Page:
Date: / /

Lecture-16

Unit - III

Form of Basic SQL Query.

Query :-
A Query is a statement requesting the retrieval of information.

=>) Query Language :-
The portion of a DML that involves information retrieval is called a Query Language.

=>) SQL :-
is stands for Structured Query Language.

⊙ SQL is the most widely used commercial relational database Language.

The Form of a Basic SQL Query.

SELECT [DISTINCT] select-list

FROM from-list
WHERE qualifications

⊙ SELECT →
Clause specifies columns to be retained in the result.

⊙ FROM → Clause specifies a Cross-product of tables.

Scanned with CamScanner

Page:
Date: / /

⊙ Option WHERE → clause specifies selection conditions on the tables mentioned in the FROM clause.

Attribute (or field)

Ex:-

Sid	Sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Record (or) tuple

Q. Find the names and ages of all sailors —

Query: SELECT DISTINCT sname, age
FROM sailors

O/P →

Sname	age
Dustin	45.0
Brutus	33.0
Lubber	55.5
Andy	25.5

Scanned with CamScanner

Rusty	35.0
Horatio	35.0
Zorba	16.0
Art	25.5
Bob	63.5

⇒ The DISTINCT keyword can be used to return only distinct (different) values.

2 Find all sailors with a rating above 7.

query:- SELECT sid, sname, rating, age
FROM sailors
WHERE rating >7
(or)
SELECT *
FROM sailors
WHERE rating >7

31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0

⇒ The SQL language has (3) parts —

Page:
Date: / /

1. DDL
2. DML
3. DCL.

DDL commands—
- CREATE
- ALTER
- DROP
- GRANT
- REVOKE
- TRUNCATE

REATE:-To create a Table using SQL and Describes data.

```
CREATE TABLE STUDENT(

NAME CHAR(20),
BRANCH VARCHAR(20),
DOB    VARCHAR(10),
Roll_No NUMBER(10));
```

Table created.

```
SQL> DESC STUDENT;
```

DML → INSERT:—

⇒ To insert values in a table using SQL and retrieve their values.—

```
INSERT INTO STUDENT VALUES ('NISHA',
```

Scanned with CamScanner

Lecture-17

Page:
Date: / /

'CSE', '13/10/91', '10');

INSERT INTO STUDENT VALUES('VIKAS', 'IT',

'26/12/91', '12');

SELECT :-

SELECT * from STUDENT;

It will give the complete records of table Student.

O/P

Name	Branch	DOB	Roll_No.
NISHA	CSE	13/10/91	10
VIKAS	IT	26/12/91	12.

- Nested Queries in SQL :-

In nested Query, a query is written inside a query.

⇒ The result of inner query is used in execution of outer query.

Ex :- we will use STUDENT, COURSE etc. STUDENT_COURSE for understanding nested queries.

STUDENT

S_ID	S_NAME	S_ADDRESS	S_PHONE	S_AGE
S1	RAM	DELHI	9455123451	18

Scanned with CamScanner

S2	RAMESH	GURGAON	9652431543	18
S3	SUJIT	ROHTAK	9156253131	20
S4	SURESH	DELHI	9156768971	18

COURSE

C_ID	C_NAME
C1	DSA
C2	Programming
C3	DBMS

STUDENT_COURSE

S_ID	C_ID
S1	C1
S1	C3
S2	C1
S3	C2
S4	C2
S4	C3

⇒ There are mainly two(2) types of nested queries :-

① Independent Nested Query :-

Page:
Date: / /

In independent nested queries, query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query.

⇒ Various operators like IN, NOTIN, ANY, ~~ANY~~, ALL etc. are used in writing independent nested queries.

IN :- If we want to ~~add~~ find out S_ID who are enrolled in C_NAME 'DSA' or 'DBMS', we can write it with the help of independent nested query and IN operator.

⇒ from COURSE table, we can find out C_ID for C_Name 'DSA' or DBMS' and we can use these C_ID for finding S_ID from STUDENT_COURSE table.

Step1 :- finding C_ID for C_NAME = 'DSA' or 'DBMS'.

Select C_ID from COURSE Where C_NAME = 'DSA' or C_NAME = 'DBMS'.

Step2 :- Using C_ID of Step1 for finding S_ID.

Scanned with CamScanner

Page:
Date: / /

Select S_ID from STUDENT_COURSE
where C_ID IN

(SELECT C_ID from COURSE where

C_NAME = 'DSA' or C_NAME = 'DBMS');

=)) The inner query will return a set with members C1 and C3 and outer query will return those S_IDs for which C_ID is equal to any member of set (C1 and C3 in this case).

So, it will return S1, S2 and S4,

Correlated Sub Queries :-
are also similar to sub-queries but here the outer query is executed first and inner query is executed for each records of outer query.

=)) That is inner query is executed as many times as the outer query results.

EX:=)) Suppose we need to select the Students whose marks have been entered into marks table.

(We can write much simpler query

Page:
Date: / /

than below by not writing WHERE clause in the sub query, but below query is written to show how correlated subquery is written and executed).

```
SELECT *
        from Students
  WHERE STD_ID IN (SELECT STD_ID FROM
MARKS m WHERE S.STD_ID = m.STD_ID);
```

=>) Here we can see that outer query columns and inner query column are joined to get the result.

