CST204 Database Management System

Module 2

Introduction

Syllabus – Overview

- Module 1 : Introduction and Entity Relationship (ER) model
- Module 2 : Relational Model
- Module 3 : SQL DML and Physical Data Organization
- Module 4 : Normalization
- Module 5: Transaction, concurrency and recovery, recent topics

Introduction

Syllabus – Module 2

- Structure of Relational Databases
  - Integrity Constraints
  - Synthesizing ER diagram to relational schema
  - Introduction to Relational Algebra
  - Select, project, cartesian product operations, join - Equi-join, natural join, query examples
  - Introduction to Structured Query Language (SQL), Data Definition Language (DDL), Table definitions and operations – CREATE, DROP, ALTER, INSERT, DELETE, UPDATE

Relational Data Model

Relation Data Model

- Based on the concept of Relation
- It is a mathematical concept based on the ideas of sets
- Strength of the relational approach is the formal foundation provided by the theory of relations
- Model was first proposed by Dr. E F Codd of IBM in 1970

Relational Data Model

Relation Data Model

- Based on the concept of Relation
- It is a mathematical concept based on the ideas of sets
- Strength of the relational approach is the formal foundation provided by the theory of relations
- Model was first proposed by Dr. E F Codd of IBM in 1970

Relational Data Model

Relation Data Model

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

Relational Data Model

Relation – Informal Definition

- Relation is defined as a table of values
- Relation is a set of rows
- Relation is a set of columns
- Relation is a set of rows in which each row represents an entity or relationship
- Relation is a set of rows in which each row has a value of an item or set of items that uniquely identifies that row in the table
- Sometimes row ids or sequential numbers are assigned to identify the rows in the table
- Each column typically is called by its column name or column header or attribute name

Relational Data Model

Relation – Formal Definition

- Schema of a Relation, $R(A_1, A_2, \ldots, A_n)$
- Relation schema R is defined over **attributes** $A_1, A_2, \ldots, A_n$
- D is called the domain of A_i and is denoted by **dom(A_i)**
- R is called the **name** of this relation
- **Degree (or arity)** of a relation is the number of attributes n of its relation schema
- For example

  STUDENT(Name, Roll_number, Home_Phone, Address, Age, CGPA)

  STUDENT(Name: string, Roll_number: integer, Home_Phone: integer, Address: string, Age: integer, CGPA: real)

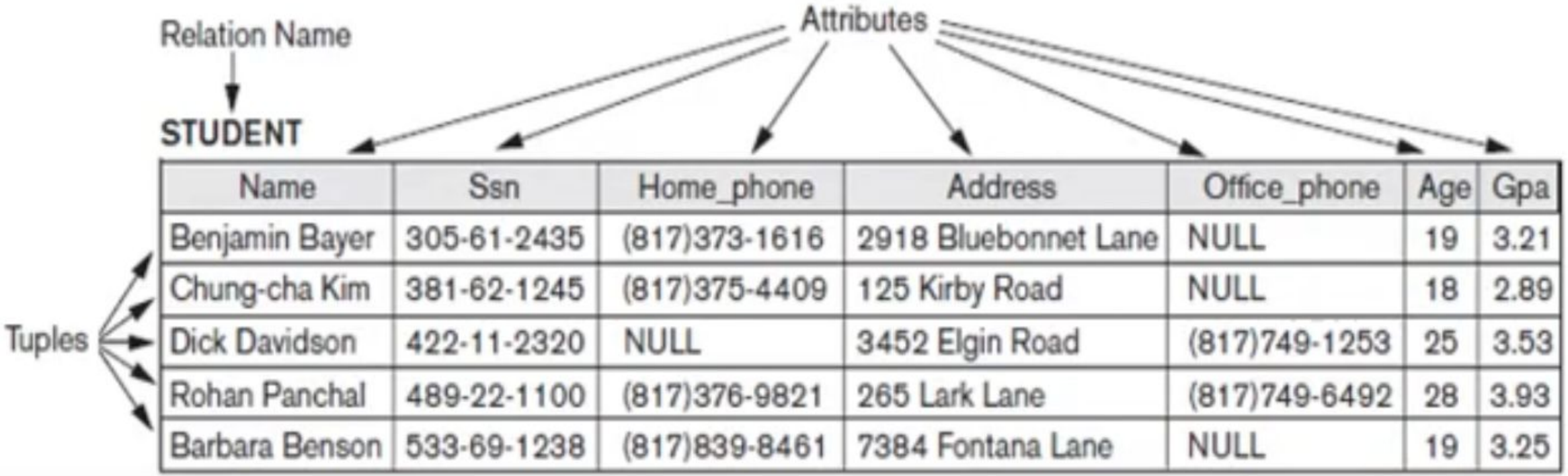
Relational Data Model

Relation – Formal Definition

- Tuple is an ordered set of values (informally called as rows in a table)
- CUSTOMER(Cust_id, Cust_name, Address, Phone_number)
- <632895, “John Smith”, “101 Main St. Atlanta, GA 30332”, “9993434394”>
- A relation may be regarded as set of tuples (rows)
- Columns in a table are also called attributes of the relation

Relational Data Model

Relation – Formal Definition



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relational Data Model

Relation – Formal Definition

- **Domain** is the pool of values from which a particular value is taken
- For example, mobile numbers are the set of 10 digit numbers taken from a domain
- Domain may have a data type or a format defined for it
- **Relation state r** of the relation schema $R(A_1, A_2, \ldots, A_n)$, also denoted by **r(R)**, is a set of n tuples $r = \{t_1, t_2, \ldots\ldots, t_n\}$
- Each tuple t is an ordered list of n values $t = v_1, v_2, \ldots, v_n$, where each value vi, 1 <= i <= n, is an element of $dom(A_i)$ or is a special NULL value

Relational Data Model

Relation – Formal Definition

- **A relation state r(R)** is a mathematical relation of degree n

$$r(R) \subseteq (dom(A1) \times dom(A2) \times \ldots \times (dom(An))$$

- R is called schema of the relation
- r(R) is a specific value or population of R
- R is also called the intension of a relation
- r is also called the extension of a relation

Relational Data Model

Relation – Definition Summary

Informal Terms		Formal Terms
Table		Relation
Column		Attribute/Domain
Row		Tuple
Values in a column		Domain
Table Definition		Schema of a Relation
Populated Table		Extension

Relational Data Model

Characteristics of Relations

- **Ordering of tuples in a relation**
  - Tuples are not considered to be ordered
- **Ordering of attributes in a relation schema R**
  - We will consider the attributes in $R(A_1, A_2, \ldots A_n)$ and the values in $t = <v_1, v_2, \ldots v_n>$ to be ordered
- **Values in a tuple**
- All values are considered atomic (indivisible)
- A special null value is used to represent values that are unknown or inapplicable to certain tuples

Relational Data Model

Relational Model Notation

- A relation schema R of degree n is denoted by $R(A_1, A_2, \ldots A_n)$
- Uppercase letters Q, P, R, S denotes relation names
- Lowercase letters q, p, r, s denotes relation states
- Letters t, u, v denote tuples

t = <'Barbara Benson', '533-69-1238', '(817)839-8461', '7384 Fontana Lane', NULL, 19, 3.25>

t[Name] = <'Barbara Benson'>, and t[Ssn, Gpa, Age] = <'533-69-1238',3.25, 19>.

