

CHAPTER FIVE

Structured Query Language(SQL)

Cont...

Outline

- ▶ Introduction
- ▶ SQL Statements
 - ▶ Data Definition Language
 - ▶ Data Manipulation Language

Introduction

- ▮ SQL is the most widely implemented language for relational databases.
- ▮ SQL not only allows you to manage data within the database, but also manage the database itself.
- ▮ By using SQL statements, you can access an SQL database directly by using an interactive client application or through an application programming language or scripting language.
- ▮ Regardless of which method you use to access a data source, you should have a foundation in how to write SQL statements that allows you to access relational data base.

Cont...

- ▢ Structured Query Language (SQL) supports the creation and maintenance of the relational database and the management of data within that database.
- ▢ At the core of the relational model is the relation. A relation is a set of columns and rows collected in a table-like structure that represents a single entity made up of related data.
- ▢ Each relation comprises one or more attributes (columns). An attribute groups similar types of data together
- ▢ Each attribute has an associated domain. A domain defines the type of data that can be

Cont...

- ▮ A **data type** is a specific kind of constraint associated with a column, whereas a domain, as it is used in the relational model, has a much broader meaning and describes exactly what data can be included in an attribute associated with that domain.
- ▮ Data is stored in a relation in tuples (rows).
 - ▮ A tuple is a set of data whose values make up an instance of each attribute defined for that relation.
 - ▮ Each tuple represents a record of related data.
- ▮ The terms **relation**, **attribute**, and **tuple** are used primarily when referring to the relational model.

Cont...

- ▮ **SQL** uses the terms **table**, **column**, and **row** to describe these items.
- ▮ Because the **relational model** is based on **mathematical principles** and **SQL** is concerned more with the **physical implementation** of the model, the meanings for the model's terms and the SQL language's terms are slightly different, but the underlying principles are the same.
- ▮ **Two** very important considerations in the **design** and **implementation** of any relational database are the **normalization** of data and the **associations** of relations among the various types of data.

Cont...

- ▮ SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.
- ▮ SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL standard database language.
- ▮ **Why SQL?**
 - ▮ Allows users to access data in relational database management systems.
 - ▮ Allows users to describe the data.

Cont...

- ▮ Allows users to define the data in database and manipulate that data.
- ▮ Allows embedding 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

Cont...

▮ **SQL** uses the terms **table**, **column**, and **row** to describe the items relation, attribute, and tuple respectively.

▮ **Table:** The data in RDBMS is stored in database objects called tables. The table is **a** collection of related entries and it consists of columns and rows. Remember, a table is the most common and simplest form of data storage in a relational database.

▮ **Field:** Every table is broken up into smaller entities called fields. A field is a column in a table that is designed to maintain specific information about every record table.

▮ **Example:** The fields in the CUSTOMERS table consists ID, NAME, AGE, ADDRESS and SALARY.

Cont...

- ▮ **Record or row:** A record, also called a row of data, is each individual entry that exists in a table. For example: there records in the above CUSTOMERS table. A record is a horizontal entity in a table.
- ▮ **Column:** A column is a vertical entity in a table that contains all information associated with a specific field table.
- ▮ **NULL value:** A NULL value in a table is a value in a field that appears to be blank which means a field with a value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation.

Cont...

▮ **SQL Constraints:** Constraints are the rules enforced on data columns on table. These are used to limit the type of data can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be column level or table level. Column level constraints are applied only to one column where as table level constraints are applied to the whole table. Commonly used constraint variables in SQL are:

- ▮ **NOT NULL Constraint:** ensures that a column cannot have null value.
- ▮ **UNIQUE Constraint:** ensures that all values in a column are different.
- ▮ **PRIMARY KEY:** uniquely identify each rows/records in a database table.

Cont...

