UNIT-4 Relational Data Model

After designing the Conceptual Model of the Database using E-R diagram, we need to Convert the Conceptual Model into a Relational Model which can be implemented using any RDBMS language like Oracle SQL, My SQL etc.

Relational Model

Relational Model Represents how data is stored in Relational Databases. A relational database consists of a collection of tables, each of which is assigned a Unique name.

Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE and AGE shaw in the given table.

ROLL-NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	94551x	18
2	RAMESH	GURGAON	9652x	18
3	SUJIT	ROHTAK	9156x	20
4	SURESH	DELHI		18

Important Terms

Attribute: These are the properties that define an entity. eg ROLLNO, NAME ADDRESS.

Relation Schema: A Relation Schema defines the structure of the

relation and represents the name of the relation with its attributes.

Tuple: Each row in the relation is known as tuple. The above relation contains 4 tuples, ~~one of~~

Relation Instance: The set of tuples of a relation at a particular instance of time is called a relation instance.

Degree: The Number of attributes in the relation is known as the degree of the relation.

Cardinality: The number of tuples in a relation is known as Cardinality.

NULL Values: The value which is not known or unavailable is called a NULL Value.

Relation Key: These are basically the keys that are used to identify the rows uniquely or also help in identifying tables.

Relational Constraints

Relational Constraints are the restrictions imposed on the database contents and operations. They ensure the correctness of data in the database.

Relational Constraints

- Domain Constraint
- Tuple Uniqueness Constraint
- Key Constraint
- Entity integrity Constraint
- Referential Integrity Constraint

1. Domain Constraint – Domain constraint defines the domain or set of values for an attribute. It specifies that the value taken by the attribute must be the atomic value from its domain.

Example – Consider the following student table–

STU-ID	Name	Age
S001	Akshay	20
S002	Shashank	21
S003	Rahul	A

Here, value 'A' is not allowed since only integer values can be taken by the age attribute.

2. Tuple Uniqueness Constraint – Tuple uniqueness constraint specifies that all the tuples must be necessarily unique in any relation.

Example –

STU-ID	Name	Age
S001	Akshay	20
S001	Akshay	20
S003	Shashank	20
S004	Rahul	21

This relation does not satisfy the tuple uniqueness constraint since here all the tuples are not unique.

3. Key Constraint – Key Constraint specifies that in any relation. All the values of primary key must be unique. The value of primary key must not be null.

Example-

STU-ID	Name	Age
S001	Akshay	20
S001	Abhishek	21
S003	Shashank	21
S004	Rahul	20

This relation does not satisfy the key constraint as here all the values of primary key are not unique.

4. Entity Integrity Constraint – Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation. This is because the presence of null value in the primary key violates the uniqueness property.

Example-

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	21
-	Rahul	20

This relation does not satisfy the entity integrity constraint as here the primary key contains a NULL value.

:- Referential Integrity Constraint – This Constraint is enforced when a foreign key references the primary key of a relation. It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

Student

STU-ID	Name	Dept_no
S001	Akshay	D10
S002	Shashank	D11
S003	Rahul	D14

Department

Dept_no	Dept_name
D10	ASET
D11	ALS
D12	ASEL

Here, the relation 'Student' does not satisfy the referential integrity Constraint. This is because in relation 'Department' no value of primary key specifies department no. 14.

Relational Algebra

1. Selection – Selection in relational Algebra return those tuple (record) in a relation that fulfill a condition (produce table contain subset of rows)

Syntax : ? Condition (relation)

The tables (for Student)

Relation : Student

? class = 12 (s)

Ex: Give all information of Student having Ad NO 105.

Sol $\sigma_{AdNO=105}$ (Student)

σ = Sigma

Ad No	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65
205	Leena	11	B	95

Output

Ad NO.	Name	Class	Section	Avg
101	Anu	12	A	85
105	Balu	12	D	65

2. Projection – Projection in relational Algebra return those Column in a relation that given in the attribute list.

Syntax : π attribute list (Relation)

Ex– π Ad no, Name (S)

S. Student

Ad NO.	Name
101	Anu
102	Balu
103	Lena
104	Neha
105	Seema
106	Ram

3. Union – The Union operator is used to Combine two or more tables Each table within the Union should have the same Number of Columns, similar data-types and also the Column must be in same order.