Diff. B/W subquery and correlated sub-query :-

Sub Query	Correlated Sub Query
Inner Query is executed first	Outer Query is executed first.
Inner query is executed only once and its results is used by outer query.	Inner query is executed for each of the records that outer query returns.
Uses using =, <, >, >=, <=, IN, BETWEEN operators.	Can use using =, <, >, >=, <=, IN, BETWEEN operators,

Scanned with CamScanner

Page:
Date: / /

	but it mainly uses EXISTS and NOT EXISTS clause.
(4) is always used in the WHERE clause.	(4) is used in WHERE clause as well as columns of SELECT Statement.

Comparison operator :-

or relational operator is a mathematical symbol which is used to compare two values.

=> It is used in conditions that compares one expressions with another.

=> The result of comparison can be TRUE, FALSE or UNKNOWN (an operator that has one or two Null expressions returns UNKNOWN).

operator =, >, <, >=, <=, <> → Not equal to

Syntax:- Select [column_name | * | expression]

name of a table

all the columns of a table

< comparison operator > [column_name | * | expression]
FROM <table_name>

WHERE <expression> [comparison operator] <expression>;

Scanned with CamScanner

Page:
Date: / /

EX:- CREATE TABLE EMPLOYEE (
ENAME VARCHAR(20),
ECODE NUMBER (20),
ESALARY NUMBER (30),
EGENDER CHAR(2),
EDEPTT CHAR(10),
ECITY CHAR(15));

SQL> insert into Employee Values('RAHUL','101',
'2000000','M','IT','BENGALURU');

SQL> insert into Employee values('NISHA','102',
'100000','F','CSE','DELHI');

Select * FROM EMPLOYEE;

O/P

ENAME	ECODE	ESALARY	EG	EDEPTT	ECITY
RAHUL	101	2000000	M	IT	BENGALURU
NISHA	102	100000	F	CSE	DELHI

SELECT ESALARY, EDEPTT FROM EMPLOYEE WHERE ESALARY > 1000000;

ESALARY	EDEPTT
2000000	IT

EX: = SELECT ENAME FROM EMPLOYEE
WHERE ENAME = 'NISHA';

ENAME
NISHA

Scanned with CamScanner

Page:
Date: / /

Lecture-18

Aggregative operators :-

Aggregate f^n allows you to perform a calculation on a set of values to return a single scaler value.

⇒ We often use aggregate f^n with the GROUP BY and HAVING clauses of the SELECT statement.

⇒ The following are the most commonly used SQL aggregate f^n :-

1. AVG :- calculates the averages of a set of values.
2. COUNT :- Counts rows in a specified table or view.
3. MIN :- gets the minimum value in a set of values.
4. MAX :- gets the maximum value in a set of values.
5. SUM :- calculates the sum of values.

SQL aggregate f^n Syntax :-

⇒ To call an aggregate f^n, you use the following syntax :

Scanned with CamScanner

Page:
Date: / /

aggregate_function(DISTINCT | ALL expression)

① First specify an aggregate f^n that you want to use Ex:- MIN, MAX, AVG, SUM, or COUNT.

② Second, put DISTINCT or ALL modifier followed by an expression inside parenthesis.

⇒ If you explicitly use the DISTINCT modifier, the aggregate f^n ignores duplicate values and only consider the unique values.

⇒ If you use the ALL modifier, the aggregate f^n uses all values for calculation or evaluation.

Ex: COUNT f^n ex:-

⇒ To get the no. of products in the products table, you use the COUNT f^n as follows:-

```
SELECT
    COUNT(*)
FROM
    products;
```

O/P →

COUNT(*)
77

Scanned with CamScanner

Page:
Date: / /

AVG f^n

To calculate the average units in stock of the products, you use the AVG f^n as follows: —

```
SELECT
    AVG(unitsinstock)

FROM
    products;
```

O/P

AVG(unit in stock)
40.5065

SUM f^n: -

To calculate the sum of units in stock by product category we can use the SUM f^n with the GROUPBY clause as the foll. query: —

```
SELECT
    categoryid, SUM(unitsinstock)

FROM
    products

GROUP BY categoryid;
```

Scanned with CamScanner

Page:
Date: / /

category id	sum(unitinstock)
1	559
2	507
3	386
4	393
5	308
6	165
7	100
8	701

MIN function :-

To get the minimum units in stocks of products table.

Ex: of Aggregate f^n.

1. count()
2. sum()
3. Avg()
4. Min()
5. Max()

Id	Name	Salary.
1	A	80
2	B	40
3	C	60
4	D	70
5	E	60
6	F	Null

Scanned with CamScanner

Page:
Date: / /

① Count(*) → Returns total no. of records i.e 6.

• Count(Salary) → return no. of Non Null Values over the column Salary i.e 5.

• Count (Distinct salary) :- Return no. of distinct Non null value over the Column Salary i.e 4.

Syntax :- Select COUNT (Column_name) From table :name WHERE condition;

② Sum() :-

Sum (Salary) :- Sum all Non null Values of column Salary i.e 310.

Sum (Distinct salary) :- sum of all distinct Non-null Values i.e 250.

Syntax :- Select sum (column name) from tablename WHERE condition;

③ Avg ( ) :

Avg(salary) = sum (salary) / count(salary)

= 310/5 = 62.