Relational Data Model

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation instances
- Different types of constraints used in relational model are
- Inherent model-based constraints or implicit constraints
- Constraints inherent in the data model
- Schema based constraints or explicit constraints
- Specified using DDL commands
- Application based or semantic constraints or business rules
- Specified using application program

Relational Data Model

Relational Integrity Constraints

- Our focus is on Inherent model based constraints which consists of:
- Key constraints
- Entity Integrity constraints
- Referential Integrity constraints

Relational Data Model

Inherent model based constraints - Key constraints

- A key in DBMS is an attribute or a set of attributes that help to uniquely identify a tuple (or row) in a relation (or table)
- Keys are also used to establish relationships between the different tables and columns of a relational database
- Individual values in a key are called key values
- There are mainly four types of keys in DBMS:
  - Primary Key
  - Candidate Key
  - Super Key
  - Foreign Key

Relational Data Model

Inherent model based constraints - Key constraints

- Primary Key
- A primary key is a column of a table or a set of columns that helps to identify every record present in that table uniquely
- There can be only one primary Key in a table
- Also, the primary Key cannot have the same values repeating for any row
- Every value of the primary key has to be different with no repetitions
- Super Key
- Super Key is the set of all the keys which help to identify rows in a table uniquely
- This means that all those columns of a table than capable of identifying the other columns of that table uniquely will all be considered super keys

Relational Data Model

Inherent model based constraints - Key constraints

- Candidate Key
- Candidate keys are those attributes that qualifies to be a primary key
- The Primary Key of a table is selected from one of the candidate keys
- So, candidate keys have the same properties as the primary keys

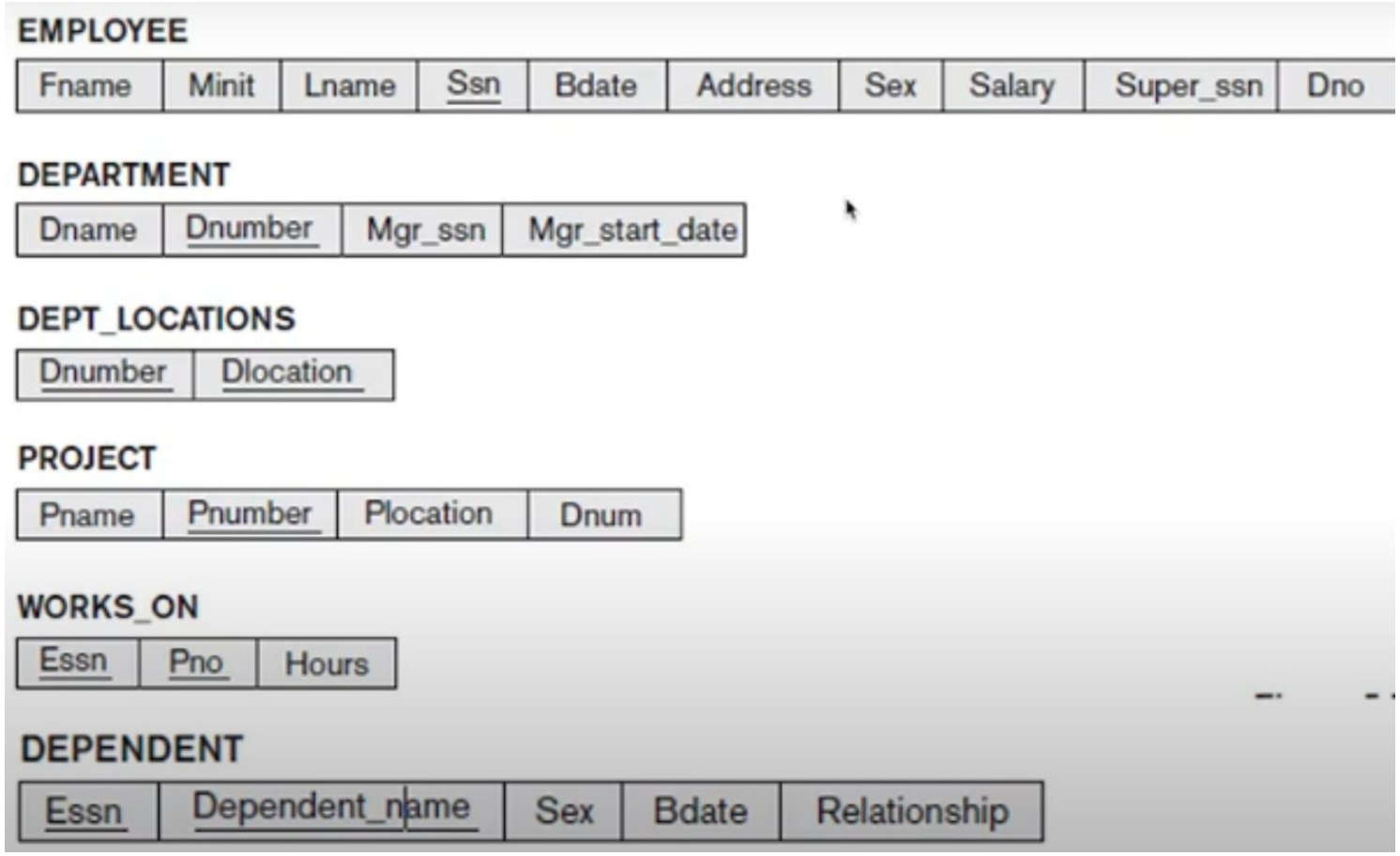
Relational Data Model

Inherent model based constraints – Entity integrity constraints

- Relational Database Schema
- A set S of relation schemas that belong to the same database
- S is the name of the database
- $S = \{R_1, R_2, \ldots, R_n\}$
- Entity Integrity
- Primary key attribute PK of each relation schema R in S cannot have null values in any tuple of r
- t[PK] not equal to null for any tuple t in r(R)

Relational Data Model

Inherent model based constraints – Entity integrity constraints



Relational Data Model

Inherent model based constraints – Entity integrity constraints

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

Relational Data Model

Inherent model based constraints – Entity integrity constraints

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

Relational Data Model

Inherent model based constraints Referential integrity constraints

- It is the constraints involving two relations
- Used to specify a relationship among tuples in two relations – the referencing relation and the referenced relation
- Tuples in the referencing relation R1 attributes Foreign Key (FK) that reference the primary key attributes PK of the referenced relation R2
- Foreign Key is used to establish relationships between two tables
- A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table
- Foreign keys help to maintain data and referential integrity