Syntax– σ (Student 1)
Union
σ (Student 2)
Or
π ad.no, Name (Student 1)
Union
π ad.no, Name (Student 2)

Ex:

R = {A, B, C, D}

S = {A, C, E, F}

RUS = {A, B, C, D E, F}

Table : student 1

AdNO	Name	Class
101	Ram	12
102	Shyam	12
103	Seeta	12

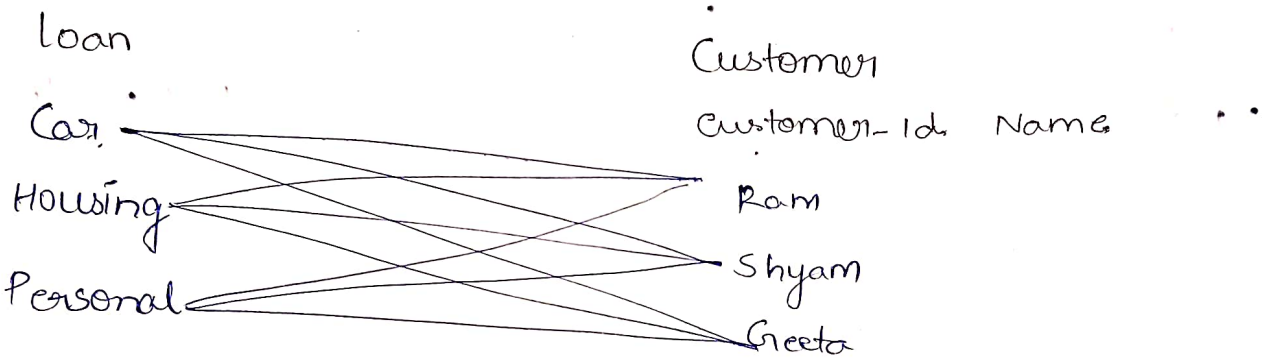
Table : student 2

AdNO	Name	Class
104	Seeta	12
105	Reeta	11
106	Shyam	12

Output

AdNo.	Name	Class
101	Ram	12
102	Shyam	12
103	Seeta	12
104	Reeta	11

4. Cartesian product -



SQL

SQL is structured query language. SQL is standard language for accessing and manipulating database. It is non-procedural in nature. The SQL standard specifies data definition, data manipulations and other associated facility of a DBMS that support the relational data model.

- SQL is a comprehensive language for controling and interacting with database mgmt system.

- SQL command are used to implement the following -

1- SQL can retrieve data from a database.

2- SQL can insert record in database.

3- SQL can update, delete and

4- It can create new tables and views in a database.

1) Create Table Command - Create table command is used to Create table structure. In this command we need to give information about tables such as - Number of columns, types of each column and Constraints.

Create table command requires name of the table name of the field, definition and, Constraints for each Field.

Constraints - In SQL we have following Constraints.

1. NOT NULL → To check a column cannot store a null value.
2. Primary key → To check that a column have a unique identify which help to find a particular record in table.

Syntax:

```
CREATE TABLE <table name>
<Column name 1> <datatype> [size] [constraints]
<Column name 2> <datatype> [size] [ Constraints]
```

2) INSERT INTO Command -

Syntax: INSERT INTO <table name> [Column name 1 , Column name n

or

INSERT INTO <table name > Values (Value 1 , Value 2 , - - - Value n)

3) SELECT Command -

```
SELECT *
FROM <table name>
[where <condition> ]
```

3- SQL can update, delete and create record in database.

4- It can create new tables and views in a database.

1) Create Table Command – Create table command is used to create table structure. In this command we need to give information about tables such as - Number of columns, types of each column and constraints.

Create table command requires name of the table name of the field, definition and constraints for each field.

Constraints – In SQL we have following constraints.

1. NOT NULL → To check a column cannot store a null value.
2. Primary key → To check that a column have a unique identity which help to find a particular record in table.

Syntax:

```
CREATE TABLE <table name>
<Column name 1> <datatype> [size] [constraints]
<Column name 2> <datatype> [size] [ Constraints]
```

2) INSERT INTO Command –

Syntax: INSERT INTO <table name> [Column name 1 , Column name

or

INSERT INTO <tablename > Values (Value 1 , Value 2 , - - Value n)

3) SELECT Command –

```
SELECT *
FROM <table name>
[where <condition> ]
```

4) WHERE Clause –

It is used to extract only those records that fulfills a specified condition.