Avg (Distinct Salary) = Sum(Distinct Salary) / Count (Distinct salary)

Scanned with CamScanner

Page:
Date: / /

Lecture-19

$= \frac{250}{4}$ Syntax:- Select AVG(column name) from table_name WHERE condition;

(4) Min():

Min(salary):- Minimum value in the salary column except NULL i.e 40.

Syntax:- select MIN(column_nam) from table name WHERE condition;

Max(salary):- Maximum value in the salary i.e 80.

Syntax:- SELECT MAX (column name) from table name WHERE condition.

SQL Null Values:- The SQL NULL is the term used to represent a missing value.

⇒ A Null value in a table is a value in a field that appear to be blank.

⇒ A field with a Null 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.

Scanned with CamScanner

Page:
Date: / /

Syntax:- of NULL while creating a table.

```
CREATE TABLE CUSTOMER (

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

Here, NOT NULL → signifies that columns should always accept an explicit value of the given data type. There are two columns where we did not use NOT NULL, which means these columns could be NULL.

Ex:-

ID	NAME	AGE	ADDRESS	SALARY
1	Nisha	27	Delhi	20000
2	Vikas	27	AHMEDABAD	25000
3	Shivanshi	25	Kota	65000
4	Naveen	31	Mumbai	18500
5	Kulku	30	Bhopal	24000
6	Rahul	24	MP	—
7	Gajendra	24	INDORE	—

Usage of the <u>IS NOT NULL</u> operator.

Select ID, NAME, AGE, ADDRESS, SALARY

Scanned with CamScanner

Page:
Date: / /

FROM CUSTOMERS
WHERE SALARY IS NOT NULL;

It will show the result upto ID (5).

* USAGE of the IS NULL operator

SELECT ID, NAME, AGE, ADDRESS, SALARY
FROM CUSTOMER
WHERE SALARY IS NULL;

O/P →

ID	NAME	AGE	ADDRESS	SALARY
6	Rahul	24	MP	
7	Rajendra	24	INDORE	

SQL LOGICAL OPERATORS:-

There are three (3) logical operators namely:-

AND, OR, and NOT.

⇒ These operators compare two conditions at a time to determine whether a row can be selected for the output.

⇒ When retrieving data using a SELECT statement, we can use logical operators in the WHERE clause, which allows you to

Scanned with CamScanner

Page:
Date: / /

Combine more than one condition.

Logical operators	Description
OR	For the row to be selected at least one of the conditions must be true.
AND	For a row to be selected all the specified cond^n must be true.
NOT	For a row to be selected the specified cond^n must be false.

"OR" Logical operator:-

If you want to select rows that satisfy atleast one of the given conditions, you can use the logical operators (OR).

For Ex:-

If you want to find the names of students who are studying either maths or science the query would be like:-

```
SELECT first_name, last_name, subject
FROM Student_details

WHERE subject = 'Maths' OR subject =
'Science'
```

Scanned with CamScanner

Page:
Date: / /

O/P ⇒

first_name	last_name	Subject
Anjali	Bhagwat	Maths.
Shekhar	Gowda	Maths.
Rahul	Sharma	Science
Stephen	Fleming	Science

⇒ The following table describes how logical operator "OR" selects a row.

Column 1 Satisfied?	Column 2 satisfied?	Row selected
YES	YES	YES
YES	NO	YES
NO	YES	YES
NO	NO	NO

AND Logical operator :-

⇒ If you want to select rows that must satisfy all the given cond", you can use the logical operator, AND.

for Ex :- To find the names of the students between the age 10 to 15 years, the query would be like :-

```
SELECT first_name, last_name, age
FROM Student_details
WHERE age >= 10 and age <= 15;
```

Scanned with CamScanner

Page:
Date: / /

O/P :-

first_name	last_name	age
Rahul	Sharma	10
Anjali	Bhagwat	12
Shekhar	Gewda	15

⇒ The foll. table describes how logical "AND" operator selects a row.

Column1 satisfied?	Column2 satisfied?	Row selected
YES	YES	YES
YES	NO	NO
NO	YES	NO
NO	NO	NO.

"NOT" operator :-

If you want to find rows that do not satisfy a condn, you can use the logical operator NOT

FOR EX :- If you want to find out the names of the students who do not play football, the query would be like :-

