Creating sequences, synonyms, and indexes

Sequences

A sequence:

- Can automatically generate unique numbers
- Is a shareable object
- Can be used to create a primary key value
- Replaces application code
- Speeds up the efficiency of accessing sequence values when cached in memory

A sequence is a user-created database object that can be shared by multiple users to generate integers.

You can define a sequence to generate unique values or to recycle and use the same numbers again.

A typical usage for sequences is to create a primary key value, which must be unique for each row. A 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 independent of tables. Therefore, the same sequence can be used for multiple tables.

CREATE SEQUENCE Statement: Syntax

Define a sequence to generate sequential numbers automatically:

```
CREATE SEQUENCE [ schema. ] sequence
  [ { INCREMENT BY | START WITH } integer
  | { MAXVALUE integer | NOMAXVALUE }
  | { MINVALUE integer | NOMINVALUE }
  | { CYCLE | NOCYCLE }
  | { CACHE integer | NOCACHE }
  | { ORDER | NOORDER }
];
```

EX: CREATE SEQUENCE DEPT_SEQ

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.)
ORDER	Specify ORDER to guarantee that sequence numbers are generated in order of request. This clause is useful if you are using the sequence numbers as timestamps.
NOORDER	Specify NOORDER if you do not want to guarantee that sequence numbers are generated in order of request. This is the default.
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 pre-allocates and keeps in memory (By default, the Oracle server caches 20 values.)

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 280
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
```

```
sequence DEPT_DEPTID_SEQ created.
```

Do not use the CYCLE option if the sequence is used to generate primary key values

Note: The sequence is not tied to a table. Generally, you should name the sequence after its intended use. However, the sequence can be used anywhere, regardless of its name.

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.

```
SELECT DEPT_SEQ.NEXTVAL FROM DUAL

SELECT DEPT_SEQ.CURRVAL FROM DUAL
```

Rules for Using NEXTVAL and CURRVAL

You can use NEXTVAL and CURRVAL in the following contexts:

- The SELECT list of a SELECT statement that is not part of a subquery
- The SELECT list of a subquery in an INSERT statement
- The VALUES clause of an INSERT statement
- The SET clause of an UPDATE statement

You cannot use NEXTVAL and CURRVAL in the following contexts:

- The SELECT list of a view
- A SELECT statement with the DISTINCT keyword
- A SELECT statement with GROUP BY, HAVING, or ORDER BY clauses
- A subquery in a SELECT, DELETE, or UPDATE statement

Example

```
INSERT INTO departments(department_id,
            department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
```

```
1 rows inserted
```

The sequences is used in Insert Statement usually

- The DEFAULT expression can include the sequence pseudocolumns CURRVAL and NEXTVAL, as long as the sequence exists and you have the privileges necessary to access it.

```
CREATE SEQUENCE s1 START WITH 1;
CREATE TABLE emp (a1 NUMBER DEFAULT s1.NEXTVAL NOT
NULL, a2 VARCHAR2(10));
INSERT INTO emp (a2) VALUES ('john');
INSERT INTO emp (a2) VALUES ('mark');
SELECT * FROM emp;
```

```
sequence S1 created.
table EMP created.
1 rows inserted.
1 rows inserted.
A1 A2
-- ----------
 1 john
 2 mark
```

This is not recommended

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

Gaps in the Sequence

Although sequence generators issue sequential numbers without gaps, this action occurs independently of a commit or rollback. Therefore, if you roll back a statement containing a sequence, the number is lost.

Another event that can cause gaps in the sequence is a system crash. If the sequence caches values in memory, those values are lost if the system crashes.

Because sequences are not tied directly to tables, the same sequence can be used for multiple tables. However, if you do so, each table can contain gaps in the sequential numbers.

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;
```

```
sequence DEPT_DEPTID_SEQ altered.
```

If you reach the MAXVALUE limit for your sequence, no additional values from the sequence are allocated and you will receive an error indicating that the sequence exceeds the MAXVALUE. To continue to use the sequence, you can modify it by using the ALTER SEQUENCE statement.