Relational Data Model

Inherent model based constraints Referential integrity constraints

EMPLOYEE

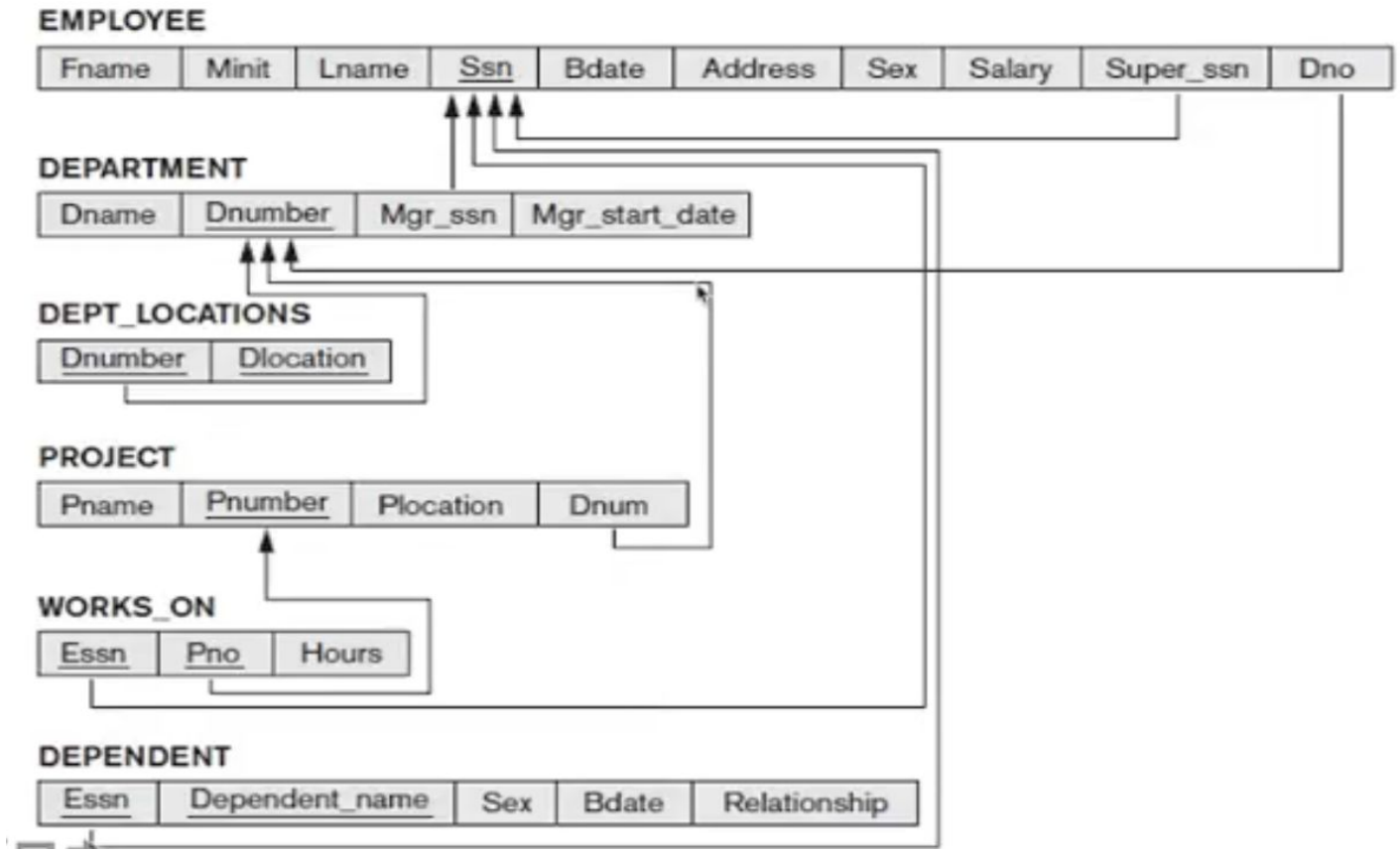
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	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

Relational Data Model

Inherent model based constraints Referential integrity constraints



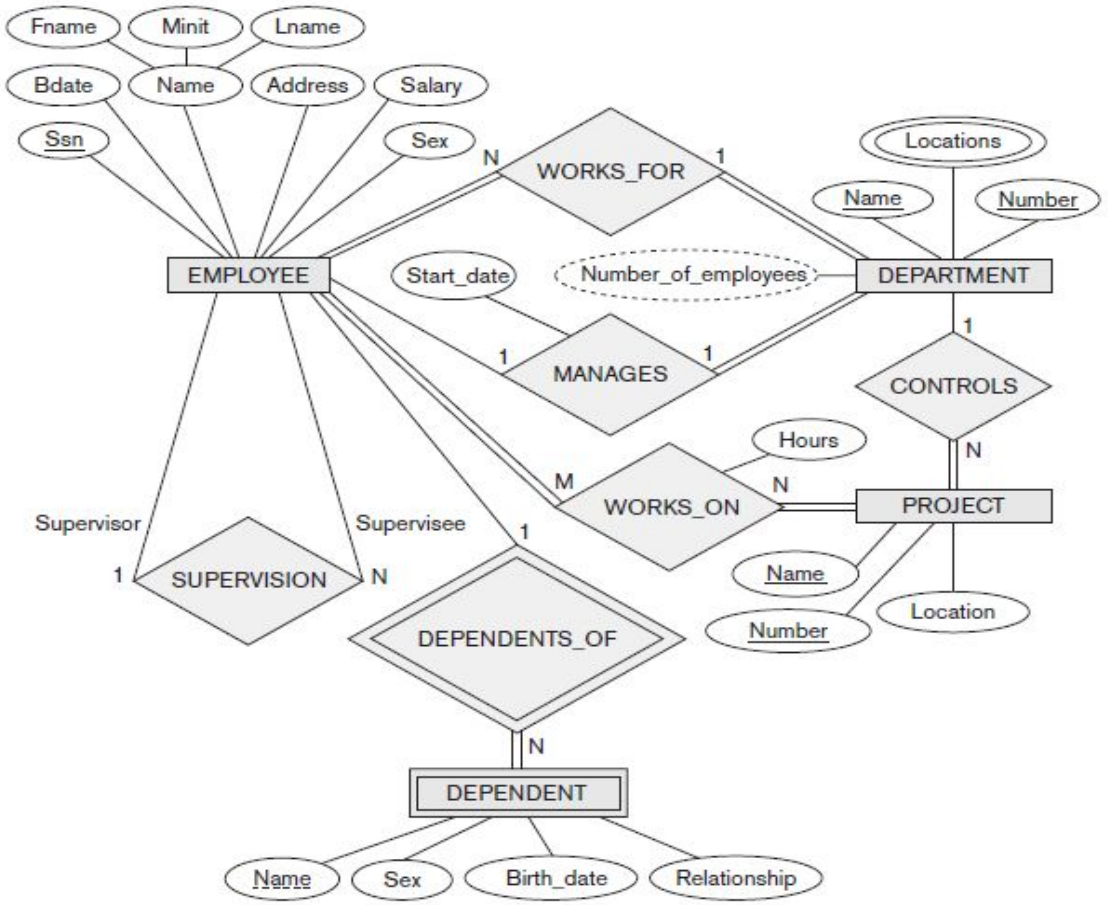
Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 1: Mapping of regular entity types
- Step 2: Mapping of weak entity types
- Step 3: Mapping of binary 1:1 relation types
- Step 4: Mapping of binary 1:N relationship types
- Step 5: Mapping of Binary M:N relationship types
- Step 6: Mapping of multivalued attributes
- Step 7: Mapping of N-ary relationship types