```
SELECT first_name, last_name,
       games
FROM student_details
```

Scanned with CamScanner

Page:
Date: / /

WHERE NOT games = 'Football'

O/P →

first_name	last_name	games
Rahul	Sharma	Cricket
Stephen	Fleming	Cricket
Shekhar	Gowda	Badminton
priya	Chandra	Chess

Join :—
is a combination of a cartesian product followed by a selection process.

⇒ Join operation pairs two tuples from different relations, if and only if a given Join condition is satisfied.

Types —

① Theta(θ) Join :—
Combines tuples from different relations provided they satisfy the theta cond^n.

⇒ The ~~Theta (θ)~~ Join cond^n is denoted by the symbol θ.

Notation : —

Scanned with CamScanner

Page:
Date: / /

$R_1 \bowtie_\theta R_2$

⇒ R_1 and R_2 are relations having attributes $(A_1, A_2, \ldots, A_n)$ and $(B_1, B_2, \ldots, B_n)$ such that the attributes don't have anything in common, that is $R_1 \cap R_2 = \phi$

⇒ Theta Join can use all kinds of comparison operators.

Student.

SID	Name	Std
101	Alex	10
102	maria	11

Subjects

Class	Subjects
10	Math
10	English
11	Music
11	Sports.

Student_Detail —

Student $\bowtie_{\text{student.Std} = \text{subject.Class}}$ SUBJEC

Student_detail

SID	NAME	Std	Class	Subject

Scanned with CamScanner

Page:
Date: / /

101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports.

Equijoin :-

When Theta Join uses only equality comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

Natural Join (⋈) :-

It does not use any comparison operator. It does not concatenate the way a cartesian product does. We can perform a Natural Join only if there is atleast one common attribute that exists between two relations.

⇒ Natural Join acts on those matching attributes where the value of attributes in both the relations are same.

Courses.

CID	course	Dept.
CS 01	Database	CS
ME 01	Mechanics	ME
EE 01	Electronics	EE

Scanned with CamScanner

Page:
Date: / /

HOD.

Dept	Head
CS	Alex
ME	Maya
EE	Misa

Course ⋈ HOD

Dept	CID	Course	Head
CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya
EE	EE01	Electronics	Misa

Outer Joins :-

⇒ Theta Join, Equijoin, and Natural Join are called inner Joins.
An inner Join includes only those tuples with matching attributes and the rest are discarded in the resulting relation.

⇒ Therefore, we need to use outer Joins to include all the tuples from the participating relations in the resulting relation.

⇒ There are three(3) kinds of outer Joins —

① Left Outer Join (R ⟕ S)

Scanned with CamScanner

Page:
Date: / /

② Right outer Join (R ⟖ S)

③ Full outer Join (R ⟗ S)

① Left outer Join :-

All the tuples from the Left relation, R are included in the resulting relation.

⇒ If there are tuples in R without any matching tuples in the Right relation S, then the S-attributes of the resulting relation are made NULL.

Left :

A	B
100	Database
101	Mechanics
102	Electronics

Right :

A	B
100	Alex
102	Maya
104	Mira

Courses ⟖ HOD

A	B	C	D
101	Database	100	Alex
104	Mechanics	--	--
102	Electronics	102	Maya.

Scanned with CamScanner

Page:
Date: / /

Right outer Join : (R ⟖ S)

⇒ All the tuples from the Right relation, S, are included in the resulting relation.

⇒ If there are tuples in S without any matching tuples in R, then the R- attributes of resulting relation are made NULL.

courses ⟖ HOD.

A	B	C	D
100	Database	100	Alex
102	Electronics	102	Maya
—	— —	104	Mira

Full Outer Join (R ⟗ S).

⇒ All the tuples from both participating relations are included in the resulting relation.

⇒ If there are no matching tuples for both relations, their respective unmatched attributes are made NULL.

Courses ⟗ HOD

A	B	C	D
100	Database	100	Alex
101	Mechanics	— —	— —
102	Electronics	102	Maya
— —	— —	104	Mira

Scanned with CamScanner

Page:
Date: / /

Lecture - 20

SQL constraints :-
are used to specify rules for data in a table.

SQL Create constraints :-
Constraints can be specified when the table is created with the CREATE TABLE stmt, or after the table is created with the ALTER TABLE Statement.

Syntax :-

```
CREATE TABLE table_name (

  Column1 datatype constraint,
  Column2 datatype constraint,
  Column3 datatype constraint,
  ------
  );
```

⇒ constraints can be column level or table Level.

Column Level :-
Constraints apply to a Column, and ——

Table Level constraints :-
apply to the Whole table.
Types ——

Scanned with CamScanner

Page:
Date: / /

① NOT NULL → Ensures that a column cannot have a NULL value.

② UNIQUE :-
Ensures that all values in a column are different.

③ Primary Key :-
A combination of a NOT NULL and UNIQUE. Uniquely identifies each rows in a table.

④ Foreign Key :-
Uniquely identifies a row/ record in another table.

⑤ Check :-
Ensures that all values in a column satisfies a specific condⁿ.

⑥ Default :-
Sets a default value for a column when no value is specified.

⑦ INDEX :-
Used to create and retrieve data from the database very quickly.

① SQL Not NULL Constraints —

By default, a column can hold

Scanned with CamScanner

Page
Date / /

Null values.

⇒ The NOT NULL constraints enforces a column to Not accept Null values.

Ex:- SQL NOT NULL on CREATE TABLE

```
Create table persons(

 ID int Not NULL,
 Last Name varchar(255) NOT NULL,
 First Name varchar(255) Not NULL,
 Age int );
```

SQL Not NULL on alter table