- ▮ **FOREIGN KEY:** is a key used to link two or more tables together. It is sometimes called a **referencing key**.
- ▮ **CHECK constraint:** ensures that all values in a column satisfy certain conditions.
- ▮ **Data Integrity:** The following categories of the data integrity exist with each other relational DBMS.
 - ▮ **Entity Integrity:** enforces no duplication rows in a table.
 - ▮ **Domain Integrity:** enforces valid entries for a given column by restricting the type, the format, or the range of values.
 - ▮ **Referential Integrity:** rows cannot be deleted which are used by other records.
 - ▮ **User-define Integrity:** enforces some business rules that do not fall in to entity, domain, or referential integrity.

Cont...

▮ **Domain Types in SQL:** The SQL standard supports a variety of built-in domain types, including:

- ▮ **Char(n):** A fixed-length character string with user-specified length n.
- ▮ **Varchar(n):** A variable-length character string with user-specified maximum length n.
- ▮ **Int:** An integer (a finite subset of the integers that is machine dependent).
- ▮ **Smallint:** A small integer (a machine-dependent subset of the integer domain type).

Cont...

- ▮ **Numeric(p, d):** A fixed-point number with user-specified precision. The number consists of p digits (plus a sign), and d of the p digits are to the right of the decimal point. Thus, **numeric(3,1)** allows **44.5 to be stored exactly, but neither 444.5 or 0.32 can be stored exactly in a field of this type.**
- ▮ **Real, double precision:** Floating-point and double-precision floating-point numbers with machine-dependent precision.
- ▮ **Float(n):** A floating-point number, with precision of at least n digits.
- ▮ **Date:** A calendar date containing a (four-digit) year, month, and day of the month.

Cont...

▮ **Time**: The time of day, in hours, minutes, and seconds. A variant, **time(*p*)**, *can* be used to specify the number of fractional digits for seconds (the default being 0). It is also possible to store time zone information along with the time.

▮ **Timestamp**: A combination of date and time. A variant, **timestamp(*p*)**, *can be* used to specify the number of fractional digits for seconds (the default here being 6).

❖ Date and time values can be specified like this:

❖ **Date '2001-04-25'**

❖ **Time '09:30:00'**

❖ **Timestamp '2001-04-25 10:29:01.45'**

SQL Statement

- ▮ **SQL Syntax:** SQL is followed by unique set of rules and guidelines called **Syntax**.
- ▮ All the SQL statements start with any of the keywords like ***SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW*** and all the statements end with a semicolon (;).
- ▮ Important point to be noted is that **SQL is case insensitive which means SELECT is equal to select in SQL statements.**

Cont...

SQL language has two main components

- Data Definition Language (DDL)

- Data Manipulation Language (DML)

□ Data Definition Language (DDL)

- ▶ It provides commands for defining databases and relation schemas, deleting relations and modifying relation schemas within the data base.
- ▶ It has the following keywords
 - ▶ **CREATE**: is used to create a database and a database objects.

Cont...

►To create a database:

- **Syntax:** **CREATE database 'database Name';**
Where Database Name is a name of the new name.
- Example: create a bank database with name xy.
 - **CREATE database xy;**

►To create a relation or a table

- **Syntax:** **CREATE table 'table Name'(A1D1, A2D2
..., An Dn,
[Integrity constrain1]...[integrity constrain m]);**

Where A_i is the name of an attribute in the relation
and D_i is the domain type of values in the domain of
attribute A_i .

Cont...

► There can be of a number of different allowable integrity constraints, which is optional.

► **Example:** if we like to define a key constrain,

► PRIMARY KEY(Ai1, Ai2..., Ai n) Where Ai's are attributes that are combined to form a key.

► **Ex:** Create the table customer in the xy database?

► **Answer:**

```
CREATE table customer  
( CustId char(10),  
  CustName varchar(20),  
  custCity varchar(20)  
  PRIMARY KEY (CustId));
```

Cont...

Drop:

- Is used to delete existing database, table or constraints.
- Syntax: DROP [Database/Table/Constraints] name; where name is a name of the object going to be deleted.
- **Example:** Delete customer table from the xy database?
 - DROP table customer;

Cont...

