II B.Sc Computer Science
18BCS42C – DATABASE MANAGEMENT SYSTEM
UNIT-III

RELATIONAL MODEL: Introduction to Relational Data Model – Basic concepts – Enforcing data Integrity constraints – Relational Algebra Operations – Extended Relational Algebra Operations.

RELATIONAL DATABASE MANIPULATION: Introduction to Structured Query Language (SQL) – SQL Commands for defining Database, Constructing database, Manipulations on database – Basic data retrieval operations – Advanced Queries in SQL – Aggregation –Updates in SQL.

Relational Model

Introduction to Relational Data Model

- Relational data model is the primary data model, which is used widely around the world for data storage and processing.
- This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

Concepts

- **Tables** − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represent records and columns represent the attributes.
- **Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.
- **Relation instance** − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- **Relation schema** − A relation schema describes the relation name (table name), attributes, and their names.
- **Relation key** − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- **Attribute domain** − Every attribute has some pre-defined value scope, known as attribute domain.

Constraints

- Every relation has some conditions that must hold for it to be a valid relation.
- These conditions are called **Relational Integrity Constraints**.
- There are three main integrity constraints −
    - Key constraints
    - Domain constraints
    - Referential integrity constraints

Key Constraints

- There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely.
- This minimal subset of attributes is called **key** for that relation.
- If there are more than one such minimal subsets, these are called ***candidate keys***.
- Key constraints force that –
    - in a relation with a key attribute, no two tuples can have identical values for key attributes.
    - a key attribute can not have NULL values.
    - Key constraints are also referred to as Entity Constraints.

Domain Constraints

- Attributes have specific values in real-world scenario.
- For example, age can only be a positive integer.
- The same constraints have been tried to employ on the attributes of a relation.
- Every attribute is bound to have a specific range of values.
- For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

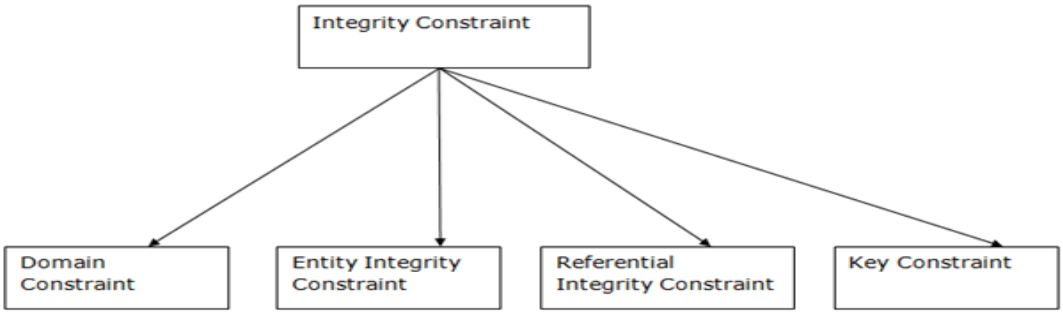
Referential integrity Constraints

- Referential integrity constraints work on the concept of Foreign Keys.
- A foreign key is a key attribute of a relation that can be referred in other relation.
- Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

Integrity Constraints

- Integrity constraints are a set of rules.
- It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc.
- The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

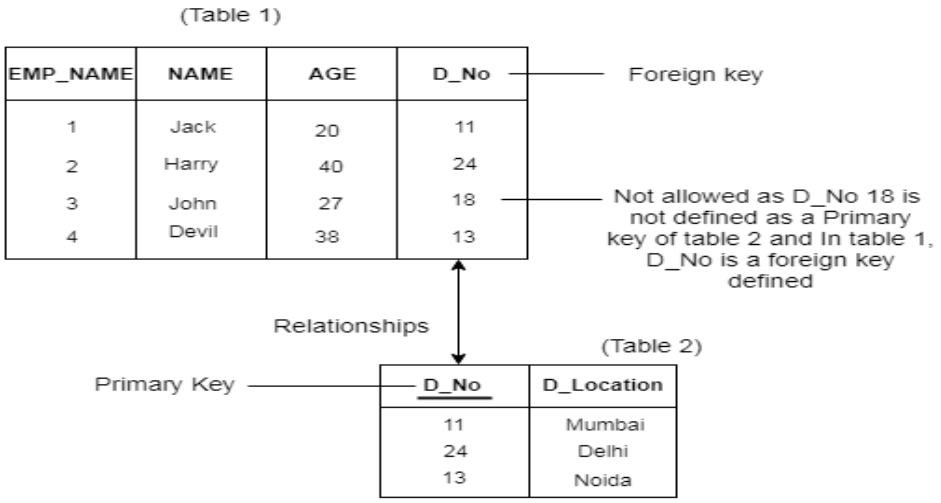
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:



4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key.
- A primary key can contain a unique and null value in the relational table.

