

**DEPARTMENT OF BCA**

**DATA ANALYTICS**

1

**SUBJECT: RDBMS SEMESTER: 3rd BCA JAIN UNIVERSITY BCA**

MODULE-2

**RELATIONAL**

**ALGEBRA**

**TRIGGERS**

2 

**JAIN UNIVERSITY BCA**

3

**JAIN UNIVERSITY BCA**

**What are Triggers in PL/SQL?**

Triggers are stored programs that are fired automatically when some events occur. The code to be fired can be defined as per the requirement.

Oracle has also provided the facility to mention the event upon which the trigger needs to be fire and the timing of the execution.

4

**JAIN UNIVERSITY BCA**

**Types of Triggers in Oracle**

 Triggers can be classified based on the following parameters.

 Classification based on the **timing**

 BEFORE Trigger: It fires before the specified event has occurred.

 AFTER Trigger: It fires after the specified event has occurred.

 Classification based on the **level**

 STATEMENT level Trigger: It fires one time for the specified event statement.  ROW level Trigger: It fires for each record that got affected in the specified event. (only for DML)  Classification based on the **Event**

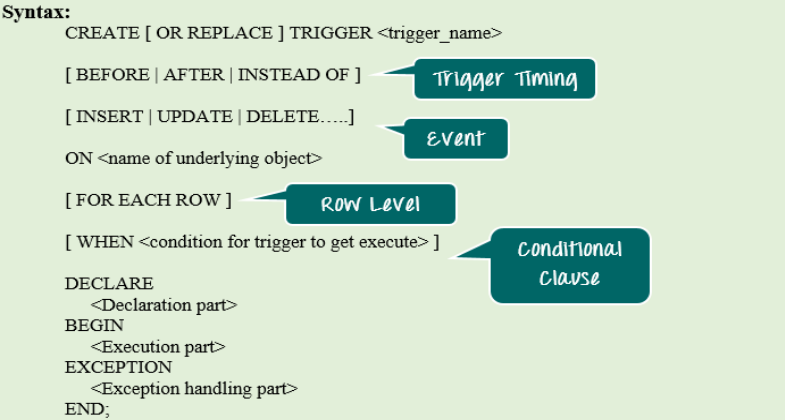
 DML Trigger: It fires when the DML event is specified (INSERT/UPDATE/DELETE)  DDL Trigger: It fires when the DDL event is specified (CREATE/ALTER)

 DATABASE Trigger: It fires when the database event is specified

(LOGON/LOGOFF/STARTUP/SHUTDOWN)

 So each trigger is the combination of above parameters.

**How to Create Trigger** 5

 **Below is the syntax for creating a trigger. **JAIN UNIVERSITY BCA

6

**JAIN UNIVERSITY BCA**

EXAMPLE ON TRIGGER

Create or replace trigger trig\_error

before insert or update or delete

on emp

begin

if to\_char(sysdate,'DY') IN ('MON','TUE') then

raise\_application\_error(-20111,'No changes can be made on tuesday');

dbms\_output.put\_line(‘Cant insert/delete on Tuesday’); end if;

end;

7

**EXAMPLE-2: Create a trigger to insert a new record to log table after inserting a record to dept table**

Create or replace trigger depttrig

AFTER INSERT ON dept

FOR EACH ROW

BEGIN

INSERT INTO dept\_log

VALUES(:new.deptno,:new.dname,:new.loc);

END;

**EXAMPLE-3: Create a trigger to insert aold record to log table after updating or deleting a record to dept table**

Create or replace trigger dept\_trig

BEFORE update or delete on dept

FOR EACH ROW

BEGIN

INSERT INTO dept\_log

VALUES(:old.deptno,:old.dname,:old.loc);

**JAIN UNIVERSITY BCA** END

8 **EXAMPLE-3: Create a trigger to count number of records from emp table after deleting a record to emp table**

CREATE OR REPLACE TRIGGER emp\_count

AFTER DELETE ON EMP

DECLARE

n INTEGER;

BEGIN

SELECT COUNT(\*) INTO n FROM emp;

DBMS\_OUTPUT.PUT\_LINE(' There are now ' || n || ' employees.'); END;

**JAIN UNIVERSITY BCA**

9 **Example 4:Create a trigger Convert ename to upper**

CREATE OR REPLACE TRIGGER before\_emp

BEFORE UPDATE OR INSERT ON emp

FOR EACH ROW

begin

/\* convert character values to upper case \*/

:new.ename := upper( :new.ename );

:new.job := upper( :new.job);

end;

**JAIN UNIVERSITY BCA**

10 **Dictionary for triggers is user\_triggers**

**SQL>Select trigger\_name,trigger\_body,table\_name,description**

**from user\_triggers**

**Enabling/Diasabling Triggers**

**To enable a disabled trigger, use the ALTER TRIGGER statement with the ENABLE clause.**

Example:

**SQL>ALTER TRIGGER trig\_error ENABLE;**

To enable all triggers defined for a specific table, use the ALTER TABLE statement with the ENABLE clause and the ALL TRIGGERS option.