```
ALTER TABLE Persons

MODIFY Age int NOT NULL;
```

Scanned with CamScanner

Page:
Date: / /

Lecture-21

Triggers :-

A Triggers is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE or DELETE statement) is performed on a specified table.

⇒ Triggers are Useful for tasks such as enforcing business rules, Validating input data, and Keeping an audit trail.

Uses of Triggers :-

1. Enforces Business Rules.
2. Validate input data.
3. Access System function.
4. Generate a unique value for a newly inserted row in a different file.
5. Replicate data to different files to achieve data consistency.

Creation of MYSQL Triggers :-

⇒ A Triggers is a named database object that is associated with a table, and it activates when a particular event (ex. insert, update

Page:
Date: / /

or delete) occur for the table. The statement CREATE TRIGGER creates a new trigger in MYSQL.

Syntax:-

CREATE
[DEFINER = {user | CURRENT_USER}]

TRIGGER trigger_name

trigger_time trigger_event

ON tbl_name FOR EACH ROW
trigger_body

trigger_time: {BEFORE | AFTER}
trigger_event: {INSERT | DELETE}.

Active Database:-

A database that has the ability to spontaneously react to event occuring inside as well as outside the system is called active database.

⇒ The ability to respond to external events is called active behaviour.

⇒ The active behaviour is based

Page:
Date: / /

on the rules that integrate a event with the desired effect.

⇒ This behaviour is commonly defined in terms of ECA rules allowing system to react to specific events.

There are ② types of architecture for active database —

① Built-In
② Layered.

① Built-In Architecture :-

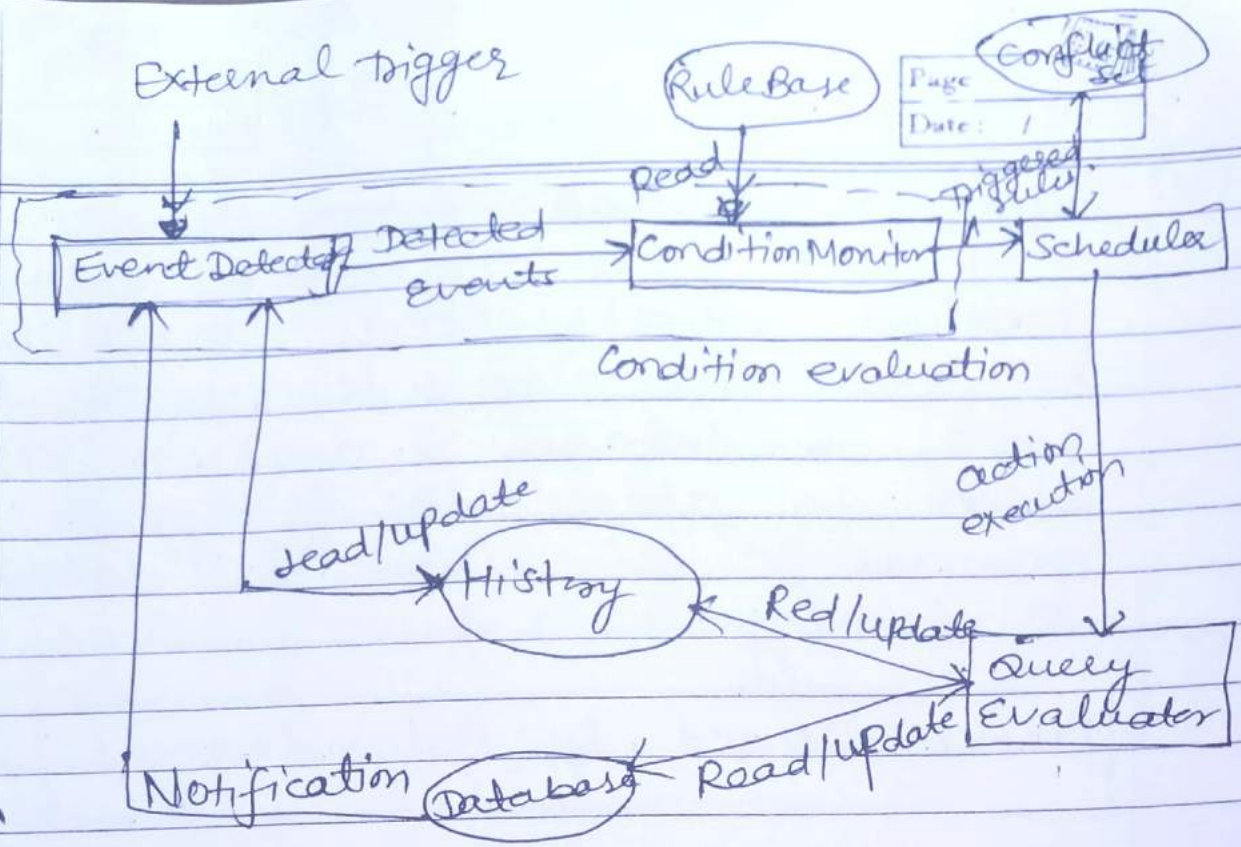
⇒ Active database Components becomes part of database.

⇒ It can be ~~activet~~ achieved by two (2) models

⊙ Implementation from Scratch :- active Components are implemented as a part of database from Scratch.

⊙ Integrated Architecture :-
This involves modifying and extending the existing passive database.

Scanned with CamScanner



(1) Event Detector :-

It detects any external event occuring and inform the condition manager about it.

(2) Condition Monitor :-

It evaluates the conditions of rules associated with events that have been detected by the event detector.

(3) Scheduler :-

It compares recently triggered rules with those that have previously been triggered, update the conflict set, and fires any rules that are scheduled for immediate processing.

Page:
Date: / /

(4) Query Evaluater :-

It executes database queries as actions. Access may be required both to the current state of database and to past states in order to support monitoring of how the database is evolving.

Problem caused by Redundancy :

Redundancy :-

means having multiple copies of same data in the database. This problem arises when a database is well normalized.

Ex:- Suppose a table of student details attributes are:-

student id, Student name, college name, college rank, course etc.

Student_Id	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	BTech	1
101	Ankit	7900734858	GEU	Btech	1
102	Ayush	7300936759	GEU	Btech	1
103	Ravi	8009015556	GEU	Btech	1

Scanned with CamScanner

Lecture-22

Page
Date: / /

~~Schema Refinement:~~

=>) As it can be observed that value of attributes college name, College Rank, Course is being repeated which can lead to problems.

=>) Problems caused due to redundancy are :-

① Insertion Anomaly —

If a student details has to be inserted whose course is not being decided yet then insertion will not be possible till the time course is decided for student.

Student-id	Name	Contact	College	Course	Rank
100	Himanshu	7300934851		GEU	1

=>) This problem happens when the insertion of a data record is not possible without adding some additional unrelated data to the record.

② Deletion Anomaly —

If the details of student in this table is deleted then the details of

Page:
Date: / /

College will also get deleted which should not occur by common sense.

⇒ This anomaly happens when deletion of a data records results in losing some unrelated information that was stored as part of the record that was deleted from a table.

③ Updation Anomaly –