Example:

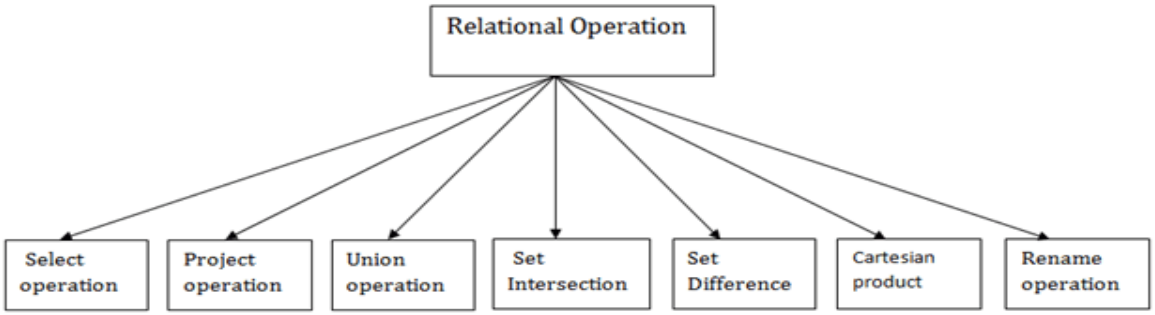
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

RELATIONAL ALGEBRA

- Relational algebra is a procedural query language.
- It gives a step by step process to obtain the result of the query.
- It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).
- Notation: σ p(r)
  Where:
  σ is used for selection prediction
  r is used for relation
  p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.
- For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result.
- Rest of the attributes will be eliminated from the table.
- It is denoted by ∏.

- Notation: ∏ A1, A2, An (r)
  Where
  A1, **A2**, **A3** is used as an attribute name of relation **r**.
- Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.
- Notation: R ∪ S
- A union operation must hold the following condition:
  - R and S must have the attribute of the same number.
  - Duplicate tuples are eliminated automatically.
- Example:
  DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	ACCOUNT_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Johnson
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.
- Notation: R ∩ S
- Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).
- Notation: R - S
- Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Williams
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.
- Notation: E X D
- Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales

2	Harry	B	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

- The rename operation is used to rename the output relation.
- It is denoted by **rho** (ρ).
- Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

  ρ(STUDENT1, STUDENT)

Extended Operators in Relational Algebra

- Extended operators are those operators which can be derived from basic operators.
- There are mainly three types of extended operators in Relational Algebra:
  - Join
  - Intersection
  - Divide

Intersection (∩):

- Intersection on two relations R1 and R2 can only be computed if R1 and R2 are **union compatible** (These two relation should have same number of attributes and corresponding attributes in two relations have same domain).
- Intersection operator when applied on two relations as R1∩R2 will give a relation with tuples which are in R1 as well as R2.
- Syntax:

Relation1 ∩ Relation2

Conditional Join($\bowtie_c$):

- Conditional Join is used when you want to join two or more relation based on some conditions.
- Example: Select students whose ROLL_NO is greater than EMP_NO of employees

$$\textbf{STUDENT} \bowtie_{\textbf{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \textbf{EMPLOYEE}$$

- In terms of basic operators (cross product and selection) :

$$\sigma_{\textbf{(STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO)}}(\textbf{STUDENT} \times \textbf{EMPLOYEE})$$

Equijoin($\bowtie$):

- Equijoin is a **special case of conditional join** where only equality condition holds between a pair of attributes.
- As values of two attributes will be equal in result of equijoin, only one attribute will be appeared in result.

Natural Join(⋈):

- It is a special case of equijoin in which equality condition hold on all attributes which have same name in relations R and S (relations on which join operation is applied).
- While applying natural join on two relations, there is no need to write equality condition explicitly.
- Natural Join will also return the similar attributes only once as their value will be same in resulting relation.
- Natural Join is by default inner join because the tuples which does not satisfy the conditions of join does not appear in result set.

