ORACLE SQL

Lesson 01: Privileges, Multitable Inserts, External Tables

Lesson Objectives

- To understand the following topics:
  - Differentiate system privileges from object privileges
  - Grant privileges on tables
  - Grant roles
  - Distinguish between privileges and roles
  - Manipulating data by using subqueries
  - Specifying explicit default values in the INSERT and UPDATE statements
  - Using the following types of multitable INSERTs:
  - Unconditional INSERT
  - Pivoting INSERT
  - Conditional INSERT ALL
  - Conditional INSERT FIRST
  - Merging rows in a table
  - Tracking the changes to data over a period of time

Privileges - Introduction

- Database security:
  - System security
  - Data security
  - System privileges: Performing a particular action within the database
  - Object privileges: Manipulating the content of the database objects
  - Schemas: Collection of objects such as tables, views, and sequences

1.1: Privileges
System Privileges

- More than 100 privileges are available.
- The database administrator has high-level system privileges for tasks such as:
- Creating new users
- Removing users
- Removing tables
- Backing up tables

1.1: Privileges
Creating Users

The DBA creates users with the CREATE USER statement.

```
CREATE USER user
IDENTIFIED BY   password;
```

```
CREATE USER  demo
IDENTIFIED BY demo;
```

1.1: Privileges
User System Privileges

After a user is created, the DBA can grant specific system privileges to that user.

An application developer, for example, may have the following system privileges:
CREATE SESSION
CREATE TABLE
CREATE SEQUENCE
CREATE VIEW
CREATE PROCEDURE

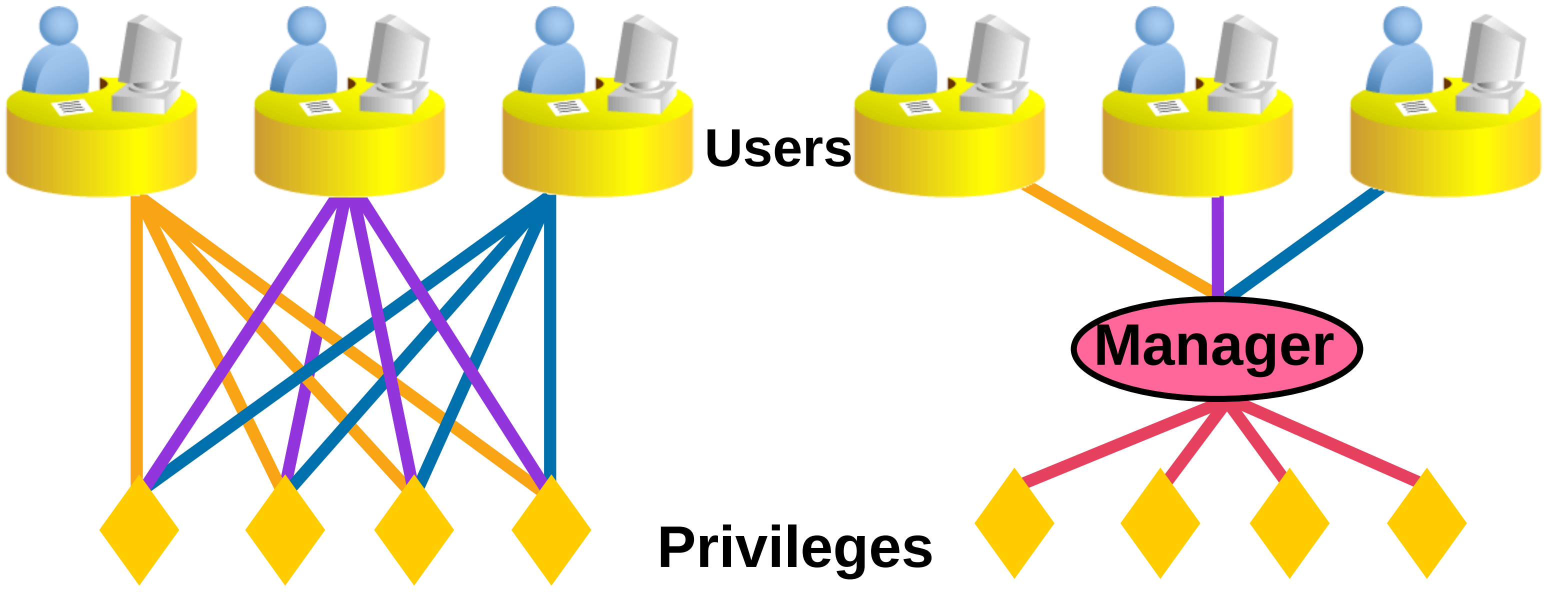
```
GRANT privilege [, privilege...]
TO user [, user| role, PUBLIC...];
```

1.1: Privileges
Granting System Privileges

The DBA can grant specific system privileges to a user.

```
GRANT  create session, create table,
       create sequence, create view
TO     demo;
```

1.1: Privileges
What Is a Role?