⇒ Suppose if the rank of the college changes the changes will have to be all over the database which will be time-consuming and computationally costly.

Student ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Ayush	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1

Scanned with CamScanner

Page
Date: / /

⇒) If updation do not occur at all places then database will be in inconsistent state.

Decomposition:—

- It is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

Properties of Decomposition:—

1. Lossless Decomposition
2. Dependency Preservation

Scanned with CamScanner

Page:
Date: / /

③ Lack of Data Redundancy.

① Looseless Decomposition :-

Decomposition must be looseless. It means that the information should not get lost from the relation that is decomposed.

⇒ It gives a guantee that the Joins will result in the same relation as it was decomposed.

Ex :-

Let's take - 'E' is the Relational Schema with instance 'e', is decomposed into : E1, E2, E3, ... En ;

With instance : e1, e2, e3, --- en, if

e1 ⋈ e2 ⋈ e3 ... ⋈ en, then it is called as 'Looseless Join Decompo-sition'.

⇒ In the above ex, it means that, if natural joins of all the decomposition give the original relation, then it is said to be laseless join decomposition.

Ex :- <Employee - Department> Table.

Scanned with CamScanner

Page
Date: / /

Eid	Ename	Age	City	Salary	Deptid	Dept Name
E001	ABC	29	pune	20000	D001	Finance
E002	PQR	30	pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

- Decompose the above relation into two relations to check whether a decomposition is lossless or lossy.

- Now, we have decomposed the relation that is Employee and Department.

Relation 1 : <Employee> Table.

Eid	Ename	Age	City	Salary
E001	ABC	29	pune	20,000
E002	PQR	30	pune	30,000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Bangalore	25000

Scanned with CamScanner

Page:
Date: / /

- Employee schema contains (Eid, Ename, Age, City, Salary).

Relation 2 : <Department> Table :-

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	HumanResource

- Department schema contains (Deptid, Eid, DeptName).

- So, the above decomposition is a lossless Join Decomposition, because the two relations contains one common field that is 'Eid' and therefore Join is possible.

- Now apply natural Join on the decomposed relations.

$\boxed{\text{Employee} \bowtie \text{Department}}$:-

Scanned with CamScanner

Lecture - 23

Page
Date: / /

Eid	Ename	Age	city	Salary	Deptid	Dept Name
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	HR.

⇒ Hence, the decomposition is Lossless Join Decomposition.

Normalization :-

① Normalization is the process of organizing the data in the database

② It is used to minimise the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like insertion, update and Deletion Anomalies.

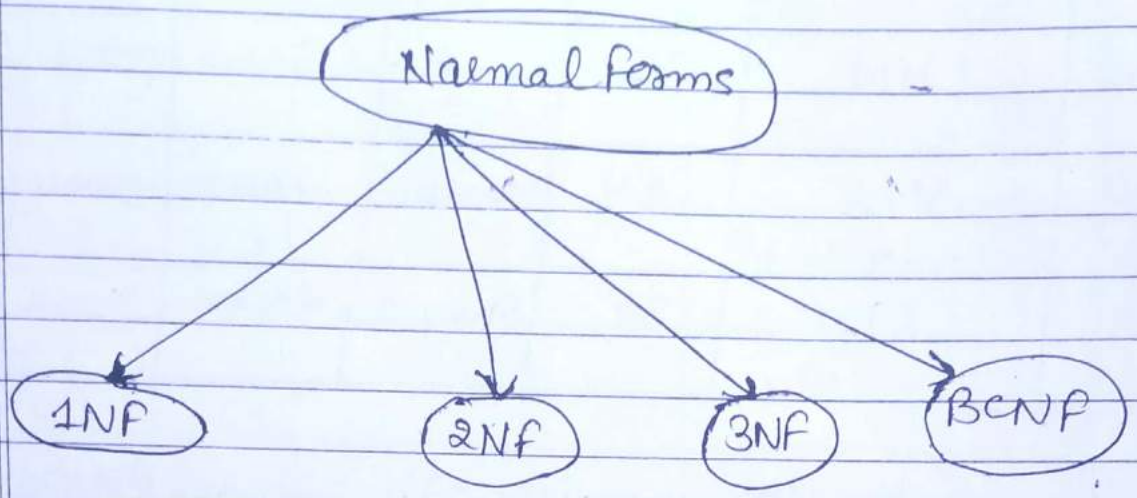
③ It divides the larger table into the smaller table and links them using relationship.

Scanned with CamScanner

Page:
Date: / /

(4) The normal form is used to reduce redundancy from the database table.

Types of Normal Form:-



1NF → A relation is in 1 NF if it contains an atomic value.

2NF → A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

3NF :- A relation will be in 3NF if it is 2NF and no transition dependency exists.

4NF :- A relation will be in 4NF if it is Boyce Codd normal form and has no multi-valued dependency.

5NF :- A relation is in 5NF if it is in

Scanned with CamScanner

4NF and not contains any join dependency and Joining should be lossless.

Functional Dependency

⇒ The functional dependency is a relationship that exists between two attributes.
It typically exists between the primary key and non-key attribute within a table.

$$X \longrightarrow Y$$

⇒ The left side of FD is known as left Determinant, as the

⇒ Right side of the production is known as dependent.

For Ex:-

⇒ Assume we have an employee table with attributes :- Emp-id, Emp-name, Emp-Address.

Here Emp-Id attribute can uniquely identify the Emp-Name attribute of employee table because

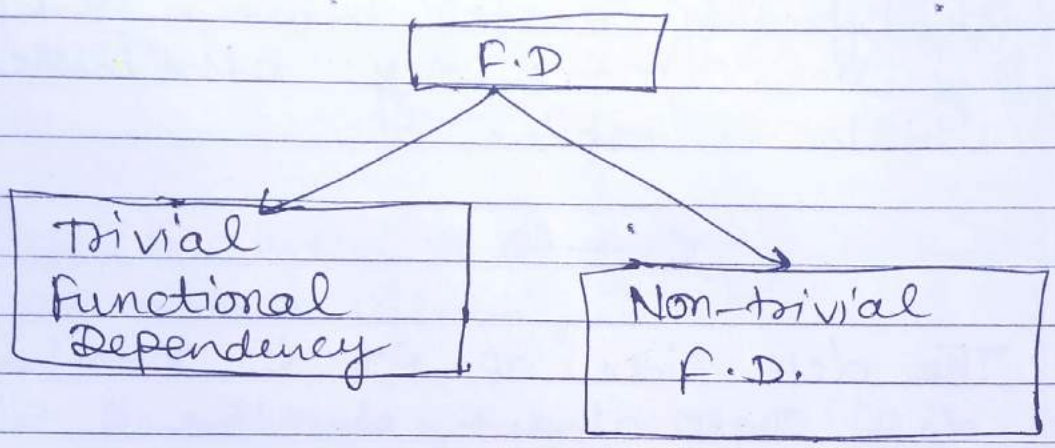
Page:
Date: / /

if we know the Emp_id, we can tell that employee name associated with it.