▮ **ALTER:** is used to modify the definition of the existing relation schema.

▮ **Syntax:** ALTER table name

ADD COLUMN 'column name' data type,...

|ADD CONSTRAINT 'constraint name' constraint type,...

Or

DROP COLUMN 'column name',...|DROP CONSTRAINT 'constraint name',...

Where table name is name of the table, column name and constraint name represents name of the column and name of the constraint that we want to modify respectively.

Cont...

□ ***Example:*** Add column AccountNumber on customer table, and define a referential integrity constraint for it.

□ ***Answer:***

```
ALTER customer  
ADD COLUMN AccountNumber char(10),  
ADD CONSTRAINT AccountNumber FOREIGN  
KEY(customer) REFERENCES account  
(AccountNumber);
```

Cont...

▮ Data Manipulation Language(DML)

- ▮ It includes a query language based on both the relational algebra and the tuple relational calculus.
- ▮ It includes commands to **insert tuples into**, **delete tuples from** and **modify tuples** in the database.
- ▮ It may be has two types
 - ▮ Data updating language and
 - ▮ Data retrieval language

Cont...

A. **Data Updating Language:** SQL DML includes three clauses that can be used for updating the content of relations.

□ The three operations that are required to data updating are **Insertion**, **Deletion** and **Updating**.

□ **Insertion:** it is a SQL clause used to add a number of tuples into a specified relation.

□ **Syntax:** **INSERT INTO** 'tablename'[(column names separated by comma)] **VALUES** (values separated by comma and with in a single quote(')(if not type of numbers))

Cont...

Example: insert the following records in to customer table?

■(cust01, selam, Ambo)

```
INSERT INTO customer
```

```
VALUES('cust01', 'selam', 'Ambo');
```

■(cust02, abebe, addis abeba)

```
INSERT INTO customer(custId, custName,  
custCity) VALUES('cust02', 'abebe', 'addis abeba');
```

Cont...

▮ **Deletion:** it is a clause used to delete a specified tuples from a relation that meets specified criteria.

▮ **Syntax:** DELETE column names
 FROM table name
 WHERE (condition)

Example: DELETE *
 FROM customer
 WHERE custId='cust01';

Cont...

▮ **Update:** is a clause used to modify an existing by new value of a tuple that satisfies a given criteria.

▮ **Syntax:** UPDATE 'table name'

SET column1 = value1, column2=value2 ...

WHERE (condition)

Where table name is a relation, and condition is a criteria based on which a deletion of tuples effected.

Example: UPDATE customer

SET custname ='belay'

WHERE custId='cust02' ;

Cont...

Exercise: Group work

1. Create a data base with a name 'test' ?
2. Create a table 'student' on the data base 'test' with columns 'studId', StudName, Age, Sex, and DoB?
3. Insert the following records in to student table?
(CEP/1001/07, Selam, 20,F,12/05/1987)
(CEP/1002/07, Abebe, 22,M,05/08/1985)
(CEP/1003/07, XY, 25,M, 01/04/1982)
4. Delete the student recode with studId = "CEP/1003/07"?
5. Modify age and DOB of the student by '23' and '06/02/1984' respectively Where studId='CEP/1002' ?

Cont...

B. Data Retrieval Language : in SQL statement , data retrieval has SELECT, FROM, and WHERE key languages.

❖ **Syntax:** SELECT 'List of columns'
FROM 'table name'
WHERE [condition]

- **SELECT clause:** it is used to list the attributes desired in the result of a query.
 - **DISTINCT:** this clause is used to retrieve only one of the records that contain duplicate data in the selected fields .

Cont...

- **All:** it is a clause that used to list all in the result relation.
- **FROM:** is a clause used to list the relation(table) to be scanned in the evaluation of the expression.
- Example: find all customer information from customer table?

```
SELECT *
```

```
FROM customer
```

2. Find name of all customer ?

```
SELECT custName
```

```
FROM customer
```

Cont...

