Intermediate SQL

Tags	Done
# Chapter	4
Date	@October 1, 2023
Person	 Tahsin Islam

Joins

→ Joins are 2 types

1. Inner Join (Natural Join)
2. Outer Join

1. Inner Join

- Combines two tables in a natural way
- Splits one large table into two
- Removes Redundancy and Inconsistency

- Need to use Foreign key for splitting

```
CREATE TABLE STUDENTS(
ID NUMBER PRIMARY KEY,
NAME VARCHAR2(30),
SALARY NUMBER(8,0),
DEPT VARCHAR2(6),
CONSTRAINTS FKSTU FOREIGN KEY (DEPT)
REFERENCING DEPTS);
```

```
CREATE TABLE DEPTS(
DEPT_NAME VARCHAR(6) PRIMARY KEY,
BUILDING VARCHAR2(20),
BUDGET NUMBER(10,0)_;
```

EXAMPLE:

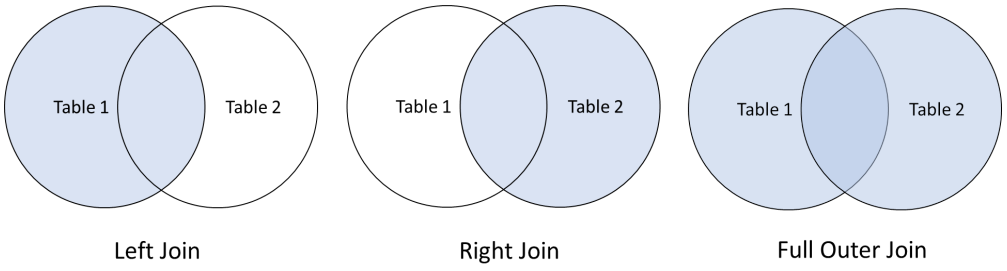
- List the names of instructors along with the course ID of the courses that they taught

```
-- Old Way

SELECT NAME, COURSE_ID
FROM STUDENTS, TAKES
WHERE STUDENT.ID = TAKES.ID;
```

```
--Using Natural Join

SELECT NAME, COURSE_ID
FROM STUDENT NATURAL JOIN TAKES;
```

2. Outer Join

- Avoids loss of information
- Only matched records are selected
- Non-matched values are replaced by NULL
- 3 types of Outer Joins:
  - Left Join ⇒ All records from the first table, only matched part from the second table
  - Right Join ⇒ All records from the second table, only matched part from the first table
  - Full Outer Join ⇒ All records from both tables. (Merge into one)



i. Left Join

```
-- ORIGINAL

SELECT NAME, COURSE_ID
FROM STUDENTS LEFT JOIN TAKES
ON STUDENT.ID = TAKES.ID;
```

```
-- ALTERNATIVE

SELECT NAME, COURSE_ID
FROM STUDENTS, TAKES
WHERE STUDENT.ID = TAKES.ID (+);
```

ii. Right Join

```
-- ORIGINAL

SELECT NAME, COURSE_ID
FROM STUDENTS Right JOIN TAKES
ON STUDENT.ID = TAKES.ID;
```

```
-- ALTERNATIVE

SELECT NAME, COURSE_ID
FROM STUDENTS, TAKES
WHERE STUDENT.ID (+) = TAKES.ID;
```

Views

What is View?

- A virtual table that presents data from one or more existing tables in a specific way.
- Doesn’t store data itself but provides a way to look at and work with the data in a particular manner

Functions of View

1. Controls security and access control of sensitive information
2. Helps to reuse the code

Creation of View

```
CREATE VIEW FACULTY AS
SELECT ID, NAME, DEPT_NAME
FROM INSTRUCTOR;
```

DML Through a View

- Most SQL implementations allow updates only on simple views:
  - The from clause has only one database relation.
  - The select clause contains only attribute names of the relation, and does not have any expressions, aggregates, or distinct specification.
  - Any attribute not listed in the select clause can be set to null.
- Condition in general:
  - Each attributes has1 - 1 correspondence of its underlying single table.
  - Not Null attributes are included

Materialized View

A materialized view is a database object that stores the results of a query in a physical table-like structure

Example:

Orders:

Customers:

OrderID	CustomerID	TotalAmount
1	101	100
2	102	200
3	101	150

CustomerID	FirstName	LastName
101	John	Doe
102	Jane	Smith

```
CREATE MATERIALIZED VIEW HighValueOrder AS
SELECT CustomerID, SUM(TotalAmount) AS TotalSpent
FROM Orders
GROUP BY CustomerID;
```

The Materialized View is:

CustomerID	TotalSpent
101	250
102	200

Benefits:

- For efficient and smooth running of the operational database
- Quick access time. (Local copy access is faster)

Transaction

- Unit of work. All or nothing property.
- Normally commit and rollback used.

