      EMPLOYEE
```

- Query: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
SELECT    MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM      EMPLOYEE, DEPARTMENT
WHERE     DNO=DNUMBER AND DNAME='Research
```

AGGREGATE FUNCTIONS (contd.)

- Queries: Retrieve the total number of employees in the company and the number of employees in the 'Research' department .

```
SELECT    COUNT (*)
FROM      EMPLOYEE

SELECT    COUNT (*)
FROM      EMPLOYEE, DEPARTMENT
WHERE     DNO=DNUMBER AND DNAME='Research'
```

GROUPING

- Applies aggregate functions to subgroups of rows having the same value for the grouping attribute(s)
- Aggregate function is applied to each subgroup independently
- GROUP BY-clause lists grouping attributes (must appear in the SELECT-clause)

- Query: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
SELECT    DNO, COUNT (*), AVG (SALARY)
FROM      EMPLOYEE
GROUP BY DNO
```

- Query : For each project, retrieve the project number, project name, and the number of employees who work on that project.

```
SELECT    PNUMBER, PNAME, COUNT (*)
FROM      PROJECT, WORKS_ON
WHERE     PNUMBER=PNO
GROUP BY PNUMBER, PNAME
```

THE HAVING-CLAUSE

- HAVING-clause retrieves functions for only groups satisfying certain conditions
- Query : For each project on which more than two employees work, retrieve the project number, project name, and the number of employees who work on that project.

```
SELECT      PNUMBER, PNAME, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       PNUMBER=PNO
GROUP BY    PNUMBER, PNAME
HAVING      COUNT (*) > 2
```

SUBSTRING COMPARISON

- LIKE compares partial strings
  - '%' (or '*' in some implementations) replaces an arbitrary number of characters,
  - '_' replaces a single arbitrary character
- Query: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX' in it.

```
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE   ADDRESS LIKE '%Houston,TX%'
```

ARITHMETIC OPERATIONS

- Arithmetic operators **'+', '-'. '*', and '/'**
- Query : Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
SELECT   FNAME, LNAME, 1.1*SALARY
FROM     EMPLOYEE, WORKS_ON, PROJECT
WHERE    SSN=ESSN AND PNO=PNUMBER AND PNAME='ProductX'
```

ORDER BY

- ORDER BY clause sorts query result based on some attributes values
- Specify the keyword DESC or ASC (default)

- Query : Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

```
SELECT      DNAME, LNAME, FNAME, PNAME
FROM        DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE       DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
ORDER BY    DNAME, LNAME
```

Summary of SQL Queries

```
SELECT          <attribute list>
FROM            <table list>
[WHERE          <condition>]
[GROUP BY       <grouping attribute(s)>]
[HAVING         <group condition>]
[ORDER BY       <attribute list>]
```

DML

- Three SQL commands : INSERT, DELETE, and UPDATE
- INSERT: add one or more rows to a table, values should be listed in the same order as the attributes were specified in the CREATE TABLE command

Example: **INSERT INTO EMPLOYEE**
VALUES ('Richard','K','Marini', '653298653', '30-DEC-52','98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4)

- Alternativle: specify explicitly the attribute names

Example: **INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)**
VALUES ('Richard', 'Marini', '653298653')

DELETE

- Removes rows from a table
  - Can have WHERE-clause to select the tuples to be deleted
  - Omitting WHERE-clause specifies that all rows tuples are to be deleted
- Examples:

```
DELETE FROM EMPLOYEE WHERE LNAME='Brown'

DELETE FROM EMPLOYEE WHERE SSN='123456789'

DELETE FROM EMPLOYEE WHERE DNO  IN
 (SELECT      DNUMBER
  FROM        DEPARTMENT WHERE DNAME='Research')

DELETE FROM EMPLOYEE
```

UPDATE

- Modify values of one or more selected rows satisfying WHERE-clause if exists while enforcing referential integrity

- Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
UPDATE PROJECT
SET        PLOCATION = 'Bellaire', DNUM = 5
WHERE      PNUMBER=10
```

- Example: Give all employees in the 'Research' department a 10% raise in salary.