3. Find name of customers from the customer table

and remove duplicate tuples from the result?

```
SELECT DISTINCT custName
```

```
FROM customer
```

■ ***WHERE clause:*** it is an optional clause, which is used to specify conditions to select some records .

■ **Example: SELECT ***

```
FROM customer
```

```
WHERE custId="cust01" ;
```

Cont...

LIKE: it is an operator that goes with the WHERE clause and used to compare a string expression to a pattern.

- It may be used with wild card (%) for representation for multiple characters in the expression.

NULL and not NULL: SQL supports the use of null value to indicate absence of information as the value of an attribute.

- Null and not Null are operations that can works with WHERE clause and are used as a representation for unknown (known) value or to check with null(not null) values respectively.

Cont...

Example-1: Distinct list of all customer's name whose name starts with a letter A?

```
SELECT DISTINCT custName  
FROM customer  
WHERE custName LIKE 'A%' ;
```

• **Example-2:** Find customers that do not have telephone information?

```
SELECT *  
FROM customer  
WHERE custTelephone is NULL;
```

Cont...

Exercise:

- ❖ Create a data base with name 'Registrar'?
- ❖ Create the following three tables
 - ▶ Student(Id, Fname, Lname, age, sex, DOB)
 - ▶ Course (code, Title, credit hour)' and
 - ▶ Enroll (Id, code, semester, academic year, GPA)
- ❖ Add column '**prerequisite course**' in course table?
- ❖ Insert the following records in student, course and enroll tables respectively?
 - ▶ (R/101/07, AA, BB, 21, m, 01/04/1987)
 - ▶ (Comp101, data base system, 4, Introduction)
 - ▶ (R/101/07, comp101, , 2007, 3.7)

Cont...

- ▶ Find all students who registered the course data base system ?
- ▶ Find all student's name whose name starts with a letter **s**?
- ▶ Retrieve name of a student whose Id = 'R/101/07'?

Cont...

▮ *Aggregate Functions (Group Functions)*

- Aggregate functions return a single value, calculated from values in a column.
- Types of group functions
 - ❖ AVG, MIN, SUM, COUNT, MAX, etc.
- **AVG ()** - returns the average value of a numeric column.
 - **Syntax:** SELECT AVG (column_name)
FROM <table_name >

Cont...

EmpId	Name	DOB	Job	Salary	Commission	Depno
1001	CLARK	09/02/1952	Manager	2750	275	10
1002	Scott	09/02/1976	Analyst	5000	500	20
1103	King	09/02/1972	Salesman	3000	300	10
1104	Jones	09/03/1985	Salesman	4500	450	30

Employee table:

- *Example: SELECT AVG(salary) as average_salary FROM Employee;*

- **Answer:**

Average Salary

3812.5

Cont...

- ❖ **MAX ()** - Returns the largest value
 - **Syntax:** *SELECT MAX (COLUMN NAME) FROM TABLE_NAME;*
 - **Example:** *Select MAX (Salary) as maximum_salary from Employee;*

◦ **Answer:**

Maximum Salary

5000

- ❖ **MIN ()** - Returns the smallest value
 - **Syntax:** *SELECT MIN (COLUMN_NAME) FROM TABLE_NAME;*
 - **Example:** *Select MIN (salary) as minimum_salary from Employee;* **Answer:**

Minimum Salary

2750

Cont...

❖ *Using the MIN and MAX Functions together*

◦ **Example:** select MAX (salary) as maximum_salary, MIN (salary) as minimum_salary from Employee;

◦ Answer:

Maximum Salary	Minimum Salary
5000	2750

❖ **SUM ()** - Returns the sum of the values.

◦ SELECT SUM (COLUMN_NAME) FROM Employee;

◦ **Example:** Select sum (salary) as SUM from Employee;

◦ Answer:

Sum
15250

Cont...