Left Outer Join(⟕):

- When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions.
- But Left Outer Joins gives all tuples of R in the result set.
- The tuples of R which do not satisfy join condition will have values as NULL for attributes of S.

Right Outer Join(⟖):

- When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions.
- But Right Outer Joins gives all tuples of S in the result set.
- The tuples of S which do not satisfy join condition will have values as NULL for attributes of R.

Full Outer Join(⟗):

- When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions.
- But Full Outer Joins gives all tuples of S and all tuples of R in the result set.
- The tuples of S which do not satisfy join condition will have values as NULL for attributes of R and vice versa.

Division Operator (÷):

- Division operator A÷B can be applied if and only if:
    - Attributes of B is proper subset of Attributes of A.
    - The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)
    - The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

STRUCTURED QUERY LANGUAGE(SQL)

- **SQL** (*Structured Query Language*) is used to perform operations on the records stored in the database such as updating records, deleting records, creating and modifying tables, views, etc.
- SQL is a query language; it is not a database.
- SQL stands for **Structured Query Language**.

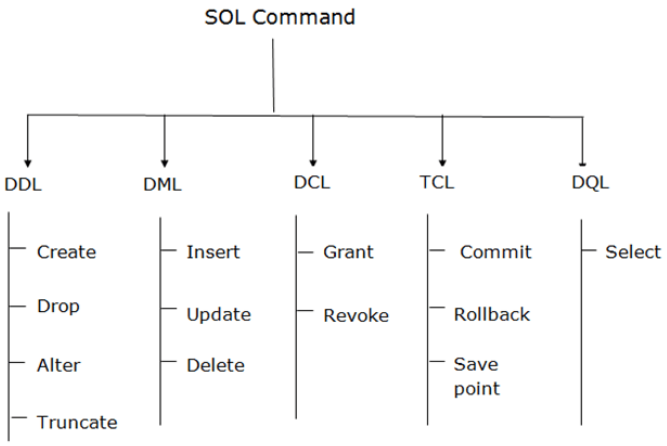
- It is designed for managing data in a relational database management system (RDBMS).
- It is pronounced as S-Q-L or sometime **See-Qwell**.
- SQL is a database language, it is used for database creation, deletion, fetching rows, and modifying rows, etc.
- SQL is based on relational algebra and tuple relational calculus.
- All DBMS like MySQL, Oracle, MS Access, Sybase, Informix, PostgreSQL, and SQL Server use SQL as standard database language.
- SQL is required:
  - To create new databases, tables and views
  - To insert records in a database
  - To update records in a database
  - To delete records from a database
  - To retrieve data from a database
- With SQL, we can query our database in several ways, using English-like statements.
- With SQL, a user can access data from a relational database management system.
- It allows the user to describe the data.
- It allows the user to define the data in the database and manipulate it when needed.
- It allows the user to create and drop database and table.
- It allows the user to create a view, stored procedure, function in a database.
- It allows the user to set permission on tables, procedures, and views.

SQL Commands

- SQL commands are instructions.
- It is used to communicate with the database.
- It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.
- Here are some commands that come under DDL:
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

a. CREATE:

It is used to create a new table in the database.

Syntax:

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

b. DROP:

It is used to delete both the structure and record stored in the table.

Syntax

DROP TABLE ;

Example

DROP TABLE EMPLOYEE;

c. ALTER:

It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

ALTER TABLE MODIFY(COLUMN DEFINITION....);

EXAMPLE

ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));

ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(2 0));

d. TRUNCATE:

It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language (DML)

- DML commands are used to modify the database.
- It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database.
- They can be rollback.
- Here are some commands that come under DML:
  - INSERT
  - UPDATE
  - DELETE

a. INSERT:

The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME

(col1, col2, col3,.... col N)
VALUES (value1, value2, value3, ... valueN);
Or
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3, ... valueN);

For example:

INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

b. UPDATE:

This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_name N = valueN] [WHERE CONDITION]

For example:

UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'

c. DELETE:

It is used to remove one or more row from a table.

Syntax:

DELETE FROM table_name [WHERE condition];

For example:

DELETE FROM javatpoint
WHERE Author="Sonoo";

3. Data Control Language(DCL)

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:
  - Grant
  - Revoke

a. Grant:

It is used to give user access privileges to a database.

Example

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

b. Revoke:

It is used to take back permissions from the user.

Example

REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, US ER2;

4. Transaction Control Language(TCL)

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.
- Here are some commands that come under TCL:
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

a. Commit:

Commit command is used to save all the transactions to the database.