Allocating privileges without a role

Allocating privileges with a role

1.1: Privileges
Creating and Granting Privileges to a Role

Create a role:

```
CREATE ROLE manager;
```

Grant privileges to a role:

```
GRANT create table, create view
TO manager;
```

Grant a role to users:

```
GRANT manager TO alice;
```

1.1: Privileges
Changing Your Password

The DBA creates your user account and initializes your password.
You can change your password by using the ALTER USER statement.

```
ALTER USER demo
IDENTIFIED BY employ;
```

1.1: Privileges
Object Privileges

Object privilege	Table	View	Sequence
ALTER	✓		✓
DELETE	✓	✓	
INDEX	✓		
INSERT	✓	✓	
REFERENCE	✓		
SELECT	✓	✓	✓
UPDATE	✓	✓	

1.1: Privileges
Object Privileges

Object privileges vary from object to object.
An owner has all the privileges on the object.
An owner can give specific privileges on that owner's object.

```
GRANT                object_priv [(columns)]
ON                   object
TO                   {user|role|PUBLIC}
[WITH GRANT OPTION];
```

1.1: Privileges
Granting Object Privileges

Grant query privileges on the EMPLOYEES table:

```
GRANT  select
ON     employees
TO     demo;
```

Grant privileges to update specific columns to users and roles:

```
GRANT  update (department_name, location_id)
ON     departments
TO     demo, manager;
```

1.1: Privileges
Passing On Your Privileges

Give a user authority to pass along privileges:

```
GRANT   select, insert
ON      departments
TO      demo
WITH    GRANT OPTION;
```

Allow all users on the system to query data from Alice’s DEPARTMENTS table:

```
GRANT   select
ON      alice.departments
TO      PUBLIC;
```

Confirming Granted Privileges

Data Dictionary View	Description
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Table privileges granted to roles
USER_ROLE_PRIVS	Roles accessible by the user
USER_SYS_PRIVS	System privileges granted to the user
USER_TAB_PRIVS_MADE	Object privileges granted on the user’s objects
USER_TAB_PRIVS_RECD	Object privileges granted to the user
USER_COL_PRIVS_MADE	Object privileges granted on the columns of the user’s objects
USER_COL_PRIVS_RECD	Object privileges granted to the user on specific columns

1.1: Privileges
Revoking Object Privileges

You use the REVOKE statement to revoke privileges granted to other users.
Privileges granted to others through the WITH GRANT OPTION clause are also revoked.

```
REVOKE {privilege [, privilege...]|ALL}
ON      object
FROM    {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

1.1: Privileges
Revoking Object Privileges

Revoke the SELECT and INSERT privileges given to the demo user on the DEPARTMENTS table.

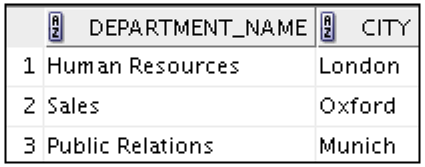
```
REVOKE  select, insert
ON      departments
FROM    demo;
```

1.1: Privileges
Using Subqueries to Manipulate Data

You can use subqueries in data manipulation language (DML) statements to:
Retrieve data by using an inline view
Copy data from one table to another
Update data in one table based on the values of another table
Delete rows from one table based on rows in another table

1.1: Privileges
Retrieving Data by Using a Subquery as Source

```
SELECT department_name, city
FROM   departments
NATURAL JOIN (SELECT l.location_id, l.city, l.country_id
              FROM   loc l
              JOIN   countries c
              ON(l.country_id = c.country_id)
              JOIN regions USING(region_id)
              WHERE region_name = 'Europe');
```



	DEPARTMENT_NAME	CITY
1	Human Resources	London
2	Sales	Oxford
3	Public Relations	Munich

1.1: Privileges
Inserting by Using a Subquery as a Target

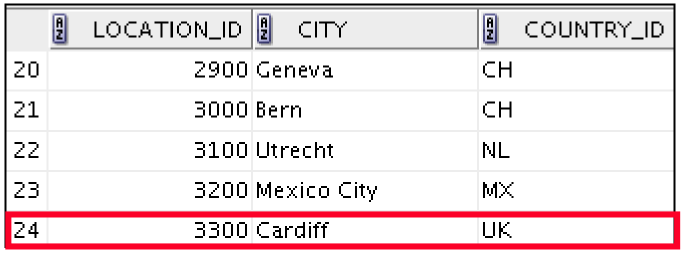
```
INSERT INTO (SELECT l.location_id, l.city, l.country_id
            FROM   locations l
            JOIN   countries c
            ON(l.country_id = c.country_id)
            JOIN regions USING(region_id)
            WHERE region_name = 'Europe')
VALUES (3300, 'Cardiff', 'UK');
```

```
1 rows inserted
```

1.1: Privileges
Inserting by Using a Subquery as a Target

Verify the results.

```
SELECT location_id, city, country_id
FROM   loc
```

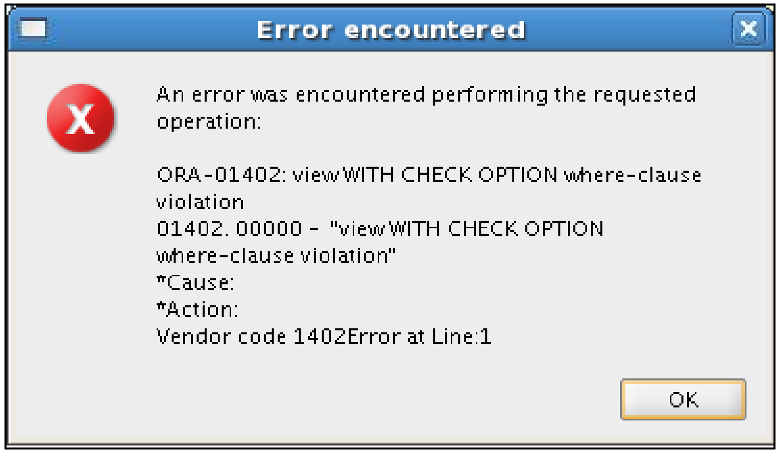


	LOCATION_ID	CITY	COUNTRY_ID
20	2900	Geneva	CH
21	3000	Bern	CH
22	3100	Utrecht	NL
23	3200	Mexico City	MX
24	3300	Cardiff	UK

1.1: Privileges

Using the WITH CHECK OPTION Keyword on DML Statements

The WITH CHECK OPTION keyword prohibits you from changing rows that are not in the subquery.



```
An error was encountered performing the requested
operation:

ORA-01402: view WITH CHECK OPTION where-clause
violation
01402. 00000 -  "view WITH CHECK OPTION
where-clause violation"
*Cause:
*Action:
Vendor code 1402Error at Line:1
```

1.1: Privileges
Overview of the Explicit Default Feature

Use the DEFAULT keyword as a column value where the default column value is desired.
This allows the user to control where and when the default value should be applied to data.
Explicit defaults can be used in INSERT and UPDATE statements.

1.1: Privileges
Using Explicit Default Values

DEFAULT with INSERT:

```
INSERT INTO deptm3
  (department_id, department_name, manager_id)
VALUES (300, 'Engineering', DEFAULT);
```

DEFAULT with UPDATE:

```
UPDATE deptm3
SET manager_id = DEFAULT
   WHERE department_id = 10;
```

1.1: Privileges
Copying Rows from Another Table

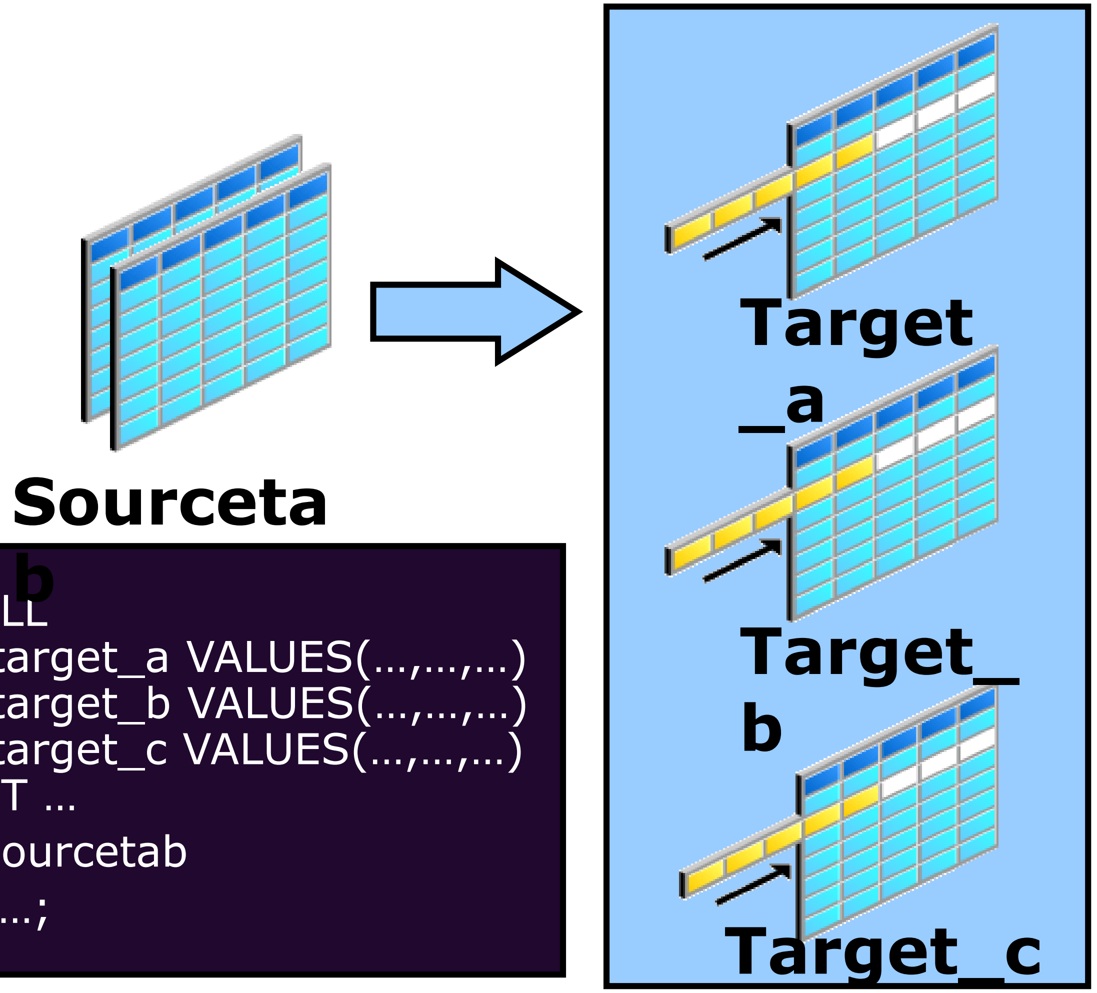
Write your INSERT statement with a subquery.

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';
```

```
33 rows inserted
```

Do not use the VALUES clause.
Match the number of columns in the INSERT clause with that in the subquery.

1.2: Multi table Inserts
Overview of Multi table INSERT Statements



```
INSERT  ALL
      INTO target_a VALUES(…,…,…)
      INTO target_b VALUES(…,…,…)
      INTO target_c VALUES(…,…,…)
      SELECT …
   FROM  sourcetab
   WHERE …;
```

1.2: Multi table Inserts
Overview of Multi table INSERT Statements

Use the INSERT…SELECT statement to insert rows into multiple tables as part of a single DML statement.

Multi table INSERT statements are used in data warehousing systems to transfer data from one or more operational sources to a set of target tables.

They provide significant performance improvement over:

Single DML versus multiple INSERT…SELECT statements

Single DML versus a procedure to perform multiple inserts by using the IF...THEN syntax

1.2: Multitable Inserts
Types of Multitable INSERT Statements

The different types of multitable INSERT statements are:

Unconditional INSERT

Conditional INSERT ALL

Pivoting INSERT

Conditional INSERT FIRST

1.2: Multitable Inserts
Multitable INSERT Statements

Syntax for multitable INSERT:

```
INSERT [conditional_insert_clause]
[insert_into_clause values_clause] (subquery)
```

Conditional_insert_clause:

```
[ALL|FIRST]
[WHEN condition THEN] [insert_into_clause values_clause]
[ELSE] [insert_into_clause values_clause]
```

1.2: Multitable Inserts
Unconditional INSERT ALL

Select the EMPLOYEE_ID, HIRE_DATE, SALARY, and MANAGER_ID values from the EMPLOYEES table for those employees whose EMPLOYEE_ID is greater than 200.

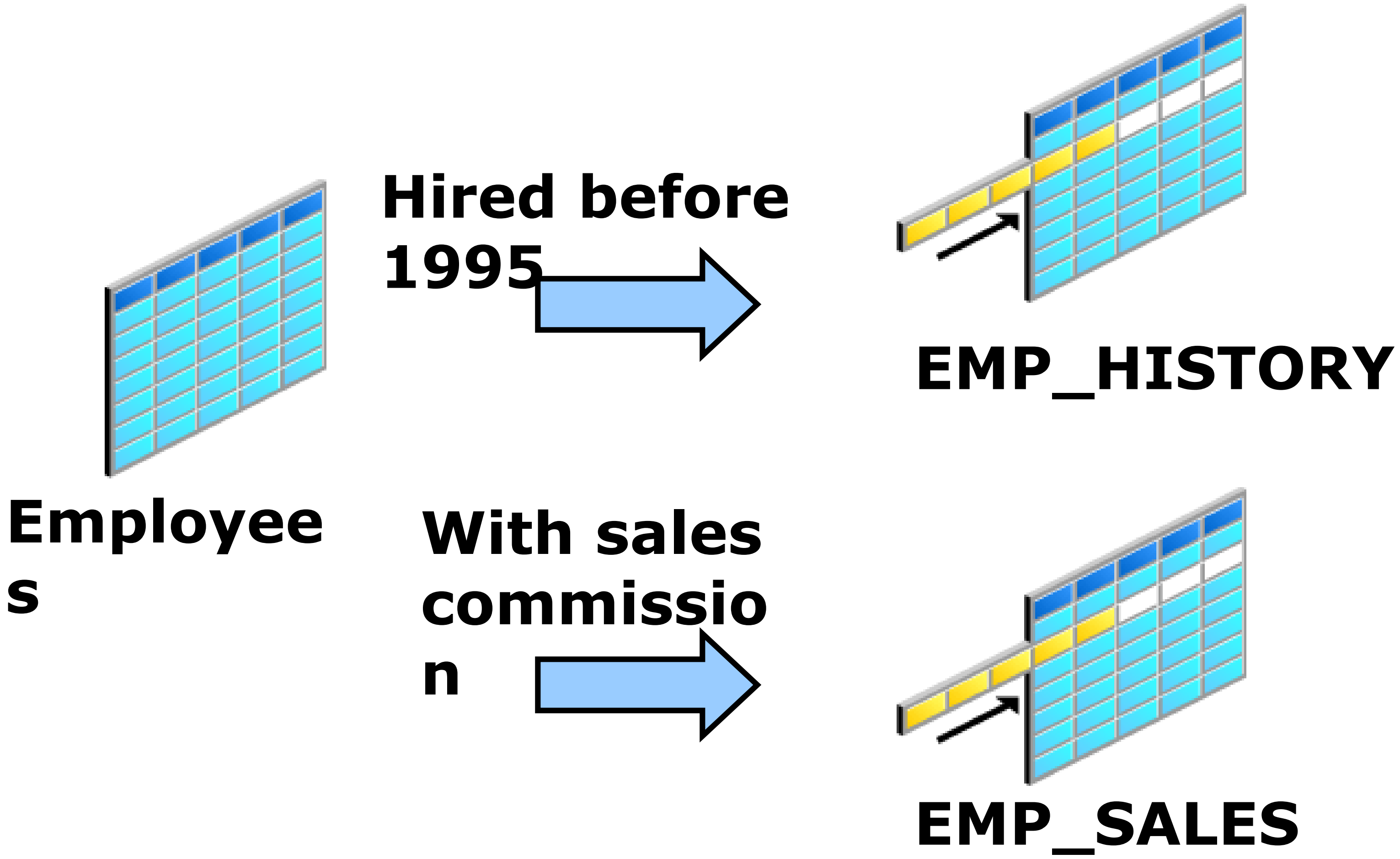
Insert these values into the SAL_HISTORY and MGR_HISTORY tables by using a multitable INSERT.

```
INSERT  ALL
    INTO sal_history VALUES(EMPID,HIREDATE,SAL)
    INTO mgr_history VALUES(EMPID,MGR,SAL)
  SELECT employee_id EMPID, hire_date HIREDATE,
         salary SAL, manager_id MGR
  FROM  employees
    WHERE employee_id > 200;
```

```
12 rows inserted
```

1.2: Multitable Inserts

Conditional INSERT ALL:Example



