ORACLE SQL

Lesson 04: Constraints, Adv. Group by, Adv. Subqueries, Managing other db objects..

Lesson Objectives

To understand the following topics:

- Constraints
- Adv Group by
- Adv subqueries
- Other DB objects

4.1: Constraints

What is Data Integrity?

Data Integrity:

- “Data Integrity” allows to define certain “data quality requirements” that must be met by the data in the database.
- Oracle uses “Integrity Constraints” to prevent invalid data entry into the base tables of the database.
  - You can define “Integrity Constraints” to enforce the business rules you want to associate with the information in a database.
  - If any of the results of a “DML statement” execution violate an “integrity constraint”, Oracle rolls back the statement and returns an error.

Advantages

Advantages of Integrity Constraints:

- Integrity Constraints have advantages over other alternatives. They are:
  - Enforcing “business rules” in the code of a database application.
  - Using “stored procedures” to completely control access to data.
  - Enforcing “business rules” with triggered stored database procedures.

Applying Constraints

Constraints can be defined at

- Column Level

```
CREATE TABLE  tablename
(column datatype  [DEFAULT expr] [column_constraint] ,
……)
```

- Tabl

```
CREATE TABLE  tablename
(column datatype,
 column datatype
……
 [CONSTRAINT constraint_name] constraint_type
(column,…))
```

4.1: Constraints

Types of Integrity Constraints

Let us see the types of Data Integrity Constraints:

- Nulls
- Unique Column Values
- Primary Key Values
- Referential Integrity

NOT NULL Constraint

The user will not be allowed to enter null value.

For Example:

- A NULL value is different from a blank or a zero. It is used for a quantity that is “unknown”.
- A NULL value can be inserted into a column of any data type.

```
CREATE TABLE  student_master
(student_code  number(4) NOT NULL,
 dept_code   number(4) CONSTRAINT dept_code_nn
                                   NOT  NULL );
```

DEFAULT clause

If no value is given, then instead of using a “Not Null” constraint, it is sometimes useful to specify a default value for an attribute.

For Example:

- When a record is inserted the default value can be considered.

```
CREATE TABLE  staff_master(
Staff_Code number(8) PRIMARY KEY,
Staff_Name varchar2(50) NOT NULL,
Staff_dob date,
Hiredate date DEFAULT sysdate,
…..)
```

UNIQUE constraint

The keyword UNIQUE specifies that no two records can have the same attribute value for this column.

For Example:

```
CREATE TABLE student_master
(student_code number(4),
 student_name varchar2(30),
 CONSTRAINT stu_id_uk  UNIQUE(student_code )) ;
```

PRIMARY KEY constraint

The Primary Key constraint enables a unique identification of each record in a table.

For Example:

```
CREATE TABLE Staff Master
 (staff_code  number(6)
 CONSTRAINT staff_id_pk PRIMARY KEY
  staff_name varchar2(20)
   ………);
```

CHECK constraint

CHECK constraint allows users to restrict possible attribute values for a column to admissible ones.

For Example:

```
      CREATE TABLE staff_master
( staff_code number(2),
  staff_name  varchar2(20),
  staff_sal   number(10,2) CONSTRAINT staff_sal_min
                           CHECK (staff_sal >1000),
   …..) ;
```

FOREIGN KEY constraint

The FOREIGN KEY constraint specifies a “column” or a “list of columns” as a foreign key of the referencing table.
The referencing table is called the “child-table”, and the referenced table is called “parent-table”.