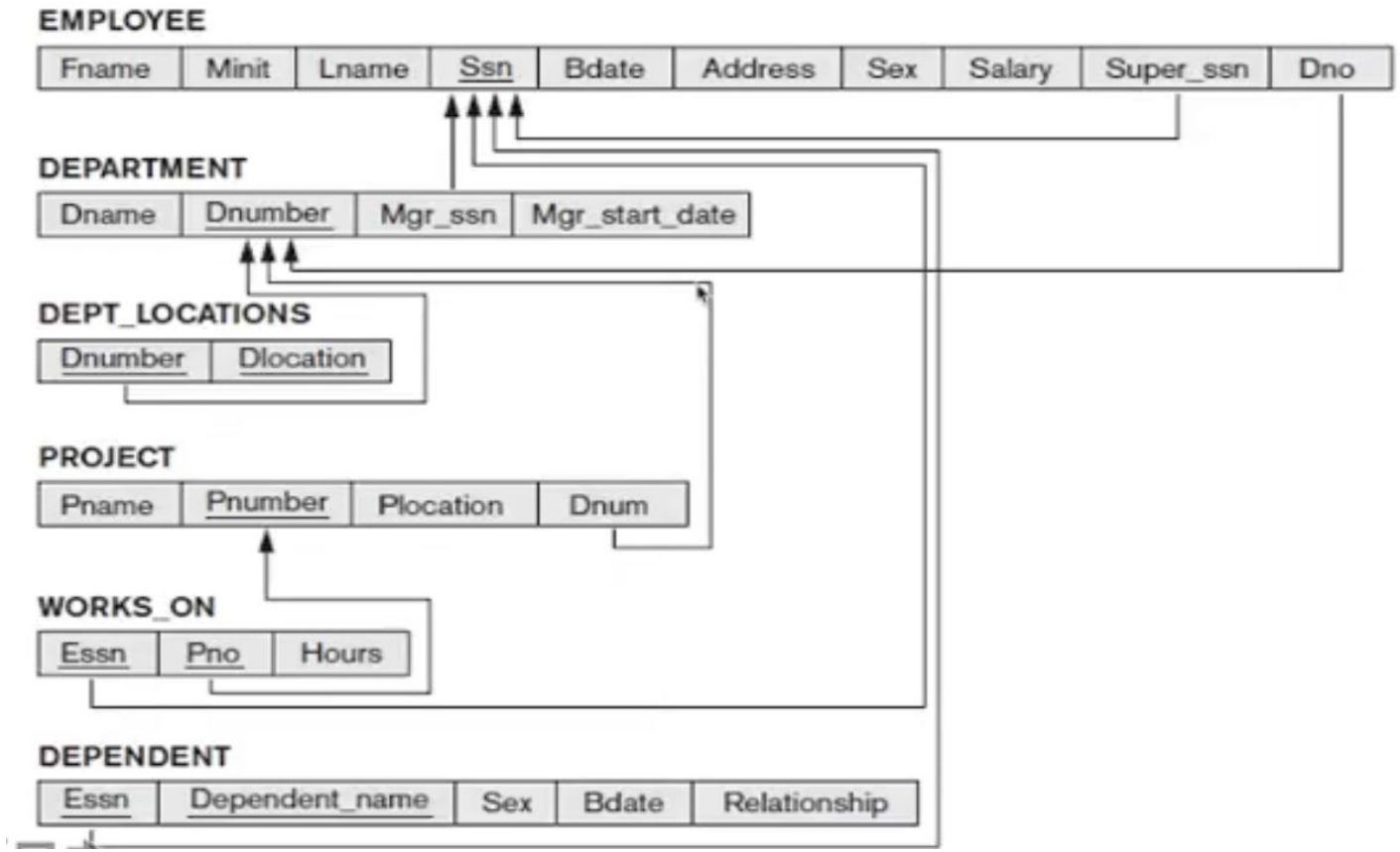
Relational Data Model

Synthesizing ER Diagram to Relational Model



Relational Data Model

Synthesizing ER Diagram to Relational Model



Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 1: Mapping of regular entity types
- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E
- Choose one of the key attributes of E as the primary key for R
- If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 1: Mapping of regular entity types
- After Step 1

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 2: Mapping of weak entity types
- For each weak entity type W with owner entity type E, create a relation R and include all simple attributes of W as attributes of R
- In addition, include as foreign key attributes of R the primary key attribute of the relations that corresponds to the owner entity type
- Primary key of R is the combination of the primary keys of the owner and the partial key of the weak entity type W if any
- After step 2

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 3: Mapping of binary 1:1 relation types
- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that corresponds to the entity types participating in R
- Three possible approaches
- *Foreign key approach*
- Choose one of the relations as S
- Include a foreign key in S the primary key of T
- It is better to choose an entity type with total participation in R in the role of S

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 3: Mapping of binary 1:1 relation types
- 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in MANAGES relationship type is total

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 3: Mapping of binary 1:1 relation types
- *Merged relation option*
- Merging the two entity types and the relationship into a single relation
- Appropriate when both participations are total
- *Cross-reference or relationship relation option*
- Set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 4: Mapping of binary 1:N relationship types
- For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N side of the relationship type
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R
- Include any simple attributes of the 1:N relationship type as attributes of S

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 4: Mapping of binary 1:N relationship types
- WORKS_FOR, CONTROLS and SUPERVISION

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Dno

PROJECT

Pname	Pnumber	Plocation	Dnum

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 4: Mapping of binary 1:N relationship types
- After step 4

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation	Dnum

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 5: Mapping of binary M:N relationship types
- For each regular binary M:N relationship type R, create a new relation S to represent R
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types
- Their combinations will form the primary key of S
- Also include any simple attributes of the M:N relationship type as attribute of S

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 5: Mapping of binary M:N relationship types
- After Step 5

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation	Dnum

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

WORKS_ON

Essn	Pno	Hours

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 6: Mapping of multivalued attributes
- For each multivalued attribute A, create a new relation R include an attribute corresponding to A plus the primary key attribute K as a foreign key in R of the relation that represents the entity type of relationship type that has A as an attribute
- The primary key of R is the combination of A and K
- If the multivalued attribute is composite, we include its simple components

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 6: Mapping of multivalued attributes
- After Step 6

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

Relational Data Model

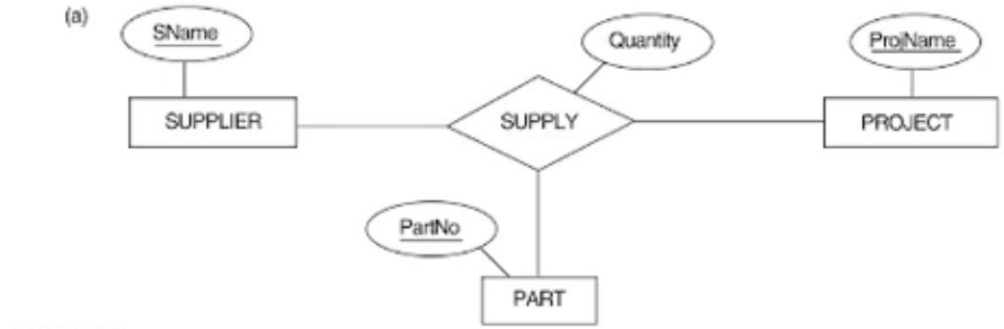
Synthesizing ER Diagram to Relational Model

- Step 7: Mapping of N-ary relationship types
- For each n-ary relationship type R, where n>2, create a new relationship S to represent R
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types
- Also include any simple attributes of the n-ary relationship type as attributes of S

Relational Data Model

Synthesizing ER Diagram to Relational Model

- Step 7: Mapping of N-ary relationship types
- After Step 7



SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

Relational Algebra

Relational Algebra

- Basic set of operations for the relational model is known as the relational algebra
- Operations enable a user to specify basic retrieval requests
- Result of a retrieval is a new relation
- Algebra operations produce a new relations
- Sequence of relational algebra operations forms a relational algebra expression
- Various types of relational algebra operations includes - Unary relational operations, operations from set theory and binary relational operations

Relational Algebra

Relational Algebra

- Provides a formal foundation for relational model operations
- Used as a basis for implementing and optimizing queries
- Some of its concepts are incorporated into the SQL which is the standard querying language of RDBMS
- The core operations and functions in the internal modules of most relational systems are based on relational algebra operations

Relational Algebra

Unary operation - SELECT

- SELECT operation is used to select a subset of the tuples from a relation that satisfy a selection condition
- A filter that keeps only those tuples that satisfy a qualifying condition
- Select operation is denoted by $\sigma_{\text{<selection condition>}}(R)$
- Where σ (sigma) is used to denote the select operator
- Selection condition is a Boolean expression specified on the attributes of relation R

Relational Algebra

Unary operation - SELECT

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

$$\sigma_{(Dno=4\ \mathbf{AND}\ Salary>25000)\ \mathbf{OR}\ (Dno=5\ \mathbf{AND}\ Salary>30000)}(EMPLOYEE)$$

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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

Relational Algebra

Unary operation - PROJECT

- PROJECT operation selects certain columns from the table and discards the other columns
- Creates a vertical partitioning
- To list each employee's first and last name and salary, the following is used $\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$
- The general form of the project operation is π<attribute list>(R)
- Pi is the symbol used to represent the project operation
- <attribute list> is the desired list of attributes from the attributes of relation R

Relational Algebra

Unary operation - PROJECT

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

$\pi_{Lname, Fname, Salary}(EMPLOYEE)$.

