2

Managing Schema Objects

ORACLE

Objectives

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

- Add constraints
- Create indexes
- Create indexes using the `CREATE TABLE` statement
- Create function-based indexes
- Drop columns and set columns as `UNUSED`
- Perform `FLASHBACK` operations
- Create and use external tables

Lesson Agenda

- Using the `ALTER TABLE` statement to add, modify, and drop a column
- Managing constraints
  - Adding and dropping a constraint
  - Deferring constraints
  - Enabling and disabling a constraint
- Creating indexes
  - Using the `CREATE TABLE` statement
  - Creating function-based indexes
  - Removing an index
- Performing flashback operations
- Creating and using external tables

`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

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);
```

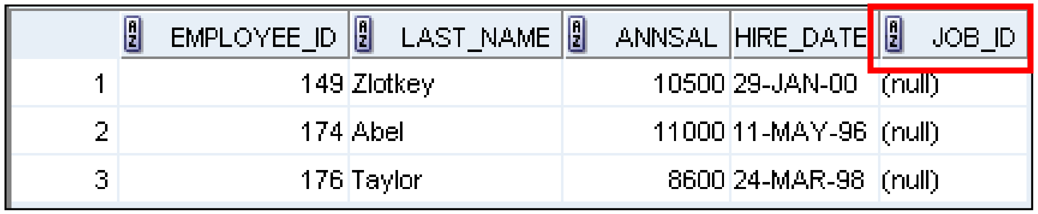
Adding a Column

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

```
ALTER TABLE dept80
ADD          (job_id VARCHAR2(9));
```

```
ALTER TABLE dept80 succeeded.
```

- The new column becomes the last column:



	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	149	Zlotkey	10500	29-JAN-00	(null)
2	174	Abel	11000	11-MAY-96	(null)
3	176	Taylor	8600	24-MAR-98	(null)

Modifying a Column

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

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
```

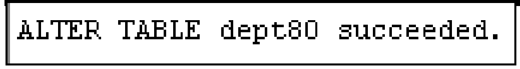
```
ALTER TABLE dept80 succeeded.
```

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

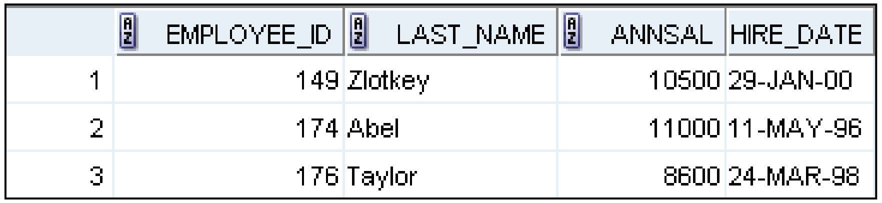
Dropping a Column

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

```
ALTER TABLE  dept80
DROP COLUMN  job_id;
```



```
ALTER TABLE dept80 succeeded.
```



	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	10500	29-JAN-00
2	174	Abel	11000	11-MAY-96
3	176	Taylor	8600	24-MAR-98

SET UNUSED Option

- You use the SET UNUSED option to mark one or more columns as unused.
- You use the DROP UNUSED COLUMNS option to remove the columns that are marked as unused.

```
ALTER TABLE  <table_name>
SET    UNUSED(<column name>);
OR
ALTER TABLE  <table_name>
SET    UNUSED COLUMN <column_name>;
```

```
ALTER TABLE <table_name>
DROP   UNUSED COLUMNS;
```

Lesson Agenda

- Using the `ALTER TABLE` statement to add, modify, and drop a column
- Managing constraints
  - Adding and dropping a constraint
  - Deferring constraints
  - Enabling and disabling a constraint
- Creating indexes
  - Using the `CREATE TABLE` statement
  - Creating function-based indexes
  - Removing an index
- Performing flashback operations
- Creating and using external tables

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;
```

```
ALTER TABLE emp2 succeeded.
```

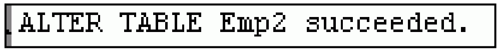
```
ALTER TABLE emp2
ADD CONSTRAINT emp_mgr_fk
  FOREIGN KEY(manager_id)
  REFERENCES emp2(employee_id);
```

```
ALTER TABLE  succeeded.
```

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(department_id) ON DELETE CASCADE;
```



```
ALTER TABLE Emp2 succeeded.
```

Deferring Constraints

Constraints can have the following attributes:

- `DEFERRABLE` or `NOT DEFERRABLE`
- `INITIALLY DEFERRED` or `INITIALLY IMMEDIATE`

```
ALTER TABLE dept2
ADD CONSTRAINT dept2_id_pk
PRIMARY KEY (department_id)
DEFERRABLE INITIALLY DEFERRED
```

Deferring constraint on creation

```
SET CONSTRAINTS dept2_id_pk IMMEDIATE
```

Changing a specific constraint attribute

```
ALTER SESSION
SET CONSTRAINTS= IMMEDIATE
```

Changing all constraints for a session

Difference Between `INITIALLY DEFERRED` and `INITIALLY IMMEDIATE`

`INITIALLY DEFERRED`	Waits to check the constraint until the transaction ends
`INITIALLY IMMEDIATE`	Checks the constraint at the end of the statement execution

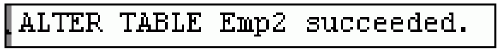
```
CREATE TABLE emp_new_sal (salary NUMBER
        CONSTRAINT sal_ck
        CHECK (salary > 100)
        DEFERRABLE INITIALLY IMMEDIATE,
        bonus NUMBER
        CONSTRAINT bonus_ck
        CHECK (bonus > 0 )
        DEFERRABLE INITIALLY DEFERRED );
```

```
create table succeeded.
```

Dropping a Constraint

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

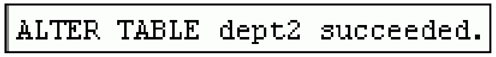
```
ALTER TABLE emp2
DROP CONSTRAINT emp_mgr_fk;
```



```
ALTER TABLE Emp2 succeeded.
```

- 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;
```

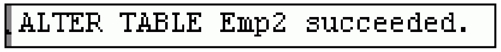


```
ALTER TABLE dept2 succeeded.
```

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;
```

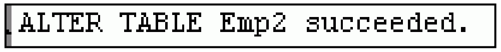


```
ALTER TABLE Emp2 succeeded.
```

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;
```



```
ALTER TABLE Emp2 succeeded.
```

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

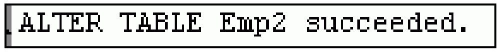
Cascading Constraints

- 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

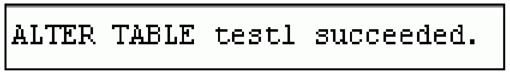
Example:

```
ALTER TABLE emp2
DROP COLUMN employee_id CASCADE CONSTRAINTS;
```



```
ALTER TABLE Emp2 succeeded.
```

```
ALTER TABLE test1
DROP (col1_pk, col2_fk, col1) CASCADE CONSTRAINTS;
```



```
ALTER TABLE test1 succeeded.
```

Renaming Table Columns and Constraints

Use the `RENAME COLUMN` clause of the `ALTER TABLE` statement to rename table columns.

a

```
ALTER TABLE marketing RENAME COLUMN team_id
TO id;
```

```
ALTER TABLE marketing succeeded.
```

Use the `RENAME CONSTRAINT` clause of the `ALTER TABLE` statement to rename any existing constraint for a table.

b

```
ALTER TABLE marketing RENAME CONSTRAINT mktg_pk
TO new_mktg_pk;
```

```
ALTER TABLE marketing succeeded.
```

Lesson Agenda

- Using the `ALTER TABLE` statement to add, modify, and drop a column
- Managing constraints
  - Adding and dropping a constraint
  - Deferring constraints
  - Enabling and disabling a constraint
- Creating indexes
  - Using the `CREATE TABLE` statement
  - Creating function-based indexes
  - Removing an index
- Performing flashback operations
- Creating and using external tables

Overview of Indexes

Indexes are created:

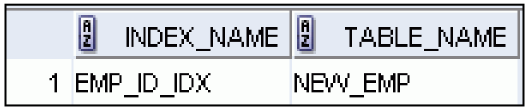
- Automatically
  - `PRIMARY KEY` creation
  - `UNIQUE KEY` creation
- Manually
  - The `CREATE INDEX` statement
  - The `CREATE TABLE` statement

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));
```

```
CREATE TABLE succeeded.
```