For Example:

```
                 CREATE TABLE student_master
           (student_code number(6) ,
            dept_code number(4) CONSTRAINT stu_dept_fk
                              REFERENCES
department_master(dept_code),
            student_name varchar2(30) );
```

4.2: Adv. subqueries

Column Comparisons

Multiple-column comparisons involving subqueries can be:

- Nonpairwise comparisons
- Pairwise comparisons

Pairwise Comparison Subquery

Display the details of the employees who are managed by the same manager and work in the same department as employees with the first name of “John.”

```
SELECT employee_id, manager_id, department_id
FROM  empl_demo
WHERE  (manager_id, department_id) IN
                    (SELECT manager_id, department_id
                     FROM empl_demo
                     WHERE first_name = 'John')
AND first_name <> 'John';
```

Nonpairwise Comparison Subquery

Display the details of the employees who are managed by the same manager as the employees with the first name of "John" and work in the same department as the employees with the first name of "John."

```
SELECT  employee_id, manager_id, department_id
FROM    empl_demo
WHERE   manager_id IN
                  (SELECT manager_id
                   FROM empl_demo
                   WHERE first_name = 'John')
AND department_id IN
                 (SELECT department_id
                  FROM empl_demo
                  WHERE first_name = 'John')
AND first_name <> 'John';
```

Scalar Subquery Expressions

A scalar subquery expression is a subquery that returns exactly one column value from one row.

Scalar subqueries can be used in:

The condition and expression part of DECODE and CASE

All clauses of SELECT except GROUP BY

The SET clause and WHERE clause of an UPDATE statement

4.2: Adv. subqueries

Scalar Subqueries: Examples

- Scalar subqueries in `CASE` expressions:
-

```
SELECT employee_id, last_name,
       (CASE
        WHEN department_id =
                 (SELECT department_id
                  FROM departments
     WHERE location_id = 1800)
         THEN 'Canada' ELSE 'USA' END) location
FROM   employees;
```

- Sca

```
SELECT   employee_id, last_name
FROM     employees e
ORDER BY (SELECT department_name
          FROM departments d
           WHERE e.department_id = d.department_id);
```

4.2: Adv. subqueries
Correlated Query

When a sub query references a column from the table referred to in the parent query it is known as correlated query

A correlated subquery answers a multiple-part question whose answer depends on the value in each row processed by the parent statement

Correlated Query Example

List the employees earning more than average salaries in their own department

```
select ename,sal,deptno
from emp a
where sal > [ A query which returns the avg salary of the department in which the employee of the outer query is working]
```

```
select ename,sal,deptno
from emp a
where sal > (select avg(sal) from emp b
where b.deptno = a.deptno)
order by deptno
```

Using Exists and Not Exists

Exists	Returns TRUE if the subquery returns a single row satisfying the condition
Not Exists	Returns TRUE if the subquery does not return any row

4.2: Adv. subqueries
Example

List the departments without employees

```
select * from dept d
where not exists  (select deptno from emp e where e.deptno=d.deptno)
```

List the departments with employees

```
select * from dept d
where  exists  (select deptno from emp e where e.deptno=d.deptno)
```

Using Correlated UPDATE

Denormalize the EMPL6 table by adding a column to store the department name.
Populate the table by using a correlated update.

```
ALTER TABLE empl6
ADD(department_name VARCHAR2(25));
```

```
UPDATE empl6 e
SET    department_name =
           (SELECT department_name
      FROM   departments d
      WHERE  e.department_id = d.department_id);
```

4.2: Adv. subqueries

Using Correlated DELETE

Use a correlated subquery to delete only those rows from the EMPL6 table that also exist in the EMP_HISTORY table.

```
DELETE FROM empl6 E
WHERE employee_id =
          (SELECT employee_id
           FROM   emp_history
           WHERE  employee_id = E.employee_id);
```

4.2: Adv. subqueries

Inline view

A subquery in the FROM or column list clause of the Select statement is known as inline view

Columns of inline view can be used in the outer query

Examples of inline view

List the employees earning more than the average salary. Also display the average salary

```
select ename, sal, average_salary
from emp , (select avg(sal) average_salary from emp )
where sal > average_salary
```

Views

A view is a stored query
A view takes the output of the query and treats it as a table.
Used for storing complex queries for easy representation
Oracle stores the definition of view
It does not contain data
It is known as virtual table
The definition is expanded at runtime when it is used

4.2: Adv. subqueries

Views

```
Create or Replace view <viewname) (column_list)
as <query>
with check option constraint
with READ ONLY

 Create view emp_view as
 select empno,ename,deptno,sal from emp;

 Create view emp_dept_view as
 select empno,ename,job,dname
 from emp e, dept d
 where e.deptno = d.deptno
```

4.2: Adv. subqueries

Views

Views are derived from base tables and hence have many similarities.