Syntax:

COMMIT;

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
COMMIT;
```

b. Rollback:

Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
ROLLBACK;
```

c. SAVEPOINT:

It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

SAVEPOINT SAVEPOINT_NAME;

5. Data Query Language(DQL)

- DQL is used to fetch the data from the database.
- It uses only one command:
  - SELECT

a. SELECT:

This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

```
SELECT expressions
FROM TABLES
WHERE conditions;
```

For example:

```
SELECT emp_name
FROM employee
WHERE age > 20;
```

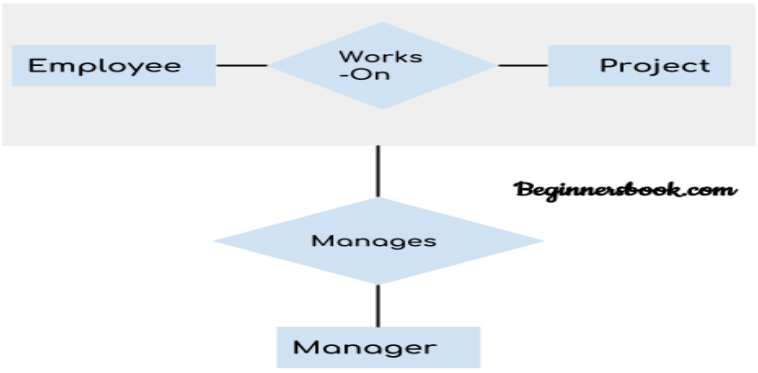
AGGREGATION

- In aggregation, the relation between two entities is treated as a single entity.
- In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

DBMS Aggregration

- **Aggregation** is a process in which a single entity alone is not able to make sense in a relationship so the relationship of two entities acts as one entity.

Aggregration Example



- In real world, we know that a manager not only manages the employee working under them but he has to manage the project as well.
- In such scenario if entity “Manager” makes a “manages” relationship with either “Employee” or “Project” entity alone then it will not make any sense because he has to manage both.

- In these cases the relationship of two entities acts as one entity.
- In our example, the relationship “Works-On” between “Employee” & “Project” acts as one entity that has a relationship “Manages” with the entity “Manager”.

Aggregation in DBMS

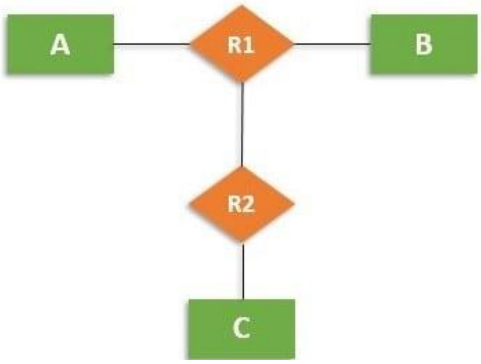
- Aggregation refers to the process by which entities are combined to form a single meaningful entity.
- The specific entities are combined because they do not make sense on their own.
- To establish a single entity, aggregation creates a relationship that combines these entities.
- The resulting entity makes sense because it enables the system to function well.
- When using data in the form of numerical values, the following operations can be used to perform DBMS aggregation:
    - **Average (AVG):** This function provides the mean or average of the data values.
    - **Sum:** This provides a total value after the data values have been added.
    - **Count:** This provides the number of records.
    - **Maximum (Max):** This function provides the maximum value of a given set of data.
    - **Minimum (Min):** This provides the minimum value of a given set of data.
    - **Standard deviation (std dev):** This provides the dispersion or variation of the sets of data.

The following are the main types of relationships in an ER model:

- **One-to-one:** Here, the trivial entity forms a relationship with only one other entity. For example, one employee can work in only one department of an organization.
- **One-to-many:** In this relationship, one entity forms a relationship with multiple entities. For example, an employee can work in multiple departments within the same organization.
- **Many-to-one:** Here, multiple entities in a certain entity set can form a relationship with only one entity. For example, many employees can work in only one department.
- **Many-to-many:** In this category, multiple entities from a certain entity set, that can form a relationship with many entities from another entity set. For

example, many employees can work in multiple departments within the same organization.

The following diagram shows a simple ER model that can be used to explain the process flow for aggregation in DBMS.