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

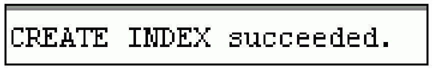


	INDEX_NAME	TABLE_NAME
1	EMP_ID_IDX	NEW_EMP

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));
```



```
CREATE INDEX succeeded.
```

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

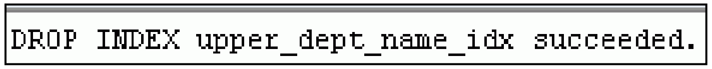
Removing an Index

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

```
DROP INDEX index;
```

- Remove the `UPPER_DEPT_NAME_IDX` index from the data dictionary:

```
DROP INDEX upper_dept_name_idx;
```

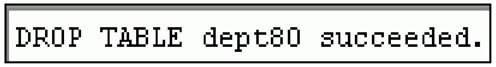


```
DROP INDEX upper_dept_name_idx succeeded.
```

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

DROP TABLE … PURGE

```
DROP TABLE dept80 PURGE;
```



```
DROP TABLE dept80 succeeded.
```

Lesson Agenda

- Using the `ALTER TABLE` statement to add, modify, and drop a column
- Managing constraints
  - Adding and dropping a constraint
  - Deferring constraints
  - Enabling and disabling a constraint
- Creating indexes
  - Using the `CREATE TABLE` statement
  - Creating function-based indexes
  - Removing an index
- Performing flashback operations
- Creating and using external tables

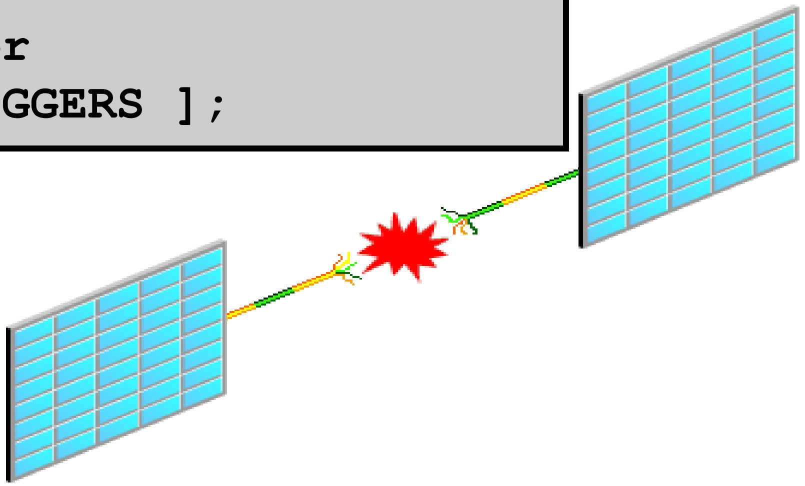
FLASHBACK TABLE Statement

- Enables you to recover tables to a specified point in time with a single statement
- Restores table data along with associated indexes, and constraints
- Enables you to revert the table and its contents to a certain point in time or SCN

FLASHBACK TABLE Statement

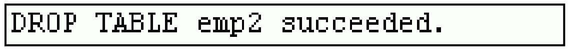
- Repair tool for accidental table modifications
  - Restores a table to an earlier point in time
  - Benefits: Ease of use, availability, and fast execution
  - Is performed in place
- Syntax:

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



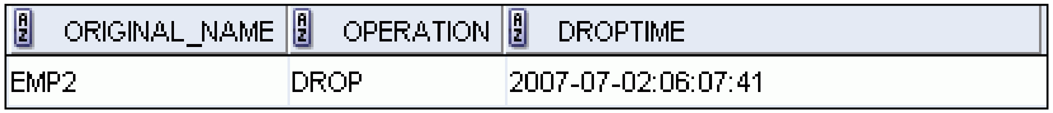
Using the `FLASHBACK TABLE` Statement