They can be described and queried

With some restrictions we can insert into, update or delete data from views

All the operations are performed on the base tables of the view and they affect the actual data of the base table subject to integrity constraint and triggers

4.2: Adv. subqueries
Views

```
Desc emp_view

Select * from emp_view

Insert into emp_view
Values(101,’Tom’,20, 4500);
```

4.2: Adv. subqueries

View Examples

```
Create view  dept_summary(dept_name,emp_count,total_salary, maximum_sal, minimum_sal) as
select dname,count(*),sum(sal),max(sal),min(sal)
from emp e, dept d
where d.deptno = e.deptno
group by dname;
```

4.2: Adv. subqueries

How Views are used

To provide table level security by restricting data to predetermined rows or columns
Hides complexity
Simplifies statements for users
Save complex queries

4.2: Adv. subqueries

Views – Check option and Read Only

```
Create view emp_dept_10 as
 select *  from emp
 where deptno = 10
 with check option
```

DML must confirm to condition specified in where clause

```
Create OR REPLACE view emp_dept_10 as
 select *  from emp
 where deptno = 10
 with READ ONLY;
```

DML cannot be performed

Rules for Performing DML Operations on a View

- You can usually perform DML operations on simple views.
- You cannot remove a row if the view contains the following:
  - Group functions
  - A `GROUP BY` clause
  - The `DISTINCT` keyword
  - The pseudocolumn `ROWNUM` keyword

Rules for Performing DML Operations on a View

You cannot modify data in a view if it contains:

- Group functions
- A `GROUP BY` clause
- The `DISTINCT` keyword
- The pseudocolumn `ROWNUM` keyword
- Columns defined by expressions

Rules for Performing DML Operations on a View

You cannot add data through a view if the view includes:

- Group functions
- A `GROUP BY` clause
- The `DISTINCT` keyword
- The pseudocolumn `ROWNUM` keyword
- Columns defined by expressions
- `NOT NULL` columns in the base tables that are not selected by the view

4.2: Adv. subqueries

Removing a View

```
Drop view <view_name>

Drop view dept_summary;
```

4.2: Adv. subqueries

Updateable View Restrictions

A view can be updateable if it does not contain :

- Set operator
- Distinct clause
- Aggregate or Analytical functions
- Group by clause
- Subquery in select list
- Joins (with some exceptions)

4.2: Adv. subqueries

Rules for updateable join view

The DML statement must affect only one table underlying in the join (known as key-preserved table)
For an UPDATE statement, all columns updated must be extracted from a key-preserved table.
For a DELETE statement, the join can have one and only one key-preserved table
For an INSERT statement, all columns into which values are inserted must come from a key-preserved table

4.2: Adv. subqueries

Updatable view example

```
create or replace view emp_dept_upd
as select EMPNO, ENAME,JOB,MGR,HIREDATE, SAL,
COMM, e.DEPTNO, dname
from emp e, dept d
where e.deptno = d.deptno;
SELECT column_name, updatable
 FROM user_updatable_columns
 WHERE table_name = 'EMP_DEPT_UPD‘;
insert into emp_DEPT_UPD (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL,COMM, DEPTNO)
values(1,'HAPPY','CLERK',7782,'01-JAN-02',1500,NULL,10);
```

Index

Index helps to locate information faster

Indexes can be created on column(s) of a table to speed up execution of SQL statements on that table

Oracle index provides a faster access path to table data

Indexes are the primary means of reducing disk I/O when properly used.