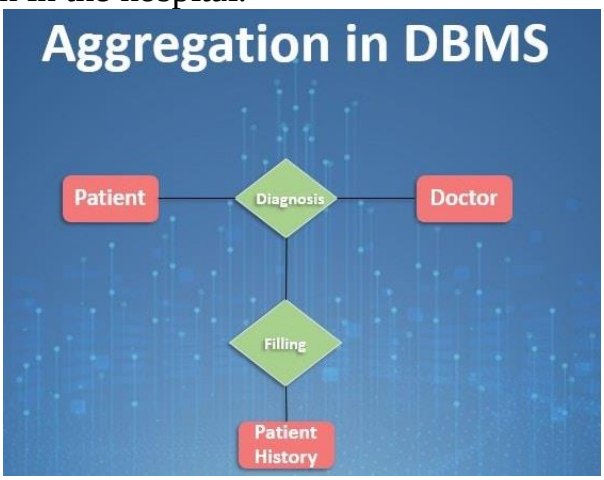


In this ER model, A, B, and C represent entities. A and B should be combined into a single complex entity. R1 is the relationship that is formed after A and B are linked. R1 needs to form a relationship with other entities for other DBMS operations to be successful.

This operation generates a new relationship (R2). R2 is linked to another entity C to enhance its functionality. This entity is also formed through aggregation.

Example of aggregation in DBMS

Let’s assume that there is a patient who has visited a doctor in the hospital to seek treatment for a certain type of illness. The following diagram shows the process flow for aggregation in the hospital.



We will follow the simple ER model described above. In the diagram above, there are three entities: patient history, the doctor, and the patient. Filing and diagnosis represent relationships. The doctor performs a diagnosis on the patient.

The database stores data regarding this diagnosis and any other patient data. Filing is required to make it easier for the doctor to retrieve the patient's information in the future.

In this example, the patient cannot work on his own. He has to form a relationship with the doctor to get a diagnosis. The doctor also cannot perform a diagnosis without the patient. In the future, the doctor will need data about the patient's history, that will require him to collect it from a filing system.

The last entity (patient's history) ensures that the entire system is functional. Getting the patient's history cannot be done without a diagnosis from the doctor and a filing system.

UPDATES IN SQL

- The SQL **UPDATE** Query is used to modify the existing records in a table.
- You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.
- Syntax
- 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];
```

- You can combine N number of conditions using the AND or the OR operators.
- Example

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;
```

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 you want to modify all the ADDRESS and the SALARY column values in the CUSTOMERS table, you 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;
```

Now, CUSTOMERS table would have the following records −

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Pune	2000.00
2	Khilan	25	Pune	1500.00
3	Kaushik	23	Pune	2000.00
4	Chaitali	25	Pune	6500.00
5	Hardik	27	Pune	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Pune	10000.00

The SQL UPDATE Statement

- The UPDATE statement is used to modify the existing records in a table.
- UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str.57	Berlin	12209	Germany
2	Ana Trujillo	Ana Trujillo	Avda. de la	Méxic o D.F.	05021	Mexico

	Empare dados y helados		Constitución 2222			
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

UPDATE Table

- The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.
- Example

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

The selection from the "Customers" table will now look like this:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str.57	Frankfurt	12209	Germany
2	Ana Trujillo Empare dados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

UPDATE Multiple Records

- It is the WHERE clause that determines how many records will be updated.
- The following SQL statement will update the ContactName to "Juan" for all records where country is "Mexico":

- Example

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

The selection from the "Customers" table will now look like this:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str.57	Berlin	12209	Germany
2	Ana Trujillo Empare dados y helados	Juan	Avda. de la Constitu ción 2222	Méxic o D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Matader os 2312	Méxic o D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	Londo n	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvs vägen 8	Luleå	S-958 22	Sweden

```
UPDATE Customers
SET ContactName='Juan';
```

The selection from the "Customers" table will now look like this:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Juan	Obere Str.57	Berlin	12209	Germany
2	Ana Trujillo Empare dados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Juan	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Juan	Berguvsväge n 8	Luleå	S-958 22	Sweden

REFERENCES

1. Silberschatz A., Korth H. and Sudarshan S., "Database System Concepts", Tata McGraw Hill, 2011.
2. Elmasri R. and Navathe S.B., "Fundamentals of Database Systems", Pearson Education, 2016.
3. www.w3schools.com
4. www.guru99.com
5. www.tutorialspoint.com
6. www.beginnersbook.com
7. www.ecomputernotes.com

----------**********----------