Database Management Systems

UNIT III

Learning Objectives

Normalization:

1. Functional Dependencies
2. Normal Forms-1 NF, 2NF, 3NF, BCNF
3. Join Dependencies
4. Multi-Valued Dependencies

PL/SQL Programming:

1. Introduction to PL/SQL
2. Structure of PL/SQL Block
3. PL/SQL Language
4. Operators
5. Control Structure
6. Cursors
7. Triggers
8. Procedures
9. Functions

Relational Database Design

- Focus on the issues involved in the design of a database scheme using the relational model.

Prof	*Course*	*Room*	*Max_Enrollment*	*Day*	*Time*
Smith	353	A532	40	mon	1145
Smith	353	A532	40	wed	1145
Smith	351	C320	60	tue	115
Smith	351	C320	60	thu	115
Clark	355	H940	300	tue	115
Clark	355	H940	300	thu	115
Turner	456	B278	45	mon	845
Turner	456	B278	45	wed	845
Jamieson	459	D110	45	tue	1015
Jamieson	459	D110	45	thu	1015

Undesirable Problems

Redundancy
Update Anomalies
Insertion Anomalies
Deletion Anomalies

Name	Course	Phone_No	Major	Prof	Grade
Jones	353	237-4539	Comp Sci	Smith	A
Ng	329	427-7390	Chemistry	Turner	B
Jones	328	237-4539	Comp Sci	Clark	B
Martin	456	388-5183	Physics	James	A
Dulles	293	371-6259	Decision Sci	Cook	C
Duke	491	823-7293	Mathematics	Lamb	B
Duke	356	823-7293	Mathematics	Bond	in prog
Jones	492	237-4539	Comp Sci	Cross	in prog
Baxter	379	839-0827	English	Broes	C

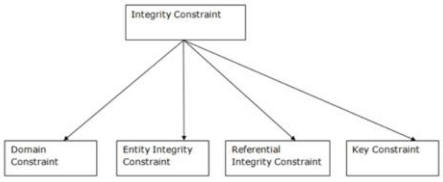
Decomposition

- A relation R is decomposed into collection of relation schemes to eliminate anomalies.
- Natural Join operations must be implemented to get the results.
- When we replace original relation with multiple schemes, referential integrity must be maintained.

STUDENT_INFO (Name, Phone-No, Major)
TRANSCRIPT (Name, Course, Grade)
TEACHER (Course, Proof)

Integrity Constraints

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.



Domain Constraints

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

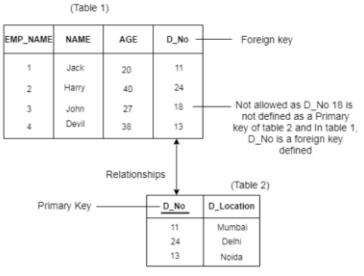
Entity Integrity Constraints

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

Referential Integrity Constraints



Key Constraints

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

Primary and Candidate Key

- The primary key uniquely identifies a record or row in the table, and it follows one key per table approach.
- The candidate key also uniquely identifies a record or row in the table, but it can have one key per table or many keys per table. The primary key is a minimal super key
- Foreign Key
- Super Key

Functional Dependencies

- If R represents an entity and the set X of attributes represents the key of R, then for any other set of attribute Y of R, X —> Y.

F1: Reflexivity: $(X \rightarrow X \text{ and } Y \subseteq Z)$
F2: Augmentation: $(X \rightarrow Y) \models (XZ \rightarrow Y, \text{ and } XZ \rightarrow YZ)$
F3: Transitivity: $(X \rightarrow Y \text{ and } Y \rightarrow Z) \models (X \rightarrow Z)$
F4: Additivity: $(X \rightarrow Y \text{ and } X \rightarrow Z) \models (X \rightarrow YZ)$
F5: Projectivity: $(X \rightarrow YZ) \models (X \rightarrow Y \text{ and } X \rightarrow Z)$
F6: Pseudotransitivity: $(X \rightarrow Y \text{ and } YZ \rightarrow W) \models (XZ \rightarrow W)$

Closure of a Set of Functional Dependencies

- The set of functional dependencies logically implied by F is called the closure of F and is written as F+.
- Let R = (A, B, C, D) and F = {A -> B, A -> C, BC -> D}.
- Let R = (A, B, C, D) and F = {A -> BC, AB -> D, B -> C}.

Trivial Functional Dependencies

- If X → Y, then Y is a subset of X. (Reflexivity)
- If X→Y, then XZ→YZ. (Augmentation)
- If X→Y and Y→Z then X→Z. (Transitive)
- If X→Y and X→Z then X→YZ. (Union)
- If X→YZ and X→Y then X→Z. (Decomposition)
- If X→Y and WY→Z then WX→ Z (Pesudotransitivity)

Second Normal Form

- Table and relation under study must be in first normal form
- All the non-prime attributes should be fully functional dependent on candidate key.

PL/SQL Programming

Print natural numbers from 1 to 5.

```
declare
   i number;
begin
   i:=1;
   loop
   dbms_output.put_line(i);
   i:=i+1;
   exit when i > 5;
   end loop;
end;
```


PL/SQL Procedure

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, …])]
{IS | AS}
BEGIN
      <procedure body>
END procedure_name;
```


PL/SQL Procedure

```
Create or replace procedure topperStudent
As
topperName student.name%type;
begin
Select name into topperName from student where marks=(select
max(marks) from student)
Dbms_output.put_line(topperName);
End;
```

PL/SQL Procedure

begin
get topperStudent;
End;

Or
exec topperStudent

PL/SQL Functions

- Functions is same as procedure except that it returns a value.

CREATE [OR REPLACE]FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [,
...])]
RETURN return_datatype
{IS | AS}
BEGIN
< function_body >
END [function_name];

PL/SQL Functions

- Functions is same as procedure except that it returns a value.

```
CREATE OR REPLACE FUNCTIONtotalCustomers
RETURN number IS
total number(2) := 0;
BEGIN
SELECT count(*) into total FROM
customers;
RETURN total;
END;
```

Calling a PL/SQL Functions

- To call a function, you simply need to pass the required parameters along with the function name and if the function returns a value, then you can store the returned value.

```
DECLARE
c number(2);
BEGIN
c := totalCustomers();
dbms_output.put_line('Total no. of
Customers: ' || c);
END;
```


PL/SQL Cursor

- A cursor is a temporary area created in the main memory when a SQL statement is executed.
- A cursor contains information on a select statement and the rows of data accessed by it.
- This temporary work area is used to store the data retrieved from the database, and manipulate this data.
- A cursor can hold more than one row, but can process only one row at a time.
- The set of rows a cursor hold is known as active set.
- There are two types of cursor:
  - Implicit
  - Explicit

Implicit Cursor

- Implicit cursors are created when we execute DMA queries like select, insert, delete, update.
- Cursor C1 IS Select Statement
  Open C1
  Fetch C1 into
  Close C1

Implicit Cursor

```
DECLARE
C_id customers.id%type;
C_name customers.name%type;
CURSOR C1 IS
 SELECT id, name from customers;
BEGIN
Open C1;
Loop
Fecth C1 INTO C.id, C_name;
EXIT WHEN C1% NotFound;
DBMS_output.put_line(c.id||''||c.name);
END Loop;
Close C1;
End;
```


PL/SQL Programming

- Procedure language extension to SQL.

```
DECLARE
    <declarations section>
BEGIN
    <executable command(s)>
EXCEPTION
    <exception handling>
END;
```