**SQL>ALTER TABLE emp ENABLE ALL TRIGGERS;**

**JAIN UNIVERSITY BCA**

11 Disabling Triggers

You might temporarily disable a trigger if:

You must perform a large data load, and you want it to proceed quickly without firing triggers.

ALTER TRIGGER trig\_error DISABLE;

To disable all triggers defined for a specific table, use the ALTER TABLE statement with the DISABLE clause and the ALL TRIGGERSoption. For example, to disable all triggers defined for the Inventory table, enter the following statement:

ALTER TABLE emp DISABLE ALL TRIGGERS;

**JAIN UNIVERSITY BCA**

12

**JAIN UNIVERSITY BCA**

**Database security** refers to the collective measures used to protect and **secure** a

**database** or **database** management software from illegitimate use and malicious threats and attacks.  It is a broad term that includes a multitude of processes, tools and methodologies that

ensure **security** within **database** environment.

13

**JAIN UNIVERSITY BCA**

**Advanced SQL features – (CASE, DECODE, Rollup, Cube) done in SQL**

**Embedded SQL–** Embedded SQL is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL. Embedded SQL statements are SQL statements written inline with the program source code, of the host language.

 **Need of Embed SQL:** Because the host language cannot parse **SQL**, the inserted **SQL** is parsed by an **embedded SQL** preprocessor. **Embedded SQL** is a robust and convenient method of combining the computing power of a programming language with **SQL's** specialized data management and manipulation capabilities

14

**JAIN UNIVERSITY BCA**

**Dynamic SQL:**

 Dynamic SQL is a programming methodology for generating and running statements at run-time. It is mainly used to write the general-purpose and flexible programs where the SQL statements will be created and executed at run-time based on the requirement.

 Embedded SQL are SQL statements in an application that do not change at run time and, therefore, can be hard-coded into the application. Dynamic SQL is SQL statements that are constructed at run time; for example, the application may allow users to enter their own queries.

15

**JAIN UNIVERSITY BCA**

**Introduction to Distributed Databases:**

 In a distributed database, there are a number of databases that may be geographically distributed all over the world. A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users.

 **The two types of distributed systems are as follows:**

 Homogeneous distributed databases system: Homogeneous distributed database system is a network of two or more databases (With same type of DBMS software) which can be stored on one or more machines. ...

 Heterogeneous distributed database system.

 Example: Oracle distributed database systems employ a distributed processing architecture to function. For example, an Oracle server acts as a client when it requests data that another Oracle server manages.

16

**JAIN UNIVERSITY BCA**

**Client/Server Databases**

A **client**/**server application** is a piece of software that runs on a **client** computer and makes requests to a remote **server**. Many such **applications** are written in high-level visual programming languages where UI, forms, and most business logic reside in the **client application**.

17 Relational Algebra

Relational Algebra

Fundamental operations

Additional Operations

Domain Relational Calculus Tuple Relational Calculus

**JAIN UNIVERSITY BCA**

18 Relational Query Languages

Languages for describing queries on a relational database *Structured Query Language* (SQL) 

Predominant application-level query language

Declarative 

*Relational Algebra*

Intermediate language used within DBMS

Procedural

**JAIN UNIVERSITY BCA**

19 What is an Algebra?

A language based on operators and a domain of values.

Operators map values taken from the domain into other domain values

Hence, an expression involving operators and arguments produces a value in the domain

When the domain is a set of all relations (and the operators are as described later), we get the *relational algebra* 

 We refer to the expression as a *query* and the value produced as the *query result* 

**JAIN UNIVERSITY BCA**

20

**JAIN UNIVERSITY BCA**

**Relational Algebra:**

 Relational algebra is a procedural query language that works on relational model. The purpose of a query language is to retrieve data from database or perform various operations such as insert, update, delete on the data. When I say that relational algebra is a procedural query language, it means that it tells what data to be retrieved and how to be retrieved.