Syntax: SELECT * FROM Customers
WHERE Country = 'Mexico';

5) ORDER BY –

The ORDER BY keyword is used to sort the result set in ascending or descending order.

Syntax:

```
SELECT column 1, Column 2, ..
FROM table_name
ORDER BY Column 1, Column 2, ...
ASC|DESC;
```

6) SQL AND Operator

AND operator is used to filter records based on more than one condition, like if you want to return all Customers from Spain that starts with the letter 'G'.

Syntax: / Example

```
SELECT *
FROM Customers
WHERE Country = 'Spain' AND
Customer Name LIKE 'G%';
```

```
SELECT col 1, Col 2, ..
FROM table_name
WHERE Condition 1 AND Condition 2
AND Condition 3...;
```

7) SQL OR operator

The OR operator is used to filter records based on more than one condition, like if you want to return all Customers from Germany but also from Spain.

Syntax:

```
SELECT Col1, Col2, ..
FROM table_name
WHERE Condition1 OR Condition2 OR
Condition3..;
```

8) NOT operator

The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.

Syntax:

```
SELECT col1, col2, ..
FROM table_name
WHERE NOT Condition;
```

9) UPDATE Statement

Update statement is used to modify the existing records in a table.

Syntax:

```
UPDATE table_name
SET Col1 = value1, col2 = value2, ..
WHERE Condition;
```

10) DELETE Statement

Delete statement is used to delete existing records in a table.

Syntax:

```
DELETE FROM table_name WHERE Condition;
```

SQL Aggregate Functions

An aggregate function is a function that performs a Calculation on a Set of values, and returns a Single value.

The most Commonly Used SQL aggregate functions are:

- MIN() - returns the smallest value within the selected Column.
- MAX() - returns the largest value within the Selected column.
- COUNT() - returns the number of rows in a Set.
- SUM() - returns the total sum of a numerical Column.
- AVG() - returns the average value of a Numerical column.

SQL Joins

A JOIN Clause is used to Combine rows from two or more tables, based on a related column between them.

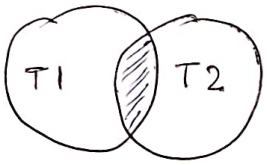
Different Types of SQL JOINS

- INNER JOIN: Return records that have matching values in both tables.
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table.

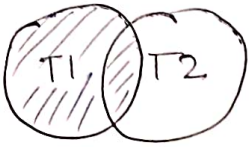
- RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table.

- FULL (OUTER) JOIN: Return all records when there is a match in either left or right table.

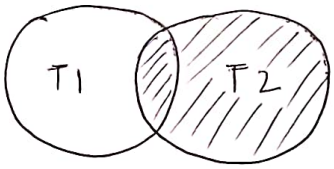
INNER JOIN



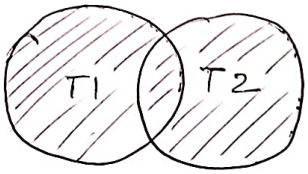
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



Syntax (INNER JOIN)

```
SELECT Column_name(s)
FROM table1
INNER JOIN table2
ON table1.colname = table2.col_name;
```

Syntax (LEFT JOIN)

```
SELECT Column_name(s)
FROM table1
LEFT JOIN table2
ON table1.col_name = table2.col_name;
```

Syntax (RIGHT JOIN)

```
SELECT Column-name(s)
FROM table1
RIGHT JOIN table2
ON Table1.col-name = Table2.col-name;
```

Syntax (FULL OUTER JOIN)

```
SELECT Column-name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.col-name = table2.col-name
WHERE condition;
```

SQL UNION Operator

Union operator is used to Combine the result-set of two or more SELECT statements.

Syntax (UNION)

```
SELECT Column-name(s) FROM table1
UNION
SELECT Column-names FROM table2;
```

GROUP BY Statement

Group By Statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

CREATE DATABASE

The Create Database statement is used to create a new SQL database.

Syntax:

```
CREATE DATABASE database name;
```

DROP DATABASE

The DROP DATABASE statement is used to drop an existing SQL database.

Syntax:

```
DROP DATABASE databasename;
```

CREATE TABLE

Syntax:

```
CREATE TABLE table_name (
 col1 datatype,
 col2 datatype,
col3 datatype,
 ...
 );
```

ALTER TABLE

Syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```