A useful tool for application tuning used by developers and DBA’s

4.3: DB Objects

Indexes…

Oracle provides several indexing schemes , the most common used is B-tree structure
Indexes are automatically created with the same name by Oracle when Primary and Unique constraints are created.
Oracle maintains and uses indexes on its own
Columns containing NULL values are not indexed

4.3: DB Objects
Index examples

```
Create unique index <index_name> on table(column_list) asc/desc

Create index empidx on emp1(empno);

Create index emp_dept on emp1(deptno,ename);
create  index emp_job on emp(job)
```

4.3: DB Objects

When to Create an Index

- A column contains a wide range of values
- A column contains a large number of null values
- One or more columns are frequently used together in a Where clause or a join condition
- The table is large and most queries are expected to retrieve less than 2 to 4 percent of the rows

When not to create an index

- The table is small
- The columns are not often used as a condition in the query
- Most queries are expected to retrieve more than 2 to 4 percent of the rows in the table
- The table is updated frequently
- The indexed columns are referenced as part of an expression

4.3: DB Objects

Removing an Index

```
Drop index <index_name>

Drop index empidx;
```

4.3: DB Objects

Sequences

Sequences are used for generating unique sequential series of numbers
Useful in multi-user environment
Reduces serialization where the statements of two transactions must generate sequential numbers at the same time
A new sequence number can be generated or the current sequence number can be used by using NEXTVAL or CURRVAL
They are generated independently of tables
Used for generating unique primary keys
The sequence number is incremented independent of transaction committing or rolling back

4.3: DB Objects

Sequence

```
Create sequence <sequence_name>
Increment by <value>
Start with <value>
Maxvalue <value> /nomaxvalue
Minvalue <value> / nominvalue
Cycle /nocycle
CACHE/NOCACHE
```

4.3: DB Objects

Sequence example

```
create sequence seq_deptno
start with 50
increment by 10
maxvalue 500 ;

insert into dept VALUES (seq_deptno.NEXTVAL,'HUMAN RESOURCE','NEW YORK');

select seq_deptno.CURRVAL from dual;
```

4.3: DB Objects

NEXTVAL and CURRVAL Pseudocolumns

NEXTVAL returns the next available sequence value. It returns a unique value every time it is referenced, even for different users.
CURRVAL obtains the current sequence value.
NEXTVAL must be issued for that sequence before CURRVAL contains a value.

4.3: DB Objects

Caching Sequence Values

Caching sequence values in memory gives faster access to those values.

Gaps in sequence values can occur when:

A rollback occurs

The system crashes

A sequence is used in another table

Modifying a Sequence

Change the increment value, maximum value, minimum value, cycle option, or cache option:

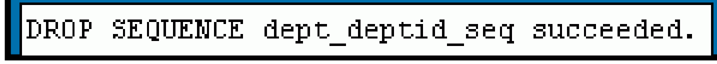
```
ALTER SEQUENCE dept_deptid_seq
               INCREMENT BY 20
               MAXVALUE 999999
               NOCACHE
               NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq succeeded.
```

Guidelines for Modifying a Sequence

You must be the owner or have the ALTER privilege for the sequence.
Only future sequence numbers are affected.
The sequence must be dropped and re-created to restart the sequence at a different number.
Some validation is performed.
To remove a sequence, use the DROP statement:

```
DROP SEQUENCE dept_deptid_seq;
```



```
DROP SEQUENCE dept_deptid_seq succeeded.
```

4.3: DB Objects

Synonyms

Synonyms are alias name for table,view,sequence,procedures,functions, package,snapshots
Hides the owner and name of the object
Provides location transparency in distributed databases.
Simplifies usage of SQL statements by the users
Provides data independence
Synonyms can be Public or Private

4.3: DB Objects

Synonym

```
Create [public] synonym <synonym_name>
For <object_name>

create  synonym balance for leave_balance;
```

4.3: DB Objects

Removing Synonym

```
Drop synonym <synonym_name>

Drop synonym balance ;
```

SUMMARY

- In this lesson you have learnt,
  - Constraints
  - Adv Group by
  - Adv subqueries
  - DB Objects

Review Question

❖ Question 1: Which constraint will not allow to enter null values?

❖ Question 2: Indexes can be created ________ or _________

❖ Question 3: __________ obtains the current sequence value