❖ **COUNT ()** - Returns the number of rows

◦ **Syntax:** Select count (*) from <table name> - returns the whole rows

◦ **Example:** *Select count (*) from Employee;*

◦ **Answer:**

COUNT(*)
4

❖ **COUNT (expression):** returns the number of rows with non-null values for expression

◦ **Example:** *SELECT count (salary) as no_of_employee FROM Employee where deptno=10;*

◦ **Answer:**

No_of_employee
2

Cont...

❖ Using the DISTINCT Keyword

- ❖ **COUNT** (DISTINCT expression) returns the number of distinct non-null values of *expression*.
- ❖ To display the number of distinct department values in the Employee table:
- ❖ **Example:** *select count (distinct deptno) from Employee;*

**COUNT(distinct
deptno)**

3

❖ Group Functions and Null Values

- ❖ Group functions ignore null values in the column:
- ❖ **Example:** *select AVG (salary) from Employee;*

Cont...

Using the all at one time Functions

Example: *SELECT MAX (Salary), MIN (Salary), SUM (salary), AVG (Salary) FROM EMPLOYEE;*

Answer:

MAX	MIN	SUM	AVG
5000	2750	15250	3812.5

Creating Groups of Data:

- GROUP BY Clause:**-it collects the rows into groups based on column values in the grouping columns.
- Operates on the rows from the FROM clause as filtered by the WHERE clause.

Cont...

▮ *Syntax: Select <group by columns or aggregate function>*

FROM <table name>

WHERE <condition>

GROUP BY <group by columns>

▮ *Example: SELECT Depno, SUM(Salary)*

FROM Employee

▪ **Answer:**

Depno	SUM(Salary)
10	5750
20	5000
30	4500

Cont...

□ **Having Clause:**

- It is an optional clause that sets up conditions that select only some groups among those created using a group by clause.
- Expressions in having clause must be single valued and it can not be specified if group by clause is not used.
- **Example:** Display a list of branch names that has more than two customers from the ff. schema?
 - **Customer**(CustId, Name, City, e-mail)
 - **Account**(Accno, branchName, balance)
 - **Branch**(branchName, branchCity, Amount)
 - **Loan**(LoanNum, branchName, Amount)
 - **Borrower**(CustId, LoanNum)
 - **Dipositor**(CustId, Accno)

Cont...

❖ **Answer:** `SELECT branchName`

`FROM Account, Depositor`

`WHERE Account.Accno = Depositor.Accno`

`GROUP BY branchName`

`HAVING (count(custId)>2)`

□ **ORDER BY clause:**

- It helps to appear the result query in sorted order.
- By default the order by clause lists in ascending order, but the order can be specified using DESC for descending order or ASC for ascending order .

Cont...

❖ **Example:** List customer information in descending order based on customer name?

❖ **Answer:** SELECT *

FROM Customer

ORDER BY Name DESC;

□ **Ambiguous Attribute Names and Aliasing**

- In SQL the same name can be used for two (or more) attributes as long as the attributes are in *different relations*.
- If this is the case, and a query refers to two or more attributes with the same name, we must qualify the attribute name with the relation name to prevent ambiguity.
- This is done by **prefixing the relation name to the attribute name and separating the two by a period**.

Cont...

Example-1:

- SELECT FName, EMPLOYEE. Name, Address
- FROM EMPLOYEE,DEPARTMENT
- WHERE DEPARTMENT. Name='Research' AND DEPARTMENT. DNumber=EMPLOYEE. DNumber;

Example-2:

- **SELECT** E.FName, E.LName, D.FNAME, D.Lname
- **FROM** Employee **AS E, Department AS D**
- **WHERE** E.Dnum=D.Dnum;
 - In this case, we are allowed to declare alternative relation names E and D, called **aliases** or **tuple variables**, for the Employee and Department relation. An **alias** can follow the keyword **AS**

Cont...

👉 **Example-3:** Retrieve all employees in department 5 whose salary is between \$30,000 and \$40,000?

👉 **Answer:**

*SELECT **

FROM EMPLOYEE

*WHERE (SALARY **BETWEEN** 30000 **AND** 40000)*

AND DNO =5;

❖ The condition (SALARY BETWEEN 30000 AND 40000) in example-3 is equivalent to the condition ((SALARY $\geq$ 30000) AND (SALARY $\leq$ 40000)).

Cont...

❖ **Example-4:** Retrieve the social security numbers of all employees who work on project numbers 1,2, or 3.

❖ **Answer:** ***SELECT*** DISTINCT ESSN

FROM WORKS_ON

WHERE PNO IN (1, 2, 3);

❖ In SQL, it is possible to rename any attribute that appears in the result of a query by adding the qualifier **AS** followed by the desired new name.

❖ **Example:**

SELECT E.LNAME **AS** EMPLOYEE_NAME,
S.LNAME **AS** SUPERVISOR_NAME

FROM EMPLOYEE **AS** E, SUPERVISING **AS** S

WHERE E.SUPERSSN=S.SSN;

Cont...

▮ **Nested Queries:** a query that is part of another query is called a sub-query(nested query).

▮ It appears in the where clause .

▮ A common use of sub-queries is to perform tests for set membership, make set comparison , and determine set cardinality.

▮ **Example:**

Select s.NAME, s.Id

From student as s

Where s.Id IN(select Id from enroll where semester ="I")

Cont...

▮ **Example-2:** Find all customer names that do have an account in bank , but do not have a loan from a bank?

▮ **Answer:**

```
SELECT custName
FROM customer
WHERE customer.custId NOT IN
      (SELECT borrower.custId
       FROM borrower
       WHERE customer.custId = borrower.custId);
```

Cont...

Example 2: Retrieve an alphabetical list of last names and IDs of all students in any class taught by faculty number 'F110'.

```
SELECT LNAME, STUID
FROM STUDENT
WHERE STUID IN
(SELECT STUID
FROM ENROLL
WHERE COURSENUM IN
(SELECT COURSENUM
FROM CLASS
WHERE FACID = 'F110'))
ORDER BY LNAME;
```

Cont...

Example 3: Find the name and IDs of students who have less than the average number of credits.

```
SELECT LNAME, STUID
FROM STUDENT, COURSE
WHERE CREDITS <
      (SELECT AVG(CREDITS)
       FROM COURSE);
```

Cont...

Join:

- The ability of using two or more relations in the query set up is one of the most powerful feature of relational database system.
- The join operations take two relations and return another relation as a result.
- Each of the join operations in SQL consists of a join type and a join condition.
- The join condition defines which tuples in the two relations match and what attributes are present in the result, where as the join type define how tuples in each relation that do not match any tuple in the other relation are treated.

Cont...

Example: Using inner join operation

- List full information about a customer who have a bank account?

```
SELECT *
```

```
FROM (customer INNER JOIN (Depositor INNER  
JOIN Account ON Deposite.AccNum =  
Account.AccNum) ON customer.custId =  
Depositor.custId)
```

```
ORDER BY customer.custId;
```

- List customer Id and total balance of a customer whose total balance of their deposite from any branch greater than 30000?

Cont...

- `SELECT` _____ `customer.custId,`
`sum(Account.balance)`
- `FROM (customer INNER JOIN(Depositor`
`INNER JOIN Account ON Depositor.AccNum`
`= Account.AccNum) ON customer.custId =`
`Depositor.custId)`
- `GROUP BY customer.custId`
- `HAVING (SUM(Account.balance)>30000)`
- `ORDER BY customer.custId`
- List customer Id who have a balance greater than all customer at “Ambo” branch?

Cont...

- SELECT custId
- FROM Account A INNER JOIN Depositor D
ON A.AccNum=D.AccNum
- WHERE balance >All
(SELECT balance
FROM Account A INNER JOIN
Depositor D ON
A.AccNum=D.AccNum
WHERE BranchName =“Ambo”);

VIEW

- ▮ A view in SQL is a single relation that is derived from other relations
- ▮ View are used when there is a need to retrieve data from a relation or reference a relation that may not necessarily exist physically.
- ▮ We may need data from two or more different relations , So instead of continually specifying a join , it is better to use view.
- ▮ **Syntax:** CREATE VIEW 'View name'[field names]
AS query expression

Cont...

■ Where:

- 'View name' is a name of the new relation,
- Field name which is optional, used to specify field names that may appear in the new relation, and
- query expression is any valid a select query expression.
- **Example:** Create a view with a name BorrowerCustomer that contains customers information which have a loan at bank?

```
CREATE VIEW BorrowerCustomer
```

```
AS (SELECT customer *
```

```
FROM customer INNER JOIN Borrower ON  
customer.custId = Borrower.custId);
```

Relational Query Language

- ▮ **Query Language:** allows manipulation and retrieval of data from a data base.
 - ▮ Query language is not equal to programming language.
 - ▮ It is not intended to be used for complex calculations.
 - ▮ It supports easy, efficient access to large data sets.
 - ▮ Relational model supports simple, powerful query language.
- ▮ Two mathematical query languages form the basis for relational language.
 - ▮ Relational Algebra
 - ▮ Relational Calculus

Cont...

▮ **Relational Algebra:** The basic set of operations for the relational model is known as the relational algebra.

- ▮ These operations enable a user to specify basic retrieval requests.
- ▮ A sequence of relational algebra operations forms a relational algebra expressions, whose result will also be a relation that represents the result of a database query.
- ▮ Relational algebra is a theoretical language with operations that work on one or more relations to define another relation without changing the original relation.

Cont...

- The output from one operation can become the input to another operation (nesting is possible).
- There are different basic operations that could be applied on relations on a database based on the requirements.
 - ▶ **Selection(σ)**: Selects a subset of rows from a relation.
 - ▶ It is a unary operator (it is applied to a single relation)
 - ▶ Set of conditions can be combined using Boolean operations (AND, OR, and NOT).
 - ▶ It is a filter that keeps only those tuples that satisfy a qualifying condition .
 - ▶ Notation: **σ <Selection condition> <Relation Name>**

Cont...

Example: Find all Employees with skill type of database?

<u>EmpId</u>	FName	LNam e	<u>Skill Id</u>	Skill	Skill Type	Scho ol	Skill level
10	AA	CV	1	SQL	Database	AAU	1
12	BB	BV	2	C++	Programmin g	AU	2
13	CC	CD	3	SQL	Database	AU	4
16	DD	GG	4	VB6	Programmin g	AAU	6
20	SS	CC	5	Oracl e	Database	AU	7

Q <Skill Type="Database">(Employee)

Cont...

❖ The result relation will be the following

<u>EmpId</u>	FName	LNam e	<u>SkillI d</u>	Skill	Skill Type	Schoo l	Skill level
10	AA	CV	1	SQL	Databas e	AAU	1
13	CC	CD	3	SQL	Databas e	AU	4
20	SS	CC	5	Oracle	Databas e	AU	7

❖ If the following

σ_{Skill Type="Database" AND School="AU"}(Employee)

<u>EmpId</u>	FName	LNam e	<u>SkillI d</u>	Skill	Skill Type	Scho ol	Skill level
13	CC	CD	3	SQL	Databas e	AU	4
20	SS	CC	5	Oracle	Databas e	AU	7

Cont...

- ▶ **Projection(π)**: Deletes unwanted columns from a relation
 - ▶ Selects certain attributes while discarding the other from the base relation.
 - ▶ Deleting attributes that are not in projection list.
 - ▶ Notation: **Π <Selected attributes> <Relation Name>**
 - ▶ **Example-1**: To display Name, Skill, and skill level of an employee, the query of the relation will be
 - ☞ **Π <Fname, Lname, Skill, Skill level> (Employee)**

Cont...

❖ **Example-2:** If we want to have the Name, Skill, and Skill level of an employee with skill SQL and Skill level greater than 5 the query will be

☞ **$\Pi_{\langle \text{Fname, Lname, Skill, Skill level} \rangle} (\sigma_{\langle \text{Skill} = \text{"SQL"} \text{ AND Skill level} > 5 \rangle} (\text{Employee}))$**

FName	LName	Skill	Skill level
DD	GG	VB6	6
SS	CC	Oracle	7

Cont...

- ▮ **Set Operations:** The three main set operations are Union, Intersection, and set difference
 - ▮ The properties of these set operations are similar with the concept we have in mathematical set theory.
 - ▮ The difference is that, in database context, the elements of each set which is a relation in database, will be tuples.
 - ▮ The set operations are Binary operations which demand the two operand relations to have type compatibility features.
 - ▮ **Type Compatibility:** Two relations R1 and R2 are said to be type compatible if:
 1. The operand relations R1(A1,A2...An) and R2(B1,B2...Bn) have the same number of attributes and

Cont...

2. The domain of corresponding attributes must be compatible, that is $\text{Dom}(A_i) = \text{Dom}(B_i)$ for $i=1,2,\dots,n$

A. Union Operation: The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S .

- Duplicate tuple is eliminated
- The two operands must be “type compatible”
- Example: $R_1 \cup R_2$
 - Employee who attend Database in any school or who attend any course at AU

Cont...

B. Intersection Operation:

- The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S.
- The two operands must be “type compatible”
- **Example:** $R1 \cap R2$
 - Employee who attend database course at AU

C. Set difference(Minus) operation:

- The result of this operation, denoted by $R - S$, is a relation that includes all tuples that are in R but not in S.
- The two operands must be “type compatible”
- **Example:** $R1 - R2$
 - Employee who attend database course but didn't take any course at AAU

Cont...

Some properties of the set operation

- ▮ $R \cup S = S \cup R$ and $R \cap S = S \cap R$
- ▮ $R \cup (S \cap T) = (R \cup S) \cap T$ and $R \cap (S \cup T) = (R \cap S) \cup T$
- ▮ $R - S \neq S - R$

D. Cartesian (cross product) operation:

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- That means, every tuple in R1 will be related with every other tuple in R2.
- In general, the result of $R(A_1, A_2 \dots A_n) \times S(B_1, B_2 \dots B_m)$ is a relation Q with degree $n + m$ attributes $Q(A_1, A_2, \dots A_n, B_1, B_2, \dots B_m)$ in that order.

Cont...

JOIN operation:

- ▢ Tuples joined from two relations based on a condition
- ▢ Using these we can build up sophisticated database queries
- ▢ This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
- ▢ The general form of a join operation on two relations $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_m)$ is: $R \underset{\text{<join condition>}}{\text{join}} S$ where R and S can be any relations that result from general *relational algebra expressions*.

Cont...

- ❖ **Example:** Suppose that we want to retrieve the name of the manager of each department.
 - ❖ To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
 - ❖ We do this by using the join operation.
 - ❖ $\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \underset{\text{EMPLOYEE}}{\text{MGRSSN=SSN}}$
- ❖ MGRSSN=SSN is the join condition
 - Combines each department record with the employee who manages the department

Cont...

□ Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).
- In a calculus expression, there is *no order of operations* to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain.
 - This is the main distinguishing feature between relational algebra and relational calculus.
- Relational calculus is considered to be a **nonprocedural** language.

Cont...

▮ This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

▮ A simple tuple relational calculus query is of the form

{t | COND(t)}

- where t is a tuple variable and COND (t) is a conditional expression involving t.
- The result of such a query is the set of all tuples t that satisfy COND (t).

▮ **Example:** To find the first and last names of all employees whose salary is above \$50,000, we can write the following tuple calculus expression:

{t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}