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

$\pi_{Sex, Salary}(EMPLOYEE)$.

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Relational Algebra

Unary operation - RENAME

- Apply several relational algebra operations one after the other, and intermediate results are stored in separate relations
- If we want to rename the intermediate relation we use the RENAME operator

$$\text{TEMP} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$R(\text{First\_name}, \text{Last\_name}, \text{Salary}) \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\text{TEMP})$$

- Rename operator is ρ

Relational Algebra

Unary operation - RENAME

TEMP

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
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

R

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

Relational Algebra

Relational operation from set theory

- Basic relational algebraic operation from set theory includes
  - UNION
  - INTERSECTION
  - DIFFERENCE
  - CARTESIAN PRODUCT
- UNION Operation
- Denoted by $R \cup S$
- Results in a relation that includes all tuples that are either in R or in S or in both R and S
- Duplicated tuples are eliminated

Relational Algebra

Relational operation from set theory

- UNION Operation
- For example, to retrieve the social security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$$
$$\text{RESULT1} \leftarrow \pi_{\text{SSN}}(\text{DEP5\_EMPS})$$
$$\text{RESULT2(SSN)} \leftarrow \pi_{\text{SUPERSSN}}(\text{DEP5\_EMPS})$$
$$\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}$$

Relational Algebra

Relational operation from set theory

- UNION Operation
- **Type compatibility** –
- operand relations $R_1(A_1, A_2, \ldots, A_n)$ and $R_2(B_1, B_2, \ldots, B_n)$ must have the same number of attributes
- The domains of corresponding attributes must be compatible; that is **$dom(A_i) = dom(B_i)$** for I = 1, 2, …., n
- The resulting relation for $R_1 \cup R_2$ has the same attribute names as the first operand realtion R_1 by convention

Relational Algebra

Relational operation from set theory

- UNION Operation - Examples

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

$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$

$RESULT1 \leftarrow \pi_{Ssn}(DEP5\_EMPS)$

$RESULT2(Ssn) \leftarrow \pi_{Super\_ssn}(DEP5\_EMPS)$

$RESULT \leftarrow RESULT1 \cup RESULT2$

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra

Relational operation from set theory

- UNION Operation - Examples

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

Relational Algebra

Relational operation from set theory

- INTERSECTION Operation
- Denoted by $R \cap S$
- Result is a relation that includes all tuples that are in both R and S
- Two operands must be “type compatible”

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Fn	Ln
Susan	Yao
Ramesh	Shah

Relational Algebra

Relational operation from set theory

- DIFFERENCE Operation
- Denoted by R – S
- Result is a relation that includes all tuples that are in both R but not in S
- Two operands must be “type compatible”

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

- Student – Instructor

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

Relational Algebra

Relational operation from set theory

- Both union and intersection are commutative

$$R \cup S = S \cup R, \text{ and } R \cap S = S \cap R$$

- Both union and intersection can be treated as n- ary operations applicable to any number of relations as both are associative operations

$$R \cup (S \cup T) = (R \cup S) \cup T, \text{ and } (R \cap S) \cap T = R \cap (S \cap T)$$

- Difference operation is not commutative

Relational Algebra

Relational operation from set theory

- CARTESIAN PRODUCT operation
- Used to combine tuples from two relations in a combinatorial fashion
- $R(A_1, A_2, \ldots, A_n)$ X $S(B_1, B_2, \ldots, B_m)$ is a relation Q with degree n+m attributes
- $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ in that order
- Resulting relation Q has one tuple for each combination of tuples – one from R and one from S
- If R has n_R tuples and S has n_S tuples, then R X S will have $n_R * n_S$ tuples

Relational Algebra

Relational operation from set theory

- CARTESIAN PRODUCT operation - Example

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

$$FEMALE\_EMPS \leftarrow \sigma_{Sex='F'}(EMPLOYEE)$$
$$EMPNAMES \leftarrow \pi_{Fname,\ Lname,\ Ssn}(FEMALE\_EMPS)$$

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

Relational Algebra

Relational operation from set theory

- CARTESIAN PRODUCT operation - Example

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

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

Relational Algebra

Relational operation from set theory

- CARTESIAN PRODUCT operation - Example

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	. . .
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	. . .
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	. . .
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	. . .
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	. . .
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	. . .
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	. . .
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	. . .
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	. . .
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	. . .
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	. . .
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	. . .
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	. . .
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	. . .
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	. . .
Joyce	English	453453453	333445555	Alice	F	1986-04-05	. . .
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	. . .
Joyce	English	453453453	333445555	Joy	F	1958-05-03	. . .
Joyce	English	453453453	987654321	Abner	M	1942-02-28	. . .
Joyce	English	453453453	123456789	Michael	M	1988-01-04	. . .
Joyce	English	453453453	123456789	Alice	F	1988-12-30	. . .
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	. . .

Relational Algebra

Relational operation from set theory

- CARTESIAN PRODUCT operation - Example

$$\text{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{\text{Ssn=Essn}}(\text{EMP\_DEPENDENTS})$$

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

Relational Algebra

Binary Operations

- Binary relational algebraic operations include
  - JOIN
  - EQUIJOIN
  - NATURAL JOIN
  - DIVISION
- JOIN Operations
- Sequence of cartesian product followed by select operation is called JOIN
- Denoted by ⋈
- Allows us to process relationships among relations

Relational Algebra

Binary Operations

- JOIN Operations
- General form of a join operation on two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ is

$$R \bowtie_{\text{<join condition>}} S$$

- Where R and S can be any relations that result from general relational algebra expressions
- Example: Retrieve the name of the manager of each department

Relational Algebra

Binary Operations

- JOIN Operations

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN=SSN}} \text{EMPLOYEE}$$

DEPT_MGR

Dname	Dnumber	Mgr_ssn	...	Fname	Minit	Lname	Ssn	...
Research	5	333445555	...	Franklin	T	Wong	333445555	...
Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
Headquarters	1	888665555	...	James	E	Borg	888665555	...

Relational Algebra

Binary Operations

- JOIN Operations - Example

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

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

Relational Algebra

Binary Operations

- JOIN Operations - Example

$$ACTUAL\_DEPENDENTS \leftarrow EMPNAMES \bowtie_{Ssn=Essn} DEPENDENT$$

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

Relational Algebra

Binary Operations

- Difference between JOIN and CARTESIAN PRODUCT
- Results of the JOIN is a relation Q with n + m attributes
- Q has one tuple for each combination of tuples – one from R and one from S whenever the combination satisfies the join condition
- JOIN – only combinations of tuples satisfying the join condition appear in the result
- CASTESIAN PRODUCT – all combinations of tuples are included in the result

Relational Algebra

Binary Operations

- JOIN Operation
- General JOIN condition is <condition> AND <condition> AND … AND <condition>
- Where each <condition> is of the form $A_i \theta B_j$, Ai is an attribute of R, Bj is an attribute of S, Ai and Bj have the same domain
- The θ theta is one of the comparison operators (=, <, <=, >, >=, !=)
- A JOIN operation with such a general join condition is called a THETA JOIN
- Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result

Relational Algebra

Binary Operations

- EQUIJOIN Operation
- Most common use of EQUIJOIN involves join conditions with equality comparisons only
- EQUIJOIN – only comparison operator used is =
- Result of an EQUIJOIN is that we always have one or more pairs of attributes that have identical values in every tuples

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$

$$ACTUAL\_DEPENDENTS \leftarrow EMPNAMES \bowtie_{Ssn=Essn} DEPENDENT$$

Relational Algebra

Binary Operations

- EQUIJOIN Operation

DEPT_MGR

Dname	Dnumber	Mgr_ssn	...	Fname	Minit	Lname	Ssn	...
Research	5	333445555	...	Franklin	T	Wong	333445555	...
Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
Headquarters	1	888665555	...	James	E	Borg	888665555	...