Integrity Constraints

- Guard against accidental damage to the database

  Example:

  - An instructor’s name cannot be null.
  - No two instructors can have the same instructor ID.
- Introduced while designing the database schema.

Integrity Constraints on a Single Relation

- not null
- primary key
- unique
- check (P), where P is a predicate’

Example:

```
CREATE TABLE EMP
( ID NUMBER PRIMARY KEY,
NAME VARCHAR2(20) NOT NULL,
SALARY NUMBER (10,2),
```

```
SHORTNAME VARCHAR2(10),
CONSTRAINT SAL_CHEWCK CHECK (SALARY > 1000)
CONSTRAINT UNIQ_NAME UNIQUE (SHORTNAME)
);
```

Referential Integrity

- This refers to the Integrity Constraints on Multiple Relations
  - Foreign Key
  - Cascading actions (mostly supported in deletion, not in updating)
- An example:

```
create table depts
(id number primary key,
name varchar2(10)
);

create table students(
  sid number primary key,
  name varchar2(10),
  cgpa number(4,2),
  dept number,constraints fk_stu foreign key(dept)
  references depts on delete cascade
);
```

Cascading Delete

- It ensures the consistency of data relationships
- If a record in the parent table is deleted, all related records in the child table(s) will also be automatically deleted.

Violation during transition

- Consider:

```
create table person(
  ID char(10) primary key,
name char(40),
mother char(10),
father char(10),
foreign key father referencing person,
foreign key mother referencing person)
```

  - How to insert a tuple without causing constraint violation?
    - Insert father and mother of a person before inserting person
    - OR, set father and mother to null initially, update after inserting all persons (run a process to it..)
    - OR defer constraint checking

Date and Time Types in SQL

2 types of date related basic datatype:

1. Date
2. Timestamp

- It allows to store point-in-time values that include both date and time with a precision of one second
- Format: `DD-MM-YY` Example: 20-Sep-18

DATE: TO_CHAR() function

Format Specifier	Meaning
YYYY	4-digit year
YY	2-digit year
MONTH	Month name (January - December)
MM	Month (1 - 12)
DD	Day (1 - 31)
DY	Abbreviated day (Sun - Sat)
Day	Day (Sunday, Monday)
HH24	Hour (0 - 23)
HH or HH12	Hour (1 - 12)
MI	Minutes (0 - 59)
SS	Seconds (0 - 59)

```
SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD') FROM dual

Output: 2012-07-19
```

DATE: ToDate function

- Is to a convert a string to DATE Datatype

```
SELECT TO_DATE( '5 Jan 2017', 'DD MON YYYY' )
FROM dual;

--Output: 05-JAN-17

--Now you can use any date function on it:

select to_date('5 Feb 2018','DD MON YYYY')+100
FROM DUAL;

--Output: 16-MAY-18
```

Built-in functions on Date

- ADD_MONTHS(date, n):

  `SELECT ADD_MONTHS(SYSDATE,13) FROM DUAL;`

  Output: 04-AUG-21]

- NEXT_DAY(date,NameOfDay):

  `SELECT NEXT_DAY(SYSDATE,'FRIDAY') FROM DUAL;`

  Output: 10-JUL-20

  • MONTHS_BETWEEN(date1, date2):

`SELECT MONTHS_BETWEEN('18-FEB-92','14-MAY-89')FROM DUAL;`

Output: 33.1290323

Large Object Types

Binary Large Object (blob)

- A large collection of uninterpreted binary data.

Character Large Object (clob)

- A large collection of character data.

Create Table Extentions

- Need similar table structure often
- We seek a shortcut to lend the structure from existing one.
- structure can be copied as well as data.
- Default it copies both structure and data.
- To exclude data use a clever trick in where clause.

```
CREATE TABLE NEW_STUDENTS AS SELECT ID, NAME, GPA FROM STUDENTS;
```

Authorization

- Role-based Access Control.
- It is used to distribute a large system into a number of users with different access controls.
- Normally SELECT, UPDATE, DELETE, INSERT are the privileges on a database for tables or views.
- Additionally, EXECUTE is another privilege for sub-program (will be covered later).
- Role is created and customized (it is like a package of actions)
- Then the owner grant (opposite is itermrevoke) the created role to a specific user.
- Roles can be defined using another role.

Grant privileges: Syntax

```
grant <privilege list>
on <relation name or view name>
to <user/role list>;

--privilege list is:
SELECT, UPDATE, INSERT, DELETE, (AND EXECUTE)
```

Role-Based Access Control: Overview

Given Scenario: Result Processing System (RPS)

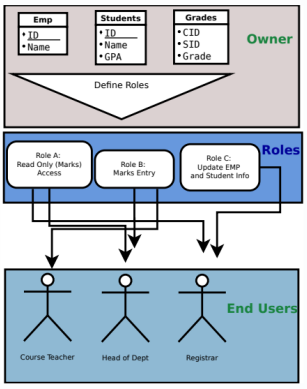


Figure: Role-based Access Control: Concept