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;
sequence DEPT_DEPTID_SEQ dropped.
```

- You must be the owner or have the ALTER privilege for the sequence to modify it. You must be the owner or have the DROP ANY SEQUENCE privilege to remove it.
- Only future sequence numbers are affected by the ALTER SEQUENCE statement.
- The START WITH option cannot be changed using ALTER SEQUENCE. The sequence must be dropped and re-created to restart the sequence at a different number.
- Some validation is performed. For example, a new MAXVALUE that is less than the current sequence number cannot be imposed.

```
ALTER SEQUENCE dept_deptid_seq
      INCREMENT BY 20
      MAXVALUE 90
      NOCACHE
      NOCYCLE;
```

- The error:

```
SQL Error: ORA-04009: MAXVALUE cannot be made to be less than the current value
04009. 00000 - "MAXVALUE cannot be made to be less than the current value"
*Cause:   the current value exceeds the given MAXVALUE
*Action:  make sure that the new MAXVALUE is larger than the current value
```

Sequence Information

- The USER_SEQUENCES view describes all sequences that you own.

```
DESCRIBE user_sequences
```

```
DESCRIBE user_sequences
Name            Null     Type
--------------- -------- -------------
SEQUENCE_NAME   NOT NULL VARCHAR2(128)
MIN_VALUE                NUMBER
MAX_VALUE                NUMBER
INCREMENT_BY    NOT NULL NUMBER
CYCLE_FLAG               VARCHAR2(1)
ORDER_FLAG               VARCHAR2(1)
CACHE_SIZE      NOT NULL NUMBER
LAST_NUMBER     NOT NULL NUMBER
PARTITION_COUNT          NUMBER
SESSION_FLAG             VARCHAR2(1)
KEEP_VALUE               VARCHAR2(1)
```

- Verify your sequence values in the USER_SEQUENCES data dictionary table.

```
SELECT   sequence_name, min_value, max_value,
         increment_by, last_number
FROM     user_sequences;
```

Synonyms

A synonym

- Is a database object
- Can be created to give an alternative name to a table or to an other database object
- Requires no storage other than its definition in the data dictionary
- Is useful for hiding the identity and location of an underlying schema object

Creating a Synonym for an Object

Simplify access to objects by creating a synonym (another name for an object). With synonyms, you can:

- Create an easier reference to a table that is owned by another user
- Shorten lengthy object names

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

To refer to a table that is 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 and provides you with an alternative name for a table, view, sequence, procedure, or other objects. This method can be especially useful with lengthy object names, such as views.

In the syntax:

PUBLIC	Creates a synonym that is accessible to all users
synonym	Is the name of the synonym to be created
object	Identifies the object for which the synonym is created

Guidelines

- The object cannot be contained in a package.
- A private synonym name must be distinct from all other objects that are owned by the same user.
- To create a PUBLIC synonym, you must have the CREATE PUBLIC SYNONYM system privilege.

hr connection

0.02 seconds

Worksheet | Query Builder

```
--Creating Private SYNONYM E  for EMPLOYEES table
CREATE SYNONYM E
FOR EMPLOYEES;


--so you can use the SYNONYM now to call the table
SELECT * FROM e;

-- the SYNONYM info in dictionary view called user_SYNONYMS/ ALL_SYNONYMS
select * from user_SYNONYMS

--you can drop the SYNONYM
DROP SYNONYM E;
```

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

```
CREATE PUBLIC SYNONYM  dept
FOR    alice.departments;
```

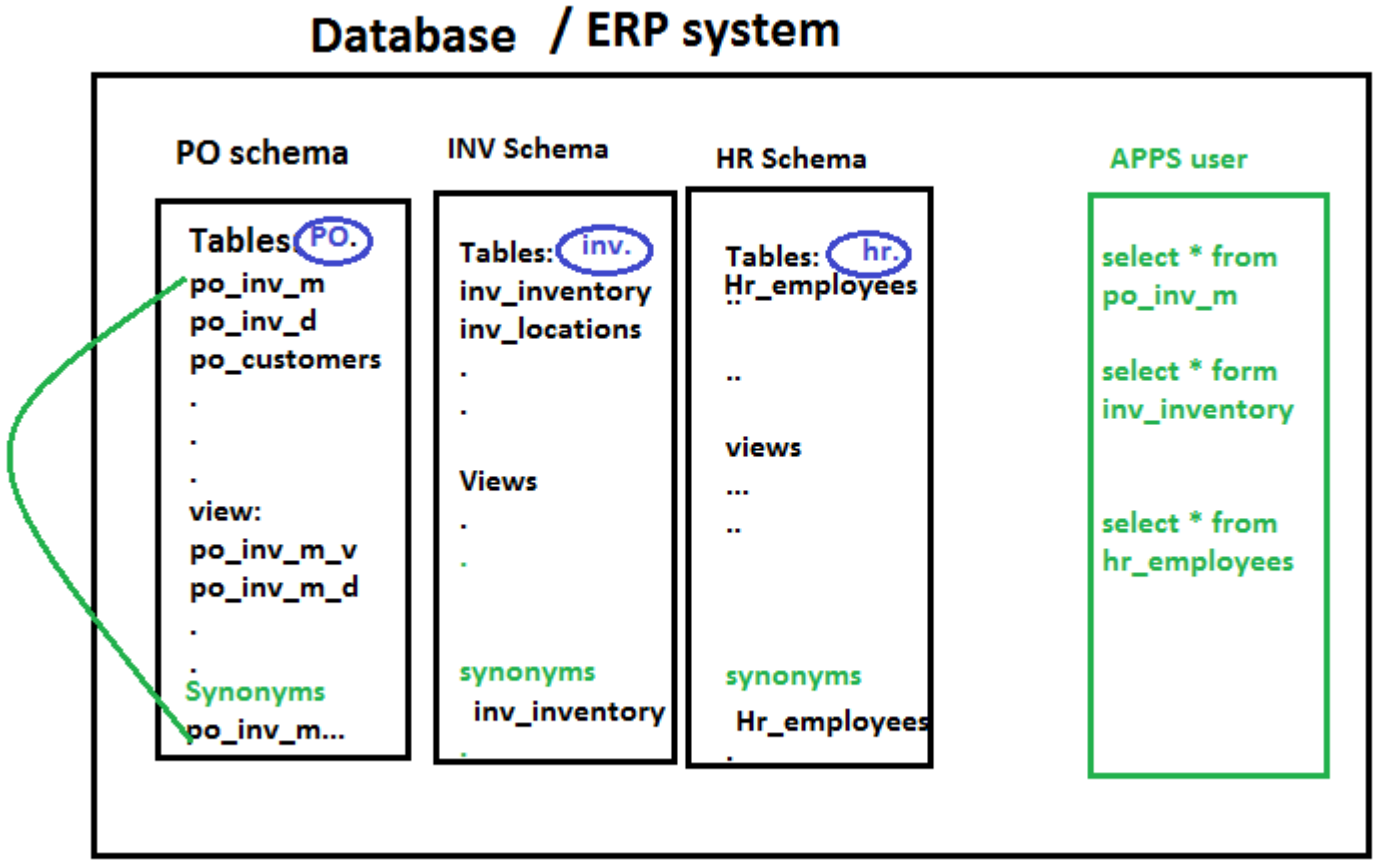
```
public synonym DEPT created.
```

Removing a Synonym

To remove a synonym, use the DROP SYNONYM statement. Only the database administrator can drop a public synonym.

```
DROP PUBLIC SYNONYM  dept;
```

Real Example for using the synonyms



Indexes

An index:

- Is a schema object
- Can be used by the Oracle Server to speed up the retrieval of rows by using a pointer
- Can reduce disk input/output (I/O) by using a rapid path access method to locate data quickly
- Is dependent on the table that it indexes
- Is used and maintained automatically by the Oracle Server

An Oracle Server index is a schema object that can speed up the retrieval of rows by using a pointer and improves the performance of some queries. Indexes can be created explicitly or automatically. If you do not have an index on the column, a full table scan occurs.

An index provides direct and fast access to rows in a table. Its purpose is to reduce the disk I/O by using an indexed path to locate data quickly. An index is used and maintained automatically by the Oracle Server. After an index is created, no direct activity is required by the user.

Indexes are logically and physically independent of the data in the objects with which they are associated. This means that they can be created or dropped at any time, and have no effect on the base tables or other indexes.

Note: When you drop a table, the corresponding indexes are also dropped.

How Are Indexes Created?

- Automatically: A unique index is created automatically when you define a PRIMARY KEY or UNIQUE constraint in a table definition.

  The name of index will be as the constraint name

- Manually: You can create unique or nonunique index on columns to speed up access to the rows.

  Here the user who give the name for the index

You can create two types of indexes.

- **Unique index:** The Oracle Server automatically creates this index when you define a column in a table to have a PRIMARY KEY or a UNIQUE constraint. The name of the index is the name that is given to the constraint.
- **Nonunique index:** This is an index that a user can create. For example, you can create the FOREIGN KEY column index for a join in a query to improve the speed of retrieval.

Note: You can manually create a unique index, but it is recommended that you create a unique constraint, which implicitly creates a unique index.

Creating an Index

- Create an index on one or more columns:

```
CREATE [UNIQUE] [BITMAP] INDEX index
ON table (column[, column]...);
```

- Improve the speed of query access to the LAST_NAME column in the EMPLOYEES table:

```
CREATE INDEX emp_last_name_idx
ON employees(last_name);
```

```
index EMP_LAST_NAME_IDX created.
```

Create an index on one or more columns by issuing the CREATE INDEX statement.

In the syntax:

- index — Is the name of the index
- table — Is the name of the table
- Column — Is the name of the column in the table to be indexed

Specify UNIQUE to indicate that the value of the column (or columns) upon which the index is based must be unique. Specify BITMAP to indicate that the index is to be created with a bitmap for each distinct key, rather than indexing each row separately. Bitmap indexes store the rowids associated with a key value as a bitmap.

CREATE INDEX with the CREATE TABLE Statement

```
CREATE TABLE NEW_EMP
(employee_id NUMBER(6)
             PRIMARY KEY USING INDEX
             (CREATE INDEX emp_id_idx ON
             NEW_EMP(employee_id)),
first_name   VARCHAR2(20),
last_name    VARCHAR2(25));
```

```
table NEW_EMP created.
```

```
SELECT INDEX_NAME, TABLE_NAME
FROM   USER_INDEXES
WHERE  TABLE_NAME = 'NEW_EMP';
```

	INDEX_NAME	TABLE_NAME
1	EMP_ID_IDX	NEW_EMP

Function-Based Indexes

- A function-based index is based on expressions.
- The index expression is built from table columns, constants, SQL functions, and user-defined functions.

```
CREATE INDEX upper_dept_name_idx
ON dept2(UPPER(department_name));
```

```
index UPPER_DEPT_NAME_IDX created.
```

```
SELECT *
FROM   dept2
WHERE  UPPER(department_name) = 'SALES';
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	80	Sales	145	2500

The Oracle Server uses the index only when that particular function is used in a query. For example, the following statement may use the index, but without the WHERE clause, the Oracle Server may perform a full table scan:

```
SELECT   *
FROM     employees
WHERE    UPPER (last_name) IS NOT NULL
ORDER BY UPPER (last_name);
```

Index Creation Guidelines

Create an index when:	
✓	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% of the rows in the table
Do not create an index when:	
✗	The columns are not often used as a condition in the query
✗	The table is small or most queries are expected to retrieve more than 2% to 4% of the rows in the table
✗	The table is updated frequently
✗	The indexed columns are referenced as part of an expression

Index Information

- USER_INDEXES provides information about your indexes.
- USER_IND_COLUMNS describes columns of indexes owned by you and columns of indexes on your tables.

Removing an Index

- Remove an index from the data dictionary by using the DROP INDEX command:

```
DROP INDEX index;
```

- Remove the emp_last_name_idx index from the data dictionary:

```
DROP INDEX emp_last_name_idx;
index EMP_LAST_NAME_IDX dropped.
```

- To drop an index, you must be the owner of the index or have the DROP ANY INDEX privilege.

You cannot modify indexes. To change an index, you must drop it and then re-create it.

Remove an index definition from the data dictionary by issuing the DROP INDEX statement. To drop an index, you must be the owner of the index or have the DROP ANY INDEX privilege.

In the syntax, *index* is the name of the index.

You can drop an index using the ONLINE keyword.

```
DROP INDEX emp_indx ONLINE;
```

ONLINE: Specify ONLINE to indicate that DML operations on the table are allowed while dropping the index.

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

Thank You