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

Relational Algebra

Binary Operations

- NATURAL JOIN Operation
- One of each pair of attributes with identical values is superfluous in EQUIJOIN
- NATURAL JOIN is denoted by * was created to get rid of the duplicate attribute in an EQUIJOIN condition
- Standard definition of natural join required that the two join attributes or each pair of corresponding join attributes have the same name in both relations
- If this is not the case, a renaming operation is applied first

Relational Algebra

Binary Operations

- NATURAL JOIN Operation

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation	Dnum

$$PROJ\_DEPT \leftarrow PROJECT * \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

$$DEPT \leftarrow \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

$$PROJ\_DEPT \leftarrow PROJECT * DEPT$$

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

Relational Algebra

Binary Operations

- DIVISION Operation
- Denoted by ÷
- Retrieve the names of employees who work on all the projects that ‘John Smith’ works on

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

Relational Algebra

Binary Operations

- DIVISION Operation

$$SMITH \leftarrow \sigma_{Fname='John' \text{ AND } Lname='Smith'}(EMPLOYEE)$$
$$SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=Ssn} SMITH)$$

SMITH_PNOS

Pno
1
2

$$SSN\_PNOS \leftarrow \pi_{Essn,\ Pno}(WORKS\_ON)$$

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

$$SSNS(Ssn) \leftarrow SSN\_PNOS \div SMITH\_PNOS$$

SSNS

Ssn
123456789
453453453

$$RESULT \leftarrow \pi_{Fname,\ Lname}(SSNS * EMPLOYEE)$$

Structured Query Language(SQL)

Structured Query Language(SQL)

- SQL is Structured Query Language
- It is a computer language for storing, manipulating and retrieving data stored in a relational database
- SQL is the standard language for Relational Database System
- All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language
- SQL is one of the most widely used query language over the databases

Structured Query Language(SQL)

Structured Query Language(SQL) - Features

- Allows users to access data in the relational database management systems
- Allows users to describe the data
- Allows users to define the data in a database and manipulate that data
- Allows to embed within other languages using SQL modules, libraries & pre-compilers
- Allows users to create and drop databases and tables
- Allows users to create view, stored procedure, functions in a database
- Allows users to set permissions on tables, procedures and views

Structured Query Language(SQL)

Structured Query Language(SQL) - Databases

- MySQL is an open source SQL database, which is developed by a Swedish company – MySQL AB
- MySQL is supporting many different platforms including Microsoft Windows, the major Linux distributions, UNIX, and Mac OS X
- MySQL has free and paid versions, depending on its usage (non-commercial/commercial) and features
- MySQL comes with a very fast, multi-threaded, multi-user and robust SQL database server
- MS SQL Server is a Relational Database Management System developed by Microsoft Inc
- Its primary query languages are –T-SQL, ANSI SQL

Structured Query Language(SQL)

Structured Query Language(SQL) - Databases

- Oracle is a very large multi-user based database management system
- Oracle is a relational database management system developed by 'Oracle Corporation'
- Oracle works to efficiently manage its resources, a database of information among the multiple clients requesting and sending data in the network
- MS Access is one of the most popular Microsoft products
- Microsoft Access is an entry-level database management software
- MS Access database is not only inexpensive but also a powerful database for small-scale projects

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- SQL Data Type is an attribute that specifies the type of data of any object
- Each column, variable and expression has a related data type in SQL
- We can use these data types while creating our tables
- We can choose a data type for a table column based on our requirement
- SQL Server offers six categories of data types for your use which are listed below:
  - Exact Numeric Data Types
  - Approximate Numeric Data Types
  - Date and Time Data Types
  - Character Strings Data Types
  - Unicode Character Strings Data Types
  - Binary Data Types
  - Misc Data Types

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Exact Numeric Data types

DATA TYPE	FROM	TO
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 -1
numeric	-10^38 +1	10^38 -1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Approximate Numeric Data Types

DATA TYPE	FROM	TO
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38

- Date and Time Data Types

DATA TYPE	FROM	TO
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079
date	Stores a date like June 30, 1991	
time	Stores a time of day like 12:30 P.M.	

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Character Strings Data Types

Sr.No.	DATA TYPE & Description
1	**char** Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
2	**varchar** Maximum of 8,000 characters.(Variable-length non-Unicode data).
3	**varchar(max)** Maximum length of 2E + 31 characters, Variable-length non-Unicode data (SQL Server 2005 only).
4	**text** Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Unicode Character Strings Data Types

Sr.No.	DATA TYPE & Description
1	**nchar** Maximum length of 4,000 characters.(Fixed length Unicode)
2	**nvarchar** Maximum length of 4,000 characters.(Variable length Unicode)
3	**nvarchar(max)** Maximum length of 2E + 31 characters (SQL Server 2005 only).(Variable length Unicode)
4	**ntext** Maximum length of 1,073,741,823 characters. (Variable length Unicode)

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Binary Data Types

Sr.No.	DATA TYPE & Description
1	**binary** Maximum length of 8,000 bytes(Fixed-length binary data)
2	**varbinary** Maximum length of 8,000 bytes.(Variable length binary data)
3	**varbinary(max)** Maximum length of 2E + 31 bytes (SQL Server 2005 only). (Variable length Binary data)
4	**image** Maximum length of 2,147,483,647 bytes. (Variable length Binary Data)

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Misc Data Types

Sr.No.	DATA TYPE & Description
1	**sql_variant** Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
2	**timestamp** Stores a database-wide unique number that gets updated every time a row gets updated
3	**uniqueidentifier** Stores a globally unique identifier (GUID)
4	**xml** Stores XML data. You can store xml instances in a column or a variable (SQL Server 2005 only).

Structured Query Language(SQL)

Structured Query Language(SQL) – Data types

- Misc Data Types

5	**cursor** Reference to a cursor object
6	**table** Stores a result set for later processing

Structured Query Language(SQL)

Structured Query Language(SQL) – Operators

- SQL contains the following operators:
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Operators used to negate conditions
- Arithmetic Operators
- Assume 'variable a' holds 10 and 'variable b' holds 20, then –

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	a + b will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand.	a - b will give -10
* (Multiplication)	Multiplies values on either side of the operator.	a * b will give 200
/ (Division)	Divides left hand operand by right hand operand.	b / a will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder.	b % a will give 0

Structured Query Language(SQL)

Structured Query Language(SQL) – Operators

- SQL Comparison Operators
- Assume 'variable a' holds 10 and 'variable b' holds 20, then –

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.

Structured Query Language(SQL)

Structured Query Language(SQL) – Operators

- SQL Comparison Operators

>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

Structured Query Language(SQL)

Structured Query Language(SQL) – Operators

- SQL Logical Operators

Sr.No.	Operator & Description
1	**ALL** The ALL operator is used to compare a value to all values in another value set.
2	**AND** The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
3	**ANY** The ANY operator is used to compare a value to any applicable value in the list as per the condition.
4	**BETWEEN** The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.