```
UPDATE     EMPLOYEE
SET        SALARY = SALARY *1.1
WHERE      DNO  IN (   SELECT DNUMBER
           FROM        DEPARTMENT
           WHERE       DNAME='Research')
```

Normalization of Relations

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations so the resulting designs are of high quality and meet the desirable properties

- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form
- 2NF, 3NF, BCNF , 4NF
  - based on keys and FDs of a relation schema

- **Denormalization**: The process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S subset-of R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]
- A **key** K is a **superkey** with the additional property that removal of any attribute from K will cause K not to be a superkey any more.
- If a relation schema has more than one key, each is called a **candidate key**.
  - One of the candidate keys is arbitrarily designated to be the **primary key**, and the others are called **secondary keys**.
- A **Prime attribute** must be a member of some candidate key
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

First Normal Form

- Disallows
  - Composite attributes
  - Multivalued attributes
  - **Nested relations**; attributes whose values for an individual tuple are non-atomic

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, AliciaJ.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

c)

EMP_PROJ1

Ssn	Ename

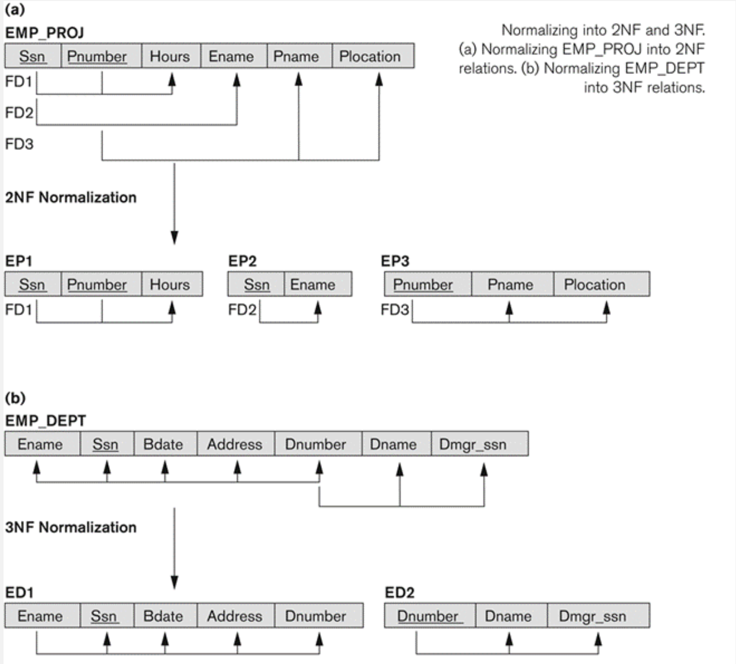
EMP_PROJ2

Ssn	Pnumber	Hours

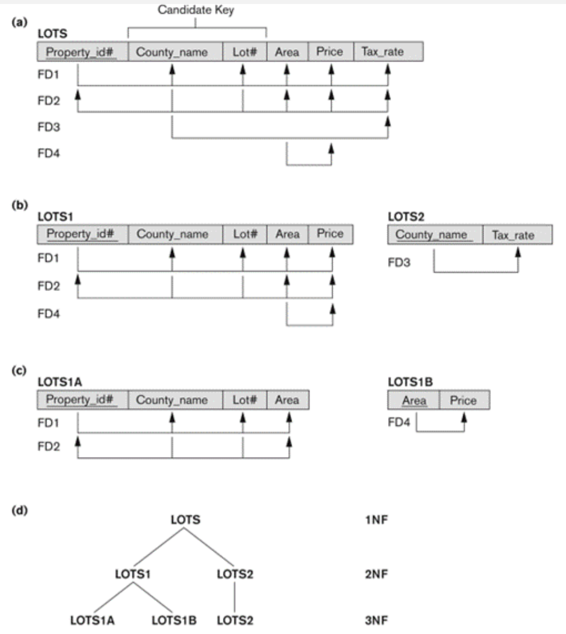
Second Normal Form

- Uses the concepts of FDs, primary key
- Definitions
  - **Prime attribute**: An attribute that is member of the primary key K
  - **Full functional dependency**: a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more
- Examples:
  - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold
  - {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) since SSN -> ENAME also holds
- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key
- R can be decomposed into 2NF relations via the process of 2NF normalization

Normalizing into 2NF and 3NF



Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.

Third Normal Form

- Definition:
  - **Transitive functional dependency**: a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z
- Examples:
  - SSN -> DMGRSSN is a transitive FD
    - Since SSN -> DNUMBER and DNUMBER -> DMGRSSN hold
  - SSN -> ENAME is non-transitive
    - Since there is no set of attributes X where SSN -> X and X -> ENAME
- A relation schema R is in third normal form (3NF) if it is in 2NF and no non-prime attribute A in R is transitively dependent on the primary key
- R can be decomposed into 3NF relations via the process of 3NF normalization
- NOTE:
  - In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key.
  - When Y is a candidate key, there is no problem with the transitive dependency .
  - E.g., Consider EMP (SSN, Emp#, Salary).
    - Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

Normal Forms Defined Informally

1st normal form

- All attributes depend on the **key**

2nd normal form

- All attributes depend on the **whole key**

3rd normal form

- All attributes depend on **nothing but the key**

Data Warehouses

W. H Inmon characterized a data warehouse as:

- **“A subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management’s decisions.”**

Data warehouses have the distinguishing characteristic that they are mainly intended for decision support applications.

- Traditional databases are transactional.

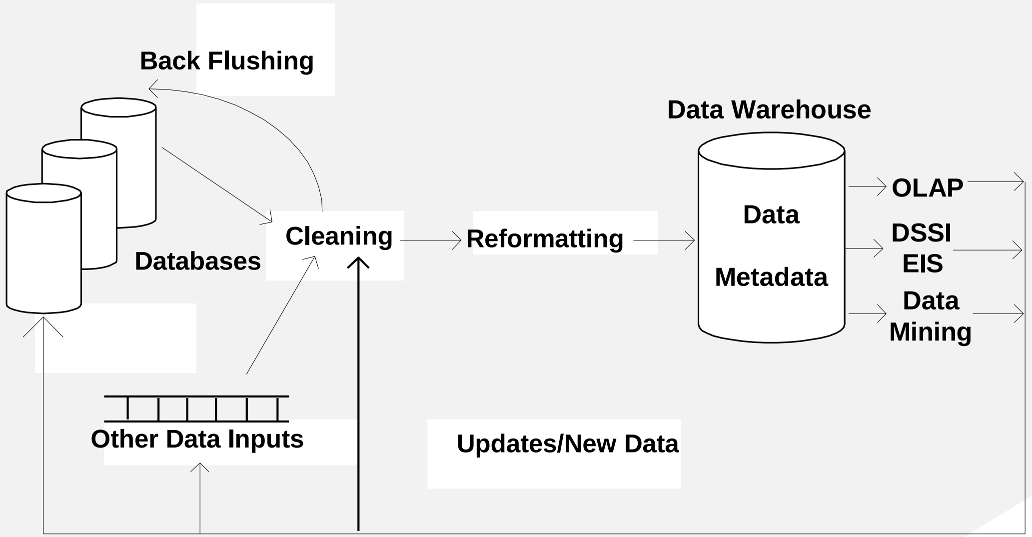
Applications that data warehouse supports are:

- **OLAP** (Online Analytical Processing) is a term used to describe the analysis of complex data from the data warehouse.