 On the other hand relational calculus is a non-procedural query language, which means it tells what data to be retrieved but doesn’t tell how to retrieve it.

 **Types of operations in relational algebra:**

 We have divided these operations in two categories:

1. Basic Operations

2. Derived Operations

Formal Languages

21

**Formal Languages**

**Relational Algebra Relational calculus**

**Tuple Relational calculus**

**Domain Relational Calculus**

**JAIN UNIVERSITY BCA**

22 Relational Algebra

*Domain*: set of relations

*Basic operators*: select, project, union, set difference, Cartesian product 

*Derived operators*: set intersection, division, join 

*Procedural*: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression

**JAIN UNIVERSITY BCA**

23

**JAIN UNIVERSITY BCA**

**Basic/Fundamental Operations:**  1. Select (σ)

 2. Project (∏)

 3. Union (∪)

 4. Set Difference (-)

 5. Cartesian product (X)

 6. Rename (ρ)

**Derived Operations:**

 1. Natural Join (⋈)

 2. Left, Right, Full outer join (⟕, ⟖, ⟗)  3. Intersection (∩)

 4. Division (÷)

24 Select Operator

 Produce table containing subset of rows of argument table satisfying condition

σ*condition relation*

 Example:

Person σ*Hobby*=‘stamps’(Person) *Id Name Address Hobby Id Name Address Hobby*

1123 John 123 Main stamps

1123 John 123 Main coins

5556 Mary 7 Lake Dr hiking

9876 Bart 5 Pine St stamps

**JAIN UNIVERSITY BCA**

1123 John 123 Main stamps 9876 Bart 5 Pine St stamps

25 Selection Condition

Operators: <, ≤, ≥, >, =, ≠

Simple selection condition:

<*attribute*> *operator* <*constant*>

<*attribute*> *operator* <*attribute*>

<*condition*> AND <*condition*>

<*condition*> OR <*condition*>

NOT <*condition*>

**JAIN UNIVERSITY BCA**

26 Selection Condition - Examples

σ Id>3000 Or Hobby=‘hiking’ (Person) 

σ Id>3000 AND Id <3999 (Person) 

σ NOT(Hobby=‘hiking’) (Person)

σ Hobby≠‘hiking’ (Person)

**JAIN UNIVERSITY BCA**

27 Project Operator

 Produces table containing subset of columns of argument table Π*attribute list*(*relation*)

 Example:

Person Π*Name,Hobby*(Person)

Id Name Address Hobby Name Hobby

**JAIN UNIVERSITY BCA**

1123 John 123 Main stamps 1123 John 123 Main coins 5556 Mary 7 Lake Dr hiking 9876 Bart 5 Pine St stamps

John stamps John coins Mary hiking Bart stamps

28 Project Operator Example:

Person Π*Name,Address*(Person)

*Id Name Address Hobby Name Address*

**JAIN UNIVERSITY BCA**

1123 John 123 Main stamps

1123 John 123 Main coins

5556 Mary 7 Lake Dr hiking

9876 Bart 5 Pine St stamps

Result is a table (no duplicates)

John 123 Main Mary 7 Lake Dr Bart 5 Pine St

29 Expressions

Π *Id, Name* (σ *Hobby=’stamps’* OR *Hobby=’coins’* (Person) ) Person Result

*Id Name Address Hobby Id Name*

**JAIN UNIVERSITY BCA**

1123 John 123 Main stamps 1123 John 123 Main coins 5556 Mary 7 Lake Dr hiking 9876 Bart 5 Pine St stamps

1123 John 9876 Bart

30 Set Operators

Relation is a set of tuples => set operations should apply Result of combining two relations with a set operator is a relation => all its elements must be tuples having same structure 

Hence, scope of set operations limited to *union compatible relations* 

**JAIN UNIVERSITY BCA**

31 Union Compatible Relations

Two relations are *union compatible* if 

Both have same number of columns

Names of attributes are the same in both

Attributes with the same name in both relations have the same domain

Union compatible relations can be combined using *union*, *intersection*, and *set difference* 

JAIN UNIVERSITY BCA

32 **Union Operation (**∪**)**

 It performs binary union between two given relations and is defined as −  r ∪ s = { t | t ∈ r or t ∈ s} **Notation** − r U s

 Where **r** and **s** are either database relations or relation result set .  For a union operation to be valid, the following conditions must hold −  **r**, and **s** must have the same number of attributes.