Structured Query Language(SQL)

Structured Query Language(SQL) – Operators

- SQL Logical Operators

5	**EXISTS** The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion.
6	**IN** The IN operator is used to compare a value to a list of literal values that have been specified.
7	**LIKE** The LIKE operator is used to compare a value to similar values using wildcard operators.
8	**NOT** The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.**

Structured Query Language(SQL)

Structured Query Language(SQL) – Operators

- SQL Logical Operators

9	**OR** The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
10	**IS NULL** The NULL operator is used to compare a value with a NULL value.
11	**UNIQUE** The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

Structured Query Language(SQL)

SQL Commands

- The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP
- These commands can be classified into the following groups based on their nature as:
- Data Definition Language (DDL) – CREATE, ALTER, DROP
- Data Manipulation Language (DML) – SELECT, INSERT, UPDATE, DELETE
- Data Control Language (DCL) – GRANT, REVOKE

Structured Query Language(SQL)

SQL Commands – Data Definition Language (DDL) Commands

- DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema
- It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database
- Examples of DDL commands:
- CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers)
- DROP – is used to delete objects from the database
- ALTER-is used to alter the structure of the database
- TRUNCATE–is used to remove all records from a table, including all spaces allocated for the records are removed
- COMMENT –is used to add comments to the data dictionary
- RENAME –is used to rename an object existing in the database

Structured Query Language(SQL)

SQL Commands – CREATE Command

- The SQL CREATE DATABASE statement is used to create a new SQL database
- The basic syntax of this CREATE DATABASE statement is as follows –

  CREATE DATABASE *DatabaseName*;

- Always the database name should be unique within the RDBMS
- For example:
- If we want to create a new database <testDB>, then the CREATE DATABASE statement would be as shown below –

  SQL> CREATE DATABASE *testDB*;

Structured Query Language(SQL)

SQL Commands – CREATE Command

- The SQL CREATE TABLE statement is used to create a new table
- The basic syntax of the CREATE TABLE statement is as follows −

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

Structured Query Language(SQL)

SQL Commands – CREATE Command

- For example:
- Create a CUSTOMERS table with an ID as a primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table −

```
CREATE TABLE CUSTOMERS(
    ID INT NOT NULL,
    NAME VARCHAR (20) NOT NULL,
    AGE  INT NOT NULL,
    ADDRESS  CHAR (25) ,
    SALARY  DECIMAL (18, 2),
    PRIMARY KEY (ID));
```

Structured Query Language(SQL)

SQL Commands – CREATE Command

- We can verify if our table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use the DESC command as follows

```
SQL> DESC CUSTOMERS;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| ID      | int(11)       | NO   | PRI |         |       |
| NAME    | varchar(20)   | NO   |     |         |       |
| AGE     | int(11)       | NO   |     |         |       |
| ADDRESS | char(25)      | YES  |     | NULL    |       |
| SALARY  | decimal(18,2) | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Structured Query Language(SQL)

SQL Commands – DROP Command

- The SQL DROP DATABASE statement is used to drop an existing database in SQL schema
- The basic syntax of DROP DATABASE statement is as follows −
- DROP DATABASE *DatabaseName*;
- Always the database name should be unique within the RDBMS
- For example, if we want to delete an existing database <testDB>, then the DROP DATABASE statement would be as shown below −

```
DROP DATABASE testDB;
```

- NOTE − Be careful before using this operation because by deleting an existing database would result in loss of complete information stored in the database

Structured Query Language(SQL)

SQL Commands – DROP Command

- The SQL DROP TABLE statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table
- The basic syntax of this DROP TABLE statement is as follows –
- DROP TABLE table_name;
- For example, if we want to drop the CUSTOMERS table from the database, we use the query

```
DROP TABLE CUSTOMERS;
```

Structured Query Language(SQL)

SQL Commands – ALTER Command

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table
- To add a new column to table we use the query:

  ALTER TABLE *table_name*

  ADD *column_name* datatype;

- For example:

  ALTER TABLE CUSTOMERS

  ADD Email varchar(255);

Structured Query Language(SQL)

SQL Commands – ALTER Command

- To delete a column in a table, we use the query:

  ALTER TABLE *table_name*

  DROP COLUMN *column_name*;

- For example:

  ALTER TABLE Customers

  DROP COLUMN Email;

Structured Query Language(SQL)

SQL Commands – INSERT Command

- The SQL INSERT INTO Statement is used to add new rows of data to a table in the database
- There are two basic syntaxes of the INSERT INTO statement which are shown below:

  INSERT INTO *TABLE_NAME* (*column1, column2, column3,...columnN*)

  VALUES (*value1, value2, value3,...valueN*);

- Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data
- We may not need to specify the column(s) name in the SQL query if we are adding values for all the columns of the table
- But make sure the order of the values is in the same order as the columns in the table

Structured Query Language(SQL)

SQL Commands – INSERT Command

- The other way of using INSERT TO is as follows:

  INSERT INTO *TABLE_NAME* VALUES (*value1,value2,value3,...valueN*);

- The following statements would create six records in the CUSTOMERS table.

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

Structured Query Language(SQL)

SQL Commands – INSERT Command

- We can create a record in the CUSTOMERS table by using the second syntax as shown below:
- INSERT INTO CUSTOMERS
- VALUES (7, 'Muffy', 24, 'Indore', 10000.00);
- All the above statements would produce the following records in the CUSTOMERS table as shown below

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Structured Query Language(SQL)

SQL Commands – SELECT Command

- The SQL SELECT statement is used to fetch the data from a database table which returns this data in the form of a result table
- These result tables are called result-sets
- The basic syntax of the SELECT statement is as follows –

  SELECT *column1, column2, columnN* FROM *table_name*;

- Here, column1, column2... are the fields of a table whose values we want to fetch
- If we want to fetch all the fields available in the field, then we can use the following syntax

  SELECT * FROM *table_name*;

Structured Query Language(SQL)

SQL Commands – SELECT Command

- Consider the CUSTOMERS table having the following records

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

- The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table

```
SELECT ID, NAME, SALARY FROM CUSTOMERS;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This will produce the following results as:

ID	NAME	SALARY
1	Ramesh	2000.00
2	Khilan	1500.00
3	kaushik	2000.00
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

- If we want to fetch all the fields of the CUSTOMERS table, then we should use the following query

```
SELECT * FROM CUSTOMERS;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This will produce the following results as:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Structured Query Language(SQL)

SQL Commands – SELECT Command

- The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables
- If the given condition is satisfied, then only it returns a specific value from the table.
- We should use the WHERE clause to filter the records and fetching only the necessary records
- The basic syntax of the SELECT statement with the WHERE clause is as shown below

  SELECT *column1, column2, columnN*

  FROM *table_name*

  WHERE [*condition*]

Structured Query Language(SQL)

SQL Commands – SELECT Command

- The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables
- If the given condition is satisfied, then only it returns a specific value from the table.
- We should use the WHERE clause to filter the records and fetching only the necessary records
- The basic syntax of the SELECT statement with the WHERE clause is as shown below

  SELECT *column1, column2, columnN*

  FROM *table_name*

  WHERE [*condition*]

Structured Query Language(SQL)

SQL Commands – SELECT Command

- Consider the example:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

- The following code is an example which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 −

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This would produce the following result −

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  4 | Chaitali |  6500.00 |
|  5 | Hardik   |  8500.00 |
|  6 | Komal    |  4500.00 |
|  7 | Muffy    | 10000.00 |
+----+----------+----------+
```

- The following query is an example, which would fetch the ID, Name and Salary fields from the CUSTOMERS table for a customer with the name *Hardik*
- Here, it is important to note that all the strings should be given inside single quotes (‘’), whereas, numeric values should be given without any quote as in the above example

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME = 'Hardik';
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This would produce the following result −

