Persistent

NUGGET 9:
Working with Other Database Objects

Persistent University

Key Learning Points

1. Views
2. Sequences
3. Indexes
4. Synonyms

Persistent

Database Design

Employee Table:

Name	Null?	Type
employee_id	Not null	Number(6)
FirstName		Varchar2(20)
last_name	Not null	Varchar2(25)
Email		Varchar2(25)
Phonenumber		Varchar2(20)
Hiredate		Date
Jobid		Varchar2(10)
Salary		Number(8,2)
manager_id		Number(6)
department_id		Number(4)

Department Table:

Name	Null?	Type
department_id	Not null	Number(4)
department_name	Not null	Varchar2(30)
manager_id		Number(6)

View

- A view is a logical table based on a table or another view.
- A view contains no data of its own but is like window through which data from the table can be viewed or changed. The tables on which a view is based are called base tables.
- The view is stored as a select statement in data dictionary.
- You can use views in almost the same way as tables. You can query, update, insert into and delete from views, just as you can do with standard tables.
- Advantages:
  - Views restricts access to the data by displaying selective columns from the table.
  - Views can be used to make simple queries to retrieve the results of complicated queries.
  - Views provide data independence for ad hoc users and application programs.
  - One view can be used to retrieve data from several tables.
  - Views provide group of users access to data according to their particular criteria.

View

- Syntax :

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW viewname
[(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- In the syntax :
  - OR REPLACE re-creates the view if it already exists.
  - FORCE creates the view regardless of whether or not the base tables exist.
  - NOFORCE creates the view only if the base tables exist. NOFORCE is Default option.
  - alias specifies names for the expressions selected by the view's query.
  - subquery is a complete SELECT statement.
  - WITH CHECK OPTION specifies that only rows accessible to the view can be inserted or updated.
  - constraint is the name assigned to the CHECK OPTION constraint.
  - WITH READ ONLY ensures that no DML operations can be performed on this view.

View

- Simple View
  - A Simple view is created on top of one table only.
  - Example :

```
CREATE VIEW emp_view AS
SELECT last_name, salary*12 annual_salary
FROM employee
WHERE department_id = 20;
```

- Complex view
  - A Complex view is created on more than one base table or view. It can be called as ‘Join View’.
  - It can contain various single row/multiple row functions, set operators, Group by clause and Having clause as well.
  - Example :

```
CREATE VIEW Emp_dept AS
SELECT e.employee_id, e.last_name, e.department_id, e.salary, d.department_name
FROM Employee e, Department d                    /* JOIN operation */
WHERE e.Department_id = d.Department_id;
```

Retrieving Data from a View

- User can retrieve data from a view as he/she can from any table.
- User can display either the contents of the entire view or just specific rows and columns.
- Examples:
  - SELECT * FROM emp_dept;
  - SELECT * from emp_view

    Where annual salary > 50000;
  - SELECT employee_id, last_name, salary from emp_dept

    Where department_id=20;

Modifying a View

- Views that involve joins can be modified with some restrictions.
- Considering previous example, there are restrictions on modifying either the EMP or the DEPT base table through this view.
- A modifiable join view is a view that contains more than one table in the top-level FROM clause of the SELECT statement, and that does not contain any of the following:
  - DISTINCT operator
  - Aggregate functions: AVG, COUNT, GLB, MAX, MIN, STDDEV, SUM, or VARIANCE
  - Set operations: UNION, UNION ALL, INTERSECT, MINUS
  - GROUP BY or HAVING clauses
  - START WITH or CONNECT BY clauses
  - ROWNUM pseudocolumn

Modifying a View

- Any UPDATE, INSERT, or DELETE statement on a join view can modify only one underlying base table.
  - Example:

```
UPDATE Emp_dept
SET salary= salary * 1.10
WHERE department_id=10;
```

- Insert into a view:
  - Insert into view adds data in base tables and same would reflect in the view itself.
  - Example:

```
INSERT INTO emp_dept (employee_id, last_name, department_id, salary)
VALUES (780, 'dev', 10, 90000);
```

The above insert statement attempts to insert a record into employee table.

Removing a View

- Syntax:

  DROP VIEW view_name

- Example:

  DROP VIEW emp_dept;

- Removing views:
  - DROP VIEW statement removes the view definition from the database.
  - Dropping views has no effect on the tables on which the view was based.
  - Views or other applications based on deleted views become invalid.

Sequence

- A sequence is a user created database object that can be shared by multiple users to generate unique integers.
- A typical usage for sequences is to create a primary key value, which must be unique for each row.
- The sequence is generated and incremented (or decremented) by an internal Oracle routine.
- This can be a time-saving object because it can reduce the amount of application code needed to write a sequence-generating routine.
- Sequence numbers are stored and generated independently of tables. Therefore, the same sequence can be used for multiple tables.
- Syntax :

```
CREATE SEQUENCE sequence
[INCREMENT BY n]
[START WITH n]
[{MAXVALUE n | NOMAXVALUE}]
[{MINVALUE n | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE n | NOCACHE}];
```

Sequence

- In the Syntax:
  - sequence is the name of the sequence generator
  - INCREMENT BY n specifies the interval between sequence numbers where n is an integer (If this clause is omitted, the sequence increments by 1.)
  - START WITH n specifies the first sequence number to be generated (If this clause is omitted, the sequence starts with 1.)
  - MAXVALUE n specifies the maximum value the sequence can generate
  - NOMAXVALUE specifies a maximum value of 10^27 for an ascending sequence and –1 for a descending sequence (This is the default option.)
  - MINVALUE n specifies the minimum sequence value
  - NOMINVALUE specifies a minimum value of 1 for an ascending sequence and – (10^26) for a descending sequence (This is the default option.)
  - CYCLE | NOCYCLE specifies whether the sequence continues to generate values after reaching its maximum or minimum value (NOCYCLE is the default option.)
  - CACHE n | NOCACHE specifies how many values the Oracle server preallocates and keep in memory (By default, the Oracle server caches 20 values.)