```
DROP TABLE emp2;
```



```
DROP TABLE emp2 succeeded.
```

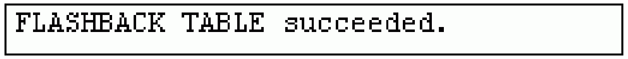
```
SELECT original_name, operation, droptime FROM
recyclebin;
```



ORIGINAL_NAME	OPERATION	DROPTIME
EMP2	DROP	2007-07-02:06:07:41

...

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

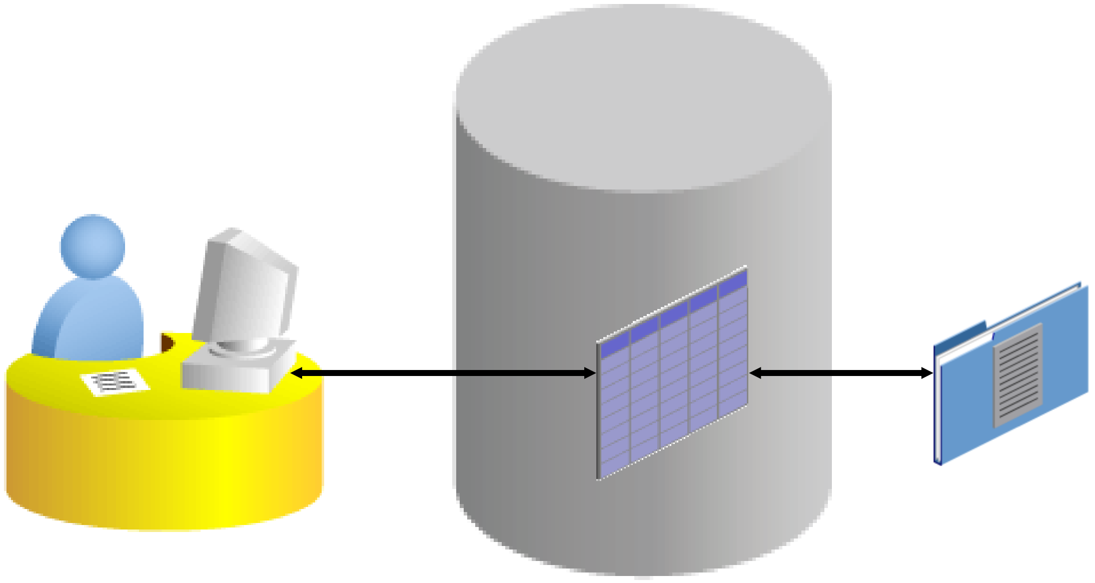


```
FLASHBACK TABLE succeeded.
```

Lesson Agenda

- Using the `ALTER TABLE` statement to add, modify, and drop a column
- Managing constraints
  - Adding and dropping a constraint
  - Deferring constraints
  - Enabling and disabling a constraint
- Creating indexes
  - Using the `CREATE TABLE` statement
  - Creating function-based indexes
  - Removing an index
- Performing flashback operations
- Creating and using external tables

External Tables



Creating a Directory for the External Table

Create a `DIRECTORY` object that corresponds to the directory on the file system where the external data source resides.

```
CREATE OR REPLACE DIRECTORY emp_dir
AS '/…/emp_dir';