FD can be written as:—
Emp_id → Emp_Name.

⇒ We can say that Emp_Name is FD on Emp_id.

Types of FD:—



(1) Trivial Functional Dependency:—

- $A \rightarrow B$ has trivial FD if B is a subset of A.
- The following dependencies are also trivial like: $A \rightarrow A$, $B \rightarrow B$.

Ex:- Consider a table with two columns Employee_Id and Employee_Name.

{Employee_id, Employee_Name} →

Page:
Date: / /

First Normal Form :- (1NF) :-

① A relation will be 1NF if it contains an atomic value.

② It states that attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

③ 1NF disallows the multi-valued attribute, composite attribute and their combinations.

Ex:- Relation EMPLOYEE is not in 1f because of Multi-valued attribute EMP_PHONE.

Employee Table :-

Emp_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8594783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

⇒ The Decomposition of the Employee Table into 1NF has been shown below :-

Scanned with CamScanner

Page
Date: / /

EmpID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Bihar
12	Sam	8589830302	Punjab.

Second Normal form (2NF):

① In the 2NF, relation must be in 1NF.

② In the second Normal Form, all non-key attributes are fully functional dependent on the primary key.

Ex:- Let's assume, A School can store the data of teacher and the subject they teach. In a school, a teacher can teach more than one subject.

Teacher table

Teacher_id	Subject	Teacher_Age
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

Scanned with CamScanner

Page:
Date: / /

⇒ In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key.

⇒ That's why it violates the rule for 2NF.

⇒ To convert the given table into 2NF, we decompose it into two tables.

TEACHER_Detail table :-

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT TABLE :-

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Scanned with CamScanner

Page
Date: / /

Third Normal form (3NF)

① A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.

② 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

③ If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

⇒ A relation is in third normal form if it holds atleast one of the following cond^n for every non-trivial FD $X \rightarrow Y$.

① X is a super key.
② Y is a prime attribute i.e each element of Y is part of some Candidate Key.

Ex:- EMPLOYEE DETAIL Table:-

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Scanned with CamScanner

Page:
Date: / /

Super key in the table above:-

{Emp_ID}, {EMP_ID; EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}..... so on.

Candidate Key:- {EMP_ID}.

Non-prime attributes:-

In the given table, all attributes except Emp_ID are non-prime.

⊙ Here, EMP_STATE and EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on Super key (EMP_ID).

It violates the rule of 3NF.

⇒ That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a primary key.

Employee Table

Emp_id	Emp_Name	Emp_zip
222	Harry	201010
333	Stephan	02228
444	Lan	60007

Scanned with CamScanner

Lecture-24

Page
Date: / /

555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP TABLE:-

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Boyce Codd Normal form (BCNF):-

1. BCNF is the advance version of 3NF. It is stricter than 3NF.

2. A Table is in BCNF if every FD $X \rightarrow Y$, X is the super key of the table.

3. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Scanned with CamScanner

Page:
Date: / /

Ex:- Let's assume there is a Company where employees work in more than one department.

EMPLOYEE table:-

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_Type	Emp_no
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

=>) In the above table FD are as follows:-

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT, EMP_DEPT_No}.

Candidate Key → EMP_ID → {EMP_COUNTRY}

EMP_DEPT → {DEPT_TYPE, EMP_DEPT}

=>) The Table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

=>) To convert the given table into

Page
Date: / /

three (3) tables:-

EMP_COUNTRY table:-

EMP_ID	EMP_COUNTRY
264	INDIA
264	INDIA

Dept

EMP_DEPT table:-

EMP_DEPT	DEPT_TYPE	EMP_DEPT_No.
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING TABLE:-