- **DSS** (Decision Support Systems) also known as EIS (Executive Information Systems) supports organization’s leading decision makers for making complex and important decisions.
- **Data Mining** is used for knowledge discovery, the process of searching data for unanticipated new knowledge.

Conceptual Structure of Data Warehouse

Data Warehouse processing involves

- Cleaning and reformatting of data
- OLAP
- Data Mining



Comparison with Traditional Databases

Data Warehouses are mainly optimized for appropriate data access.

- **Traditional databases are transactional and are optimized for both access mechanisms and integrity assurance measures.**

Data warehouses emphasize more on historical data as their main purpose is to support time-series and trend analysis.

Compared with transactional databases, data warehouses are nonvolatile.

In transactional databases transaction is the mechanism change to the database. By contrast information in data warehouse is relatively coarse grained and refresh policy is carefully chosen, usually incremental.

Characteristics of Data Warehouses

- Multidimensional conceptual view
- Generic dimensionality
- Unlimited dimensions and aggregation levels
- Unrestricted cross-dimensional operations
- Dynamic sparse matrix handling
- Client-server architecture
- Multi-user support
- Accessibility
- Transparency
- Intuitive data manipulation
- Consistent reporting performance
- Flexible reporting

Classification of Data Warehouses

Generally, Data Warehouses are an order of magnitude larger than the source databases.
The sheer volume of data is an issue, based on which Data Warehouses could be classified as follows.

- **Enterprise-wide data warehouses**
  - They are huge projects requiring massive investment of time and resources.