Sequence

- Example:

```
CREATE SEQUENCE seq1
START WITH 1
INCREMENT BY 1
MAXVALUE 1000
NOCYCLE;
```

  - The above statement creates a sequence seq1 which will start with 1 and increment by 1.
  - It’s MAXVALUE is 1000 i.e. after 1000 numbers are generated it will stop because we have mentioned NOCYCLE.
  - If specified NOMAXVALUE, the sequence will generate infinite numbers.

Sequence

- 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.
  - Example:

```
Select seq1.nextval from dual;

SEQ1

---------

1
```

- Using sequence: to insert a record in DEPARTMENT table.

```
Insert into department (department_id,department_name,manager_id)

values (seq1.nextval,'Technical',20);
```

Sequence

- Altering a sequence
  - Example:

    ALTER SEQUENCE seq1

    MAXVALUE 200;
  - Except Starting Value, you can alter any other parameter of a sequence. To change START WITH parameter you have to drop and recreate the sequence.

- Removing a Sequence
  - To remove a sequence from the data dictionary, use the DROP SEQUENCE statement.
  - Syntax:

    DROP SEQUENCE sequencename;
  - Example:

    DROP SEQUENCE seq1;

Index

- Index is a schema object that can speed up the retrieval of rows by using a pointer.
- If you do not have an index on the column, then a full table scan occurs.
- Indexes can be created explicitly or automatically.
- Index Can reduce disk I/O by using a rapid path access method to locate data quickly.
- Index is independent of the table it indexes.
- Index is used and maintained automatically by the Oracle server.
- When you drop a table corresponding indexes are also dropped.
- Syntax:

  CREATE INDEX indexname ON table (column, column]...);

- In the syntax, column is the name of the column in the table to be indexed.
- Example:

  CREATE INDEX empname_idx ON employee (last_name);

Index

- When to Create an Index:

  User should create an index if

  - Column contains wide range of values.
  - Column contains large number of null values.
  - One or more columns are frequently used in Join or Where clause.
  - The table is large and most queries are expected to retrieve only 2 to 4% of data.

  Note :

  - More indexes on a table does not mean faster queries.
  - The more indexes you have associated with a table, the more effort the Oracle server must make to update all the indexes after a DML operation.

- Removing an Index:
  - Syntax: DROP INDEX indexname;
  - Example: DROP INDEX empname_idx;

  Note: If you drop a table, indexes and constraints are automatically dropped, but views and sequences remain.

Function based index

- Function-based indexes allow you to create an index based on a function or expression. The value of the function or expression is specified by the person creating the index and is stored in the index.

- Syntax: CREATE [UNIQUE] INDEX index_name

  ON table_name (function1, function2, ... function_n)

  - UNIQUE indicates that the combination of values in the indexed columns must be unique.
  - table_name is table on which user want to create the index.
  - function1, function2, ... function_n are used in the index.

- Example: CREATE INDEX emp_idx

  ON employee (UPPER(last_name));

  In this example, we've created an index based on the uppercase evaluation of the last_name field.

Note: Be sure that the Oracle optimizer uses this index when executing your SQL statements, be sure that UPPER(last_name) does not evaluate to a NULL value.

Synonym

- To refer to a table owned by another user, you need to prefix the table name with the name of the user who created it followed by a period. Creating a synonym eliminates the need to qualify the object name with the schema.
- A synonym is an alias or alternate name for a table, view, sequence, or other schema object.
- They are used mainly to make it easy for users to access database objects owned by other users.
- This method can be especially useful with lengthy object names, such as views.
- Syntax: CREATE [OR REPLACE] [PUBLIC] SYNONYM synonymname FOR object;
- In the syntax:
  - OR REPLACE re-creates the synonym if it already exists.
  - PUBLIC creates a synonym accessible to all users but each user must have appropriate privileges on the underlying object in order to use the synonym.
  - object identifies the object for which the synonym is created.

Synonym

- Example: The database administrator can create a public synonym accessible to all users. The following example creates a public synonym named DEPT for user1's DEPARTMENTS table:

```
CREATE PUBLIC SYNONYM dept

FOR user1.department;
```

- Removing a Synonym:
  - Syntax:

    DROP [PUBLIC] SYNONYM synonym_name [force];
  - In the syntax:
    - PUBLIC Allows you to drop a public synonym.
    - Force will drop the synonym even if it has dependencies.
  - Example:

    DROP PUBLIC SYNONYM dept;

Reference Material: Sites

http://www.oracle-dba-online.com/sql/create_and_manage_views.htm

http://www.java2s.com/Tutorial/Oracle/0160__View/CreatingandUsingaView.htm

https://docs.oracle.com/cd/B10501_01/server.920/a96521/views.htm

http://www.techonthenet.com/oracle/indexes.php

http://www.tutorialspoint.com/sql_certificate/creating_other_schema_objects.htm

Session 9: Summary

- With this we have come to an end of our 9th session where we discussed about
  - Types of View.
  - Advantages of using View.
  - How to modify View?
  - How to use Sequence?
  - When and how to use Index?
  - What is composite and function based Index.
  - Usage of Synonym.

Persistent

Thank you!

Persistent University