EMP_ID	EMP_DEPTT.
D394	283
D394	300
D283	232
D283	549

Scanned with CamScanner

Page:
Date: / /

FD :-

EMP_ID → EMP_COUNTRY
EMP-DEPT → {DEPT_TYPE, EMP_DEPT_No}

Candidate Keys :-

For the first table : EMP_ID
For the Second " : EMP_DEPT
For the Third " : {EMP_ID, EMP_DEPT}

⇒ Now, this is in BCNF because left side part of both the FD is a key.

Fourth Normal Form (4NF).

- A relation will be in 4NF if it is in BCNF and has no multi-valued dependency.
- For a dependency A → B, if for single value of A, multiple value of B exists, then the relation will be a multi-valued dependency.

Ex:- STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

Scanned with CamScanner

Page
Date: / /

=> The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

=> In the STUDENT relation, a student with STUD_ID ; 21 contains two courses, computer and Math and two Hobbies, Dancing and Singing. So there is a Multi-Valued Dependency on STU_ID, which leads to unnecessary repitition of data.

=> So to make the above table into UNF, we can decompose it into the tables —

STUDENT_COURSE.

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Bialogy
59	Physics

Scanned with CamScanner

Page:
Date: / /

Lecture-25

STUDENT-HOBBY.

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey.

Fifth Normal Form (5NF):-

- A relation is in 5NF if it is in 4NF and not contains any Join dependency & joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-Join Normal Form (PJ/NF).

EX:-

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Sem1
Computer	John	Sem1
Math	John	Sem1
Math	Akash	Sem2
Chemistry	Praveen	Sem1.

Scanned with CamScanner

Page:
Date: / /

⇒ In the above table,

Dependency-Preserving Decomposition:-

⇒ It is another property of decomposed relational database schema D in which each functional dependency $X \rightarrow Y$ specified in F either appeared directly in one of the relation schemas R_i in the decomposed D or could be inferred from the dependencies that appear in some R_i.

⇒ Decomposition $D = \{R_1, R_2, R_3, \ldots, R_m\}$ of

- R is said to be dependency-preserving w.r.to F if the union of the protection of F on each R_i, in D is equivalent to F.

⇒ In other words, R C join of R_1, R_1 over X.
The dependencies are preserved because each dependency in F represent a constraint on the database.

⇒ If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Ex:- Let a relation R(A, B, C, D) and set a FDs $F = \{A \rightarrow B, A \rightarrow C\}$, and

Page:
Date: / /

$R_2 = (C, D)$ with FDs $F_2 = \{C \rightarrow D\}$.

$F' = F_1 \cup F_2 = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

So, $F' = F$

& So, $F'^{+} = F^{+}$.

=> Thus, the decomposition is dependency preserving decomposition.

Schema Refinement :-
is Just a fancy term for saying polishing tables.

=> It is the last step before considering design / tuning with physical workloads.

1. Req. Analsis : user needs.
2. Conceptual Design : high level description, often using E/R diagrams.
3. Logical design : from graph to tables (relational schema)
4. Schema Refinement :- checking tables for redundancies & anomalies.

Scanned with CamScanner

Page
Date: / /

⇒ In schema Refinement, we remove redundancy problem.

⇒ In order to identify schemas with such problems, we introduce concept of functional dependencies.

a relationship that exists when one attribute uniquely determines another attributes

Multivalued Dependency in DBMS :-

⇒ It occurs in the situation where there are multiple independent multivalued attributes in a single table.

⇒ A multivalued dependencies is a complete constraint between two sets of attributes in a relation.

⇒ It requires that certain tuples be present in a relation.

Ex :-

Car_model	Maf_year	Calar
H001	2017	Metallic
H001	2017	Green

Scanned with CamScanner

Page:
Date: / /

H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

⇒ In this ex:- maf_year and colar are independent of each other but dependent on car_model.

⇒ In these ex:, these two columns are said to be multivalue dependent on car_model.

⇒ This dependence can be represented like this:-

car_model → maf_year

car_model → colour.

Scanned with CamScanner