 Attribute domains must be compatible.

 Duplicate tuples are automatically eliminated.

 ∏ author (Books) ∪ ∏ author (Articles)

**JAIN UNIVERSITY BCA**

33 **Example**

Tables:

Person (SSN, Name, Address, Hobby)

Professor (Id, Name, Office, Phone)

are not union compatible. However

Π *Name* (Person) and Π *Name* (Professor)

are union compatible and

Π *Name* (Person) *-* Π *Name* (Professor)

makes sense.

**JAIN UNIVERSITY BCA**

**Set Difference (−)**

34

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

**Notation** − **r** − **s**

Finds all the tuples that are present in **r** but not in **s**.

∏ author (Books) − ∏ author (Articles)

**Output** − Provides the name of authors who have written books but not articles.

JAIN UNIVERSITY BCA

35

**JAIN UNIVERSITY BCA**

**Rename Operation (ρ)**

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

**Notation** − *ρ* x(E)

Where the result of expression **E** is saved with name of **x**.

**Cartesian Product (Χ)**

Combines information of two different relations into one.

Notation − r Χ s

Where r and s are relations and their output will be defined as − r Χ s = { q t | q ∈ r and t ∈ s}

σ author = ‘Martin Seligmen'(Books Χ Articles)

Output − Yields a relation, which shows all the books and articles written by Martin Seligmen.

36 Derived Operation: Join

The expression :

σ *join-condition´(R* × *S)*

where *join-condition´* is a *conjunction* of terms:

*Ai oper Bi*

in which *Ai* is an attribute of *R, Bi*is an attribute of *S,* and *oper* is one of =, <, >, ≥ ≠, ≤, is referred to as the (theta) join of *R* and *S* and denoted:

*R join-condition S*

**JAIN UNIVERSITY BCA**

37

**JAIN UNIVERSITY BCA**

Types of join operations



38

**JAIN UNIVERSITY BCA**

39

JAIN UNIVERSITY BCA

40

**JAIN UNIVERSITY BCA**

41

**JAIN UNIVERSITY BCA**

42

**JAIN UNIVERSITY BCA**

43 Join and Renaming

**Problem**: *R* and *S* might have attributes with the same name .

**Solution**:

Rename attributes prior to forming the product and use new names in *join-condition´*.

Common attribute names are qualified with relation names in the result of the join

**JAIN UNIVERSITY BCA**

44 Theta Join – Example

Output the names of all employees that earn more than their managers.

Π**Employee*.Name* (Employee *MngrId=Id* AND *Salary>Salary* Manager)** 

The join yields a table with attributes: Employee.*Name* **JAIN UNIVERSITY BCA**

45Equijoin Join - Example 

*Equijoin*: Join condition is a conjunction of *equalities*. Π*Name,CrsCode*(Student *Id=StudId* σ*Grade=‘A’ *(Transcript)) Student Transcript 

*Id Name Addr Status* 111 John ….. ….. 222 Mary ….. ….. 333 Bill ….. ….. 444 Joe ….. …..

*StudId CrsCode Sem Grade* 111 CSE305 S00 B 222 CSE306 S99 A 333 CSE304 F99 A

***The equijoin is used very***

Mary CSE306

Bill CSE304

**JAIN UNIVERSITY BCA**

***frequently since it combines related data in different relations.***

46 Schema for Student Registration System

Student (*Id, Name, Addr, Status*) 

Professor (*Id, Name, DeptId*) 

Course (*DeptId, CrsCode, CrsName, Descr*) Transcript (*StudId, CrsCode, Semester, Grade*) Teaching (*ProfId, CrsCode, Semester*) 

Department (*DeptId, Name*)

**JAIN UNIVERSITY BCA**

**47** Self-join Queries

Find Ids of all professors who taught at least two

courses in the same semester:

SELECT T1.*ProfId* 

FROM Teaching T1, Teaching T2

WHERE T1.*ProfId* = T2.*ProfId*

AND T1.*Semester* = T2.*Semester*

AND T1.*CrsCode* <> T2.*CrsCode*

*Tuple variables essential in this query!*

Equivalent to: 

π*ProfId* (σ*T1.CrsCode*≠*T2.CrsCode and* (Teaching T1 

*T1.Semester=T2.semester and T1.profId=T2.profid* Teaching T2)

**JAIN UNIVERSITY BCA**

**EXAMPLES**48

SELECT City AS city

FROM customer

WHERE Country = 'Canada';