GRANT READ ON DIRECTORY emp_dir TO hr;
```

Creating an External Table

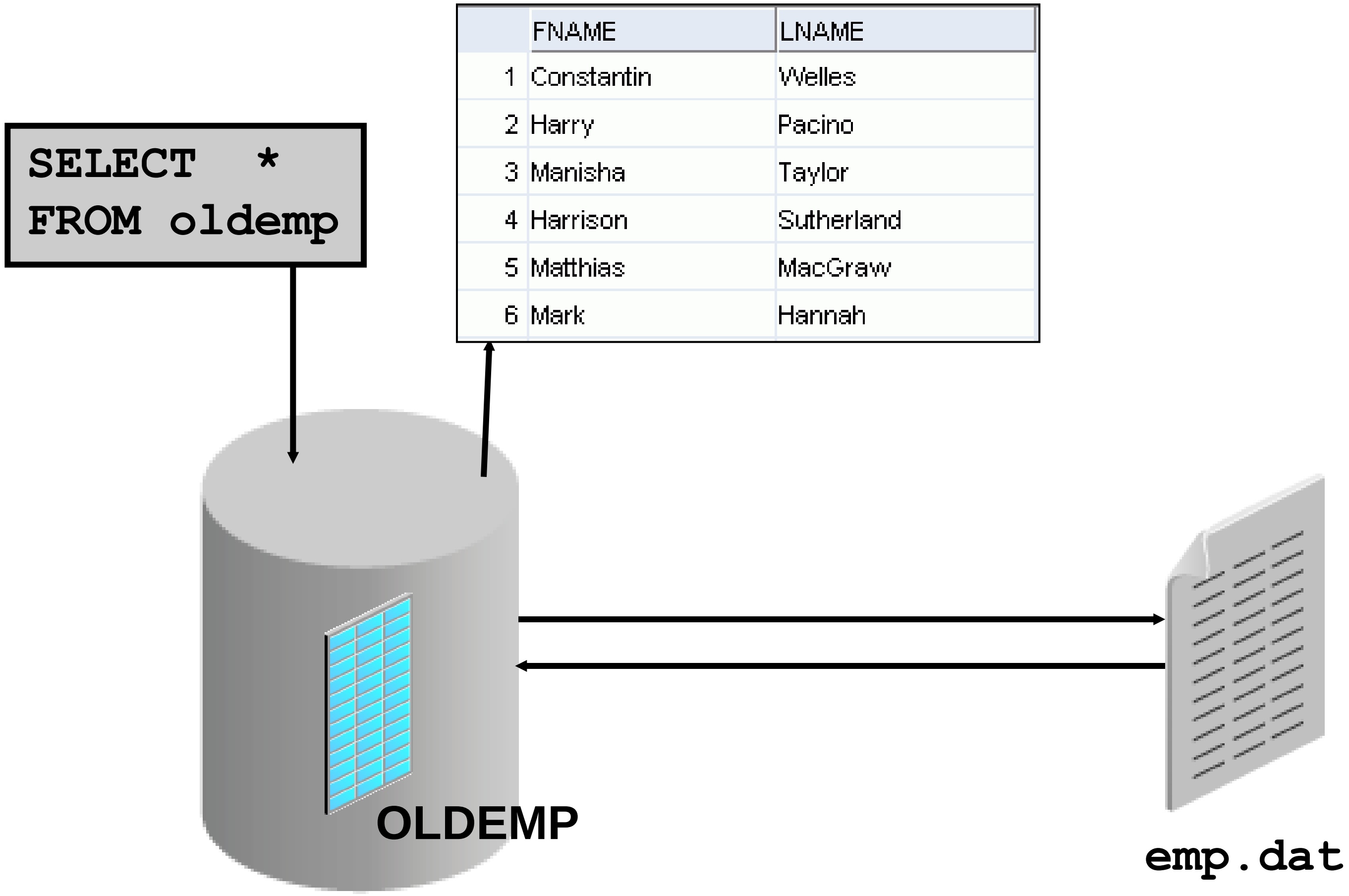
```
CREATE TABLE <table_name>
   ( <col_name> <datatype>, … )
ORGANIZATION EXTERNAL
    (TYPE <access_driver_type>
     DEFAULT DIRECTORY <directory_name>
     ACCESS PARAMETERS
       (… ) )
      LOCATION ('<location_specifier>') )
REJECT LIMIT [0 | <number> | UNLIMITED];
```

Creating an External Table by Using `ORACLE_LOADER`

```
CREATE TABLE oldemp (
  fname char(25), lname CHAR(25))
  ORGANIZATION EXTERNAL
  (TYPE ORACLE_LOADER
  DEFAULT DIRECTORY emp_dir
  ACCESS PARAMETERS
  (RECORDS DELIMITED BY NEWLINE
   NOBADFILE
   NOLOGFILE
  FIELDS TERMINATED BY ','
  (fname POSITION ( 1:20) CHAR,
   lname POSITION (22:41) CHAR))
  LOCATION ('emp.dat'))
  PARALLEL 5
  REJECT LIMIT 200;
```

```
CREATE TABLE succeeded.
```

Querying External Tables



```
SELECT  *
FROM oldemp
```

	FNAME	LNAME
1	Constantin	Welles
2	Harry	Pacino
3	Manisha	Taylor
4	Harrison	Sutherland
5	Matthias	MacGraw
6	Mark	Hannah

Creating an External Table by Using `ORACLE_DATAPUMP`: Example

```
CREATE TABLE emp_ext
  (employee_id, first_name, last_name)
   ORGANIZATION EXTERNAL
    (
     TYPE ORACLE_DATAPUMP
     DEFAULT DIRECTORY emp_dir
     LOCATION
      ('emp1.exp','emp2.exp')
    )
   PARALLEL
AS
SELECT employee_id, first_name, last_name
FROM   employees;
```

Summary

In this lesson, you should have learned how to:

- Add constraints
- Create indexes
- Create indexes using the `CREATE TABLE` statement
- Create function-based indexes
- Drop columns and set columns as `UNUSED`
- Perform `FLASHBACK` operations
- Create and use external tables

Practice 2: Overview

This practice covers the following topics:

- Altering tables
- Adding columns
- Dropping columns
- Creating indexes
- Creating external tables