- **Virtual data warehouses**
  - They provide views of operational databases that are materialized for efficient access.
- **Data marts**
  - These are generally targeted to a subset of organization, such as a department, and are more tightly focused.

Data Modeling for Data Warehouses

Traditional Databases generally deal with two-dimensional data (similar to a spread sheet).

- However, querying performance in a multi-dimensional data storage model is much more efficient.

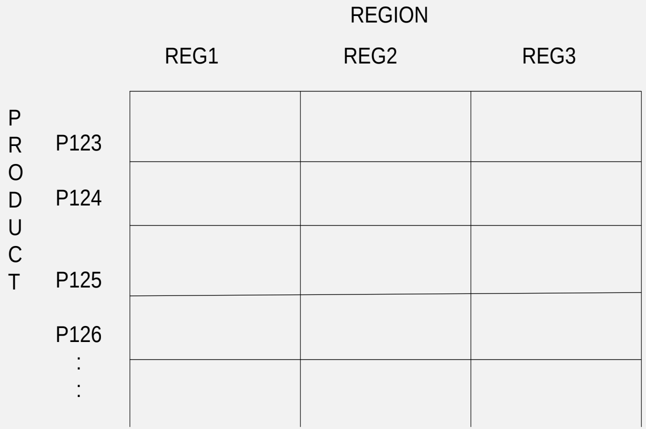
Data warehouses can take advantage of this feature as generally these are

- Non volatile
- The degree of predictability of the analysis that will be performed on them is high.

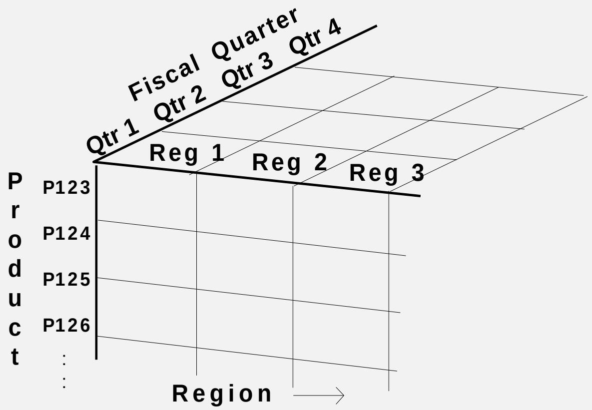
Data Modeling for Data Warehouses

Example of Two- Dimensional vs. Multi- Dimensional

Two Dimensional Model



Three dimensional data cube



Data Modeling for Data Warehouses

Advantages of a multi-dimensional model

- Multi-dimensional models lend themselves readily to hierarchical views in what is known as roll-up display and drill-down display.
- The data can be directly queried in any combination of dimensions, bypassing complex database queries.

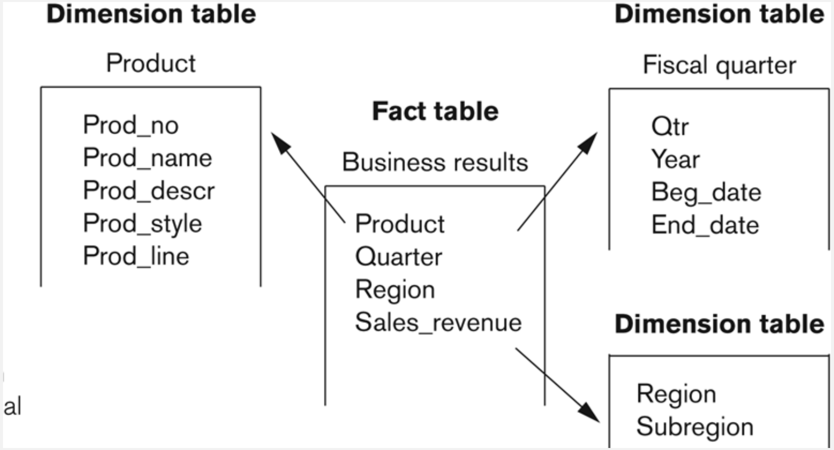
Multi-dimensional schemas are specified using:

- **Dimension table**
    - It consists of tuples of attributes of the dimension.
- **Fact table**
    - Each tuple is a recorded fact. This fact contains some measured or observed variable (s) and identifies it with pointers to dimension tables. The fact table contains the data, and the dimensions to identify each tuple in the data.

Multi-dimensional Schemas

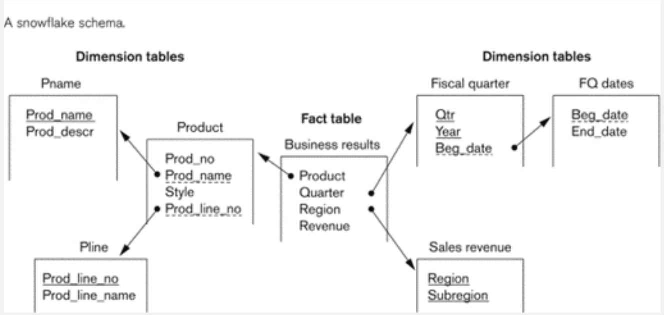
Star schema:

- Consists of a fact table with a single table for each dimension



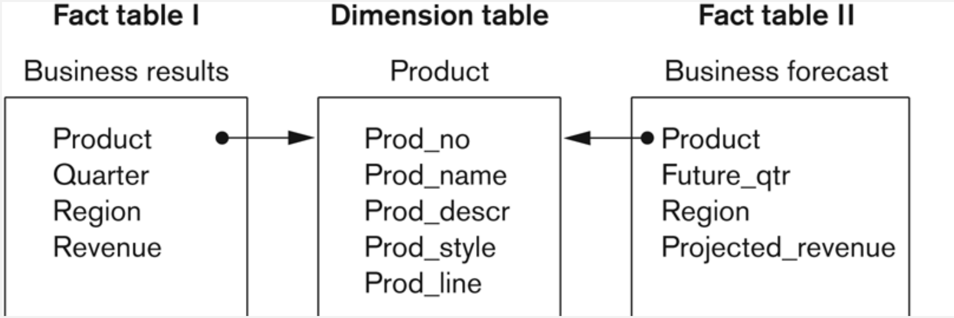
Snowflake Schema:

- It is a variation of star schema, in which the dimensional tables from a star schema are organized into a hierarchy by normalizing them.



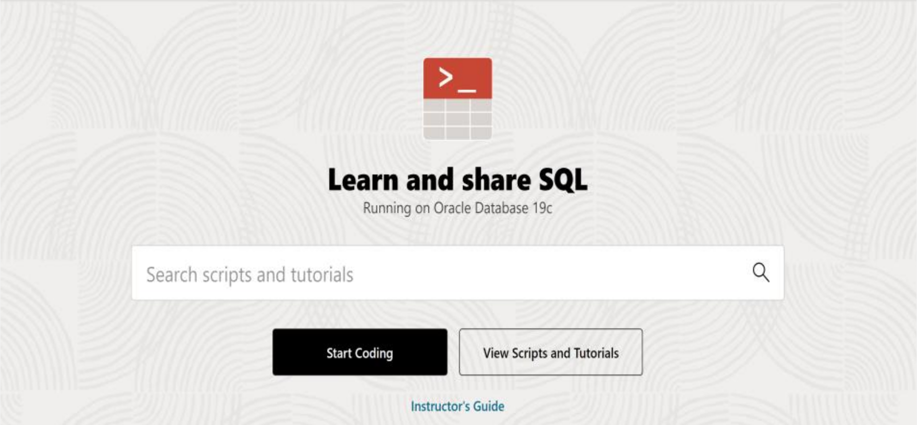
Multi-dimensional Schemas

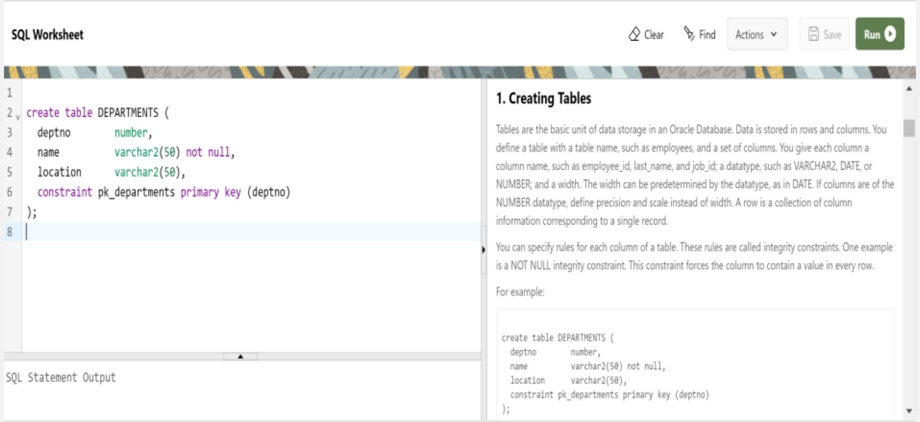
- **Fact Constellation**
  - Fact constellation is a set of tables that share some dimension tables. However, fact constellations limit the possible queries for the warehouse.



Oracle Live SQL

https://livesql.oracle.com/





Oracle Live SQL

Tutorial

Introduction to SQL

This tutorial provides an introduction to the Structured Query Language (SQL), learn how to create t...

create table, create, select, insert, update, delete, drop, drop table, recycle bin, purge

Tutorial

Sorting and Limiting Rows: Databases for Developers

An introduction to sorting data with order by and restricting rows to the top N.

order by, fetch first

Tutorial

Updating table data

This tutorial demonstrates different variations of the UPDATE statement. It includes examples of bas...

Tutorial

Aggregating Rows: Databases for Developers

An introduction to how to summarise data using aggregate functions and group by.

group by, rollup, cube, count, sum

Tutorial

Joining Tables: Databases for Developers

An introduction to the join types available in Oracle Database.

join, inner join, outer join, cross join

Tutorial

Subqueries: Databases for Developers

An introduction to using subqueries in Oracle Database

with clause, in, exists

Tutorial

Union, Minus, and Intersect: Databases for Developers

An introduction to the set operators, union, minus, and intersect

union, minus, intersect

Tutorial

Analytic Functions: Databases for Developers

An introduction to analytic functions.

analytics

Thank You

SDAIA
الهيئة السعودية للبيانات
والذكاء الاصطناعي
Saudi Data & AI Authority