SELECT \* FROM artist

WHERE Name LIKE 'Black%' ORDER BY Name ASC;

SELECT \* FROM artist art INNER JOIN album alb

ON art.ArtistId=alb.ArtistId

WHERE Name LIKE 'Black%'

ORDER BY art.Name ASC, alb.Title ASC;

**JAIN UNIVERSITY BCA **

SELECT art.ArtistId, art.Name, alb.AlbumId, alb.Title

FROM artist art LEFT OUTER JOIN album alb

49

ON art.ArtistId=alb.ArtistId

WHERE Name LIKE 'Black%'

ORDER BY art.Name, alb.Title;



SELECT \* FROM artist art LEFT OUTER JOIN album alb

ON art.ArtistId=alb.ArtistId

WHERE Name LIKE 'Black%'

ORDER BY art.Name, alb.Title;

JAIN UNIVERSITY BCA

50 **TUPLE RELATIONAL CALUCULUS**

• Relational calculus is a query language which is non-procedural, and instead of algebra, it uses mathematical predicate calculus.

• When applied to databases, it is found in two TRC,DRC.

**JAIN UNIVERSITY BCA**

51

JAIN UNIVERSITY BCA

• Let us define tuple variable (there exist t)£t

52

**First Name Last Name Marks**

James Jhonson 90

Jack Jill 20

Jones Jim 30

• {t | student(t) AND t.marks>50}

Display all

{t.fname t.lname | student(t) AND t.marks>50}

**JAIN UNIVERSITY BCA**

53

**JAIN UNIVERSITY BCA**

**For example, to specify the range of a tuple variable S as the Staff relation, we write:**

**Staff(S)**

**To express the query 'Find the set of all tuples S such that F(S) is true,' we can write:**

**{S | F(S)}**

Here, F is called a formula (well-formed formula, or wff in mathematical logic). For example, to express the query to 'Find the staffNo, fName, lName, position, sex, DOB, salary, and branchNo of all staff earning more than £10,000',

**we can write:**

{S | Staff(S) ∧ S.salary > 10000}

54

**JAIN UNIVERSITY BCA**

{t | TEACHER (t) and t.SALARY>20000}

- It implies that it selects the tuples from the TEACHER in such a way that the resulting teacher tuples will have the salary greater than 20000. This is an example of selecting a range of values.

{t | TEACHER (t) AND t.DEPT\_ID = 8}

- T select all the tuples of teachers name who work under Department 8.

55

**JAIN UNIVERSITY BCA**

Domain Relational Calculus

In domain relational calculus, filtering is done based on the domain of the attributes and not based on the tuple values.

{x1,x2,x3,... | condition(x1,x2,..}

**Example:**

emp( empno, ename, job, sal)

let us take domain variable as a, b,c,d

dept( deptno, dname, loc)

let us take domain variable x, y,z

**Query:**

List the name and job of the employee whose name of SCOTT and JOB CLERK a, b, c, d domain variables

**{a,c| there exist b, d( emp(abcd) and (a=‘SCOTT') and (c='CLERK'))}**

56

Example -2

**List all the loan number of all the loans with an amount greater than 1000**

**List the employees work for dname Accounting.**

**Loan\_no branc h**

100 BLOR E

**amount** 1000

**{t.ename|emp(t)**

**AND( d)dept(d) ^ dname=‘SALES’  AND emp.deptno=dept.deptno}**

200 MYS 1500

300 **{l | b,a there exists(loan(** HBL 1230**l,b,a) and a>1000)}**

**List all the branch where loan amount less than 1500**

**{b | b,a there exists(loan(a,b,a) and a>1000)}**

Obtain the names of courses enrolled by student named Mahesh

{c.name | course(c)

^ (∃s) (∃e) ( student(s)

^ enrollment(e)

^ s.name = “Mahesh”

^ s.rollNo = e.rollNo

^ **c.courseId = e.courseId }**

57

**JAIN UNIVERSITY BCA**

Get the names of students who have scored ‘S’ in all subjects they have enrolled. Assume that every student is enrolled in at least one course.

{s.name | student(s)

^ (∀e)(( enrollment(e)

^ e.rollNo = s.rollNo)

^e.grade =‘S’)}

Determine the departments that do not have any girl students

{d.dname|department(d)

^ ¬(∃ s)(student(s)

^ s.sex =‘F’

^ s.deptNo = d.deptId)