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  5 | Hardik   |  8500.00 |
+----+----------+----------+
```

- The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc
- The SQL AND & OR operators are used to combine multiple conditions to narrow data in an SQL statement
- These two operators are called as the conjunctive operators
- These operators provide a means to make multiple comparisons with different operators in the same SQL statement

Structured Query Language(SQL)

SQL Commands – SELECT Command

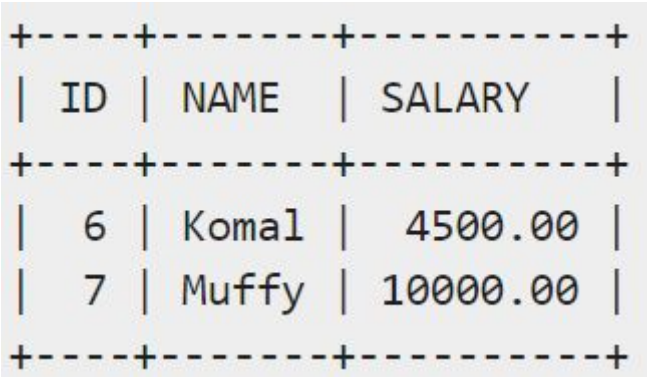
- The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause
- The basic syntax of the AND operator with a WHERE clause is as follows −
- SELECT column1, column2, columnN
- FROM table_name
- WHERE [condition1] AND [condition2]...AND [conditionN];
- Following is an example, which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 and the age is less than 25 years −

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 AND age < 25;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This would produce the following result −



```
+----+-------+----------+
| ID | NAME  | SALARY   |
+----+-------+----------+
|  6 | Komal |  4500.00 |
|  7 | Muffy | 10000.00 |
+----+-------+----------+
```

- The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause
- The basic syntax of the OR operator with a WHERE clause is as follows −

  SELECT *column1, column2, columnN*

  FROM *table_name*

  WHERE [*condition1*] OR [*condition2*]...OR [*conditionN*]

- Only any ONE of the conditions separated by the OR must be TRUE

Structured Query Language(SQL)

SQL Commands – SELECT Command

- The following query would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 OR the age is less than 25 years

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 OR age < 25;
```

- This would produce the following result –

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  3 | kaushik  |  2000.00 |
|  4 | Chaitali |  6500.00 |
|  5 | Hardik   |  8500.00 |
|  6 | Komal    |  4500.00 |
|  7 | Muffy    | 10000.00 |
+----+----------+----------+
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- The SQL LIKE clause is used to compare a value to similar values using wildcard operators
- There are two wildcards used in conjunction with the LIKE operator
  - The percent sign (%)
  - The underscore (_)
- The percent sign represents zero, one or multiple characters
- The underscore represents a single number or character
- These symbols can be used in combinations
- The basic syntax of % and _ is as follows −

```
SELECT FROM table_name
WHERE column LIKE 'XXXX%’
    or
SELECT FROM table_name
WHERE column LIKE '%XXXX%'
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

or

SELECT FROM table_name

WHERE column LIKE 'XXXX_’

or

SELECT FROM table_name

WHERE column LIKE '_XXXX’

or

SELECT FROM table_name

WHERE column LIKE '_XXXX_’

- We can combine N number of conditions using AND or OR operators
- Here, XXXX could be any numeric or string value

Structured Query Language(SQL)

SQL Commands – SELECT Command

- For example:

WHERE SALARY LIKE '200%'

Finds any values that start with 200.

WHERE SALARY LIKE '%200%'

Finds any values that have 200 in any position.

WHERE SALARY LIKE '_00%'

Finds any values that have 00 in the second and third positions.

Structured Query Language(SQL)

SQL Commands – SELECT Command

- For example:

WHERE SALARY LIKE '2_%_%' Finds any values that start with 2 and are at least 3 characters in length.
WHERE SALARY LIKE '%2' Finds any values that end with 2.
WHERE SALARY LIKE '_2%3' Finds any values that have a 2 in the second position and end with a 3.
WHERE SALARY LIKE '2___3' Finds any values in a five-digit number that start with 2 and end with 3.

Structured Query Language(SQL)

SQL Commands – SELECT Command

- Following is an example, which would display all the records from the CUSTOMERS table, where the SALARY starts with 200

```
SQL> SELECT * FROM CUSTOMERS
WHERE SALARY LIKE '200%';
```

This would produce the following result −

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
+----+----------+-----+-----------+----------+
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns
- Some databases sort the query results in an ascending order by default
- The basic syntax of the ORDER BY clause is as follows –

SELECT *column-list*

FROM *table_name*

[WHERE *condition*]

[ORDER BY *column1, column2, .. columnN*] [ASC | DESC];

Structured Query Language(SQL)

SQL Commands – SELECT Command

- Consider the CUSTOMERS table having the following records −

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

- The following code block has an example, which would sort the result in an ascending order by the NAME and the SALARY −

```
SQL> SELECT * FROM CUSTOMERS
ORDER BY NAME, SALARY;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This would produce the following result −

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
3	kaushik	23	Kota	2000.00
2	Khilan	25	Delhi	1500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
1	Ramesh	32	Ahmedabad	2000.00

- The following code block has an example, which would sort the result in an ascending order by the NAME −

```
SQL> SELECT * FROM CUSTOMERS
ORDER BY NAME ASC;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This would produce the following result −

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
3	kaushik	23	Kota	2000.00
2	Khilan	25	Delhi	1500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
1	Ramesh	32	Ahmedabad	2000.00

- The following code block has an example, which would sort the result in the descending order by NAME

```
SQL> SELECT * FROM CUSTOMERS
ORDER BY NAME DESC;
```

Structured Query Language(SQL)

SQL Commands – SELECT Command

- This would produce the following result −

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
7	Muffy	24	Indore	10000.00
6	Komal	22	MP	4500.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
4	Chaitali	25	Mumbai	6500.00

Structured Query Language(SQL)

SQL Commands – UPDATE Command

- The SQL UPDATE Query is used to modify the existing records in a table
- We can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected
- The basic syntax of the UPDATE query with a WHERE clause is as follows –

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

Structured Query Language(SQL)

SQL Commands – UPDATE Command

- Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

- The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune’
WHERE ID = 6;
```

Structured Query Language(SQL)

SQL Commands – UPDATE Command

- Now, the CUSTOMERS table would have the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00

- If we want to modify all the ADDRESS and the SALARY column values in the CUSTOMERS table, we do not need to use the WHERE clause as the UPDATE query would be enough as shown in the following code block

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune', SALARY = 1000.00;
```

Structured Query Language(SQL)

SQL Commands – UPDATE Command

- Now, CUSTOMERS table would have the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Pune	1000.00
2	Khilan	25	Pune	1000.00
3	kaushik	23	Pune	1000.00
4	Chaitali	25	Pune	1000.00
5	Hardik	27	Pune	1000.00
6	Komal	22	Pune	1000.00
7	Muffy	24	Pune	1000.00

Structured Query Language(SQL)

SQL Commands – DELETE Command

- The SQL DELETE Query is used to delete the existing records from a table
- You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted
- The basic syntax of the DELETE query with the WHERE clause is as follows

```
DELETE FROM table_name
WHERE [condition];
```

Structured Query Language(SQL)

SQL Commands – DELETE Command

- Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

- The following code has a query, which will DELETE a customer, whose ID is 6

```
SQL> DELETE FROM CUSTOMERS
WHERE ID = 6;
```

Structured Query Language(SQL)

SQL Commands – DELETE Command

- Now, the CUSTOMERS table would have the following records

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

- If we want to DELETE all the records from the CUSTOMERS table, we do not need to use the WHERE clause and the DELETE query would be as follows :

```
SQL> DELETE FROM CUSTOMERS;
```

- Now, the CUSTOMERS table would not have any record