17

Manage Schema Objects

Objectives

After completing this lesson, you should be able to do the following:

- Add constraints
- Create indexes
- Create indexes using the `CREATE TABLE` statement
- Creating function-based indexes
- Drop columns and set column `UNUSED`
- Perform `FLASHBACK` operations

The `ALTER TABLE` Statement

Use the `ALTER TABLE` statement to:

- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column

The ALTER TABLE Statement

Use the ALTER TABLE statement to add, modify, or drop columns.

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
DROP        (column);
```

Adding a Column

- You use the `ADD` clause to add columns.

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
Table altered.
```

- The new column becomes the last column.

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
145	Russell	14000	01-OCT-96	
146	Partners	13500	05-JAN-97	
147	Errazuriz	12000	10-MAR-97	
148	Cambrault	11000	15-OCT-99	
149	Zlotkey	10500	29-JAN-00	

...

Modifying a Column

- You can change a column's data type, size, and default value.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
Table altered.
```

- A change to the default value affects only subsequent insertions to the table.

Dropping a Column

Use the `DROP COLUMN` clause to drop columns you no longer need from the table.

```
ALTER TABLE  dept80
DROP COLUMN  job_id;
Table altered.
```

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
145	Russell	14000	01-OCT-96
146	Partners	13500	05-JAN-97
147	Errazuriz	12000	10-MAR-97
148	Cambrault	11000	15-OCT-99
149	Zlotkey	10500	29-JAN-00

Adding a Constraint Syntax

Use the `ALTER TABLE` statement to:

- Add or drop a constraint, but not modify its structure
- Enable or disable constraints
- Add a `NOT NULL` constraint by using the `MODIFY` clause

```
ALTER TABLE  <table_name>
ADD [CONSTRAINT <constraint_name>]
type (<column_name>);
```

Adding a Constraint

Add a `FOREIGN KEY` constraint to the `EMP2` table indicating that a manager must already exist as a valid employee in the `EMP2` table.

```
ALTER TABLE emp2
modify employee_id Primary Key;
Table altered.
```

```
ALTER TABLE emp2
ADD CONSTRAINT emp_mgr_fk
  FOREIGN KEY(manager_id)
  REFERENCES emp2(employee_id);
Table altered.
```

ON DELETE CASCADE

Delete child rows when a parent key is deleted.

```
ALTER TABLE Emp2 ADD CONSTRAINT emp_dt_fk
FOREIGN KEY (Department_id)
REFERENCES departments ON DELETE CASCADE);
Table altered.
```

Dropping a Constraint

- Remove the manager constraint from the `EMP2` table.

```
ALTER TABLE emp2
DROP CONSTRAINT emp_mgr_fk;
Table altered.
```

- Remove the `PRIMARY KEY` constraint on the `DEPT2` table and drop the associated `FOREIGN KEY` constraint on the `EMP2.DEPARTMENT_ID` column.

```
ALTER TABLE dept2
DROP PRIMARY KEY CASCADE;
Table altered.
```

Disabling Constraints

- Execute the `DISABLE` clause of the `ALTER TABLE` statement to deactivate an integrity constraint.
- Apply the `CASCADE` option to disable dependent integrity constraints.

```
ALTER TABLE emp2
DISABLE CONSTRAINT emp_dt_fk;
Table altered.
```

Enabling Constraints

- Activate an integrity constraint currently disabled in the table definition by using the `ENABLE` clause.

```
ALTER TABLE       emp2
ENABLE CONSTRAINT emp_dt_fk;
Table altered.
```

- A `UNIQUE` index is automatically created if you enable a `UNIQUE` key or `PRIMARY KEY` constraint.

Cascading Constraints

- The `CASCADE CONSTRAINTS` clause is used along with the `DROP COLUMN` clause.
- The `CASCADE CONSTRAINTS` clause drops all referential integrity constraints that refer to the primary and unique keys defined on the dropped columns.
- The `CASCADE CONSTRAINTS` clause also drops all multicolumn constraints defined on the dropped columns.

Cascading Constraints

Example:

```
ALTER TABLE emp2
DROP COLUMN employee_id CASCADE CONSTRAINTS;
Table altered.
```

```
ALTER TABLE test1
DROP (pk, fk, col1) CASCADE CONSTRAINTS;
Table altered.
```

Overview of Indexes

Indexes are created:

- **Automatically**
  - **`PRIMARY KEY` creation**
  - **`UNIQUE KEY` creation**
- **Manually**
  - **`CREATE INDEX` statement**
  - **`CREATE TABLE` statement**

CREATE INDEX with 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 created.
```

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

INDEX_NAME	TABLE_NAME
EMP_ID_IDX	NEW_EMP

DROP TABLE … PURGE

```
DROP TABLE dept80 PURGE;
```

The FLASHBACK TABLE Statement

- Repair tool for accidental table modifications
  - Restores a table to an earlier point in time
  - Benefits: Ease of use, availability, fast execution
  - Performed in place
- Syntax:

```
FLASHBACK TABLE[schema.]table[,
[ schema.]table ]...
TO { TIMESTAMP | SCN } expr
[ { ENABLE | DISABLE } TRIGGERS ];
```

The FLASHBACK TABLE Statement

```
DROP TABLE emp2;
Table dropped
```

```
 SELECT original_name, operation, droptime,
FROM recyclebin;
```

ORIGINAL_NAME	OPERATION	DROPTIME
EMP2	DROP	2004-03-03:07:57:11

...

```
FLASHBACK TABLE emp2 TO BEFORE DROP;
Flashback complete
```

Summary

In this lesson, you should have learned how to:

- Add constraints
- Create indexes
- Create a primary key constraint using an index
- Create indexes using the `CREATE TABLE` statement
- Creating function-based indexes
- Drop columns and set column `UNUSED`
- Perform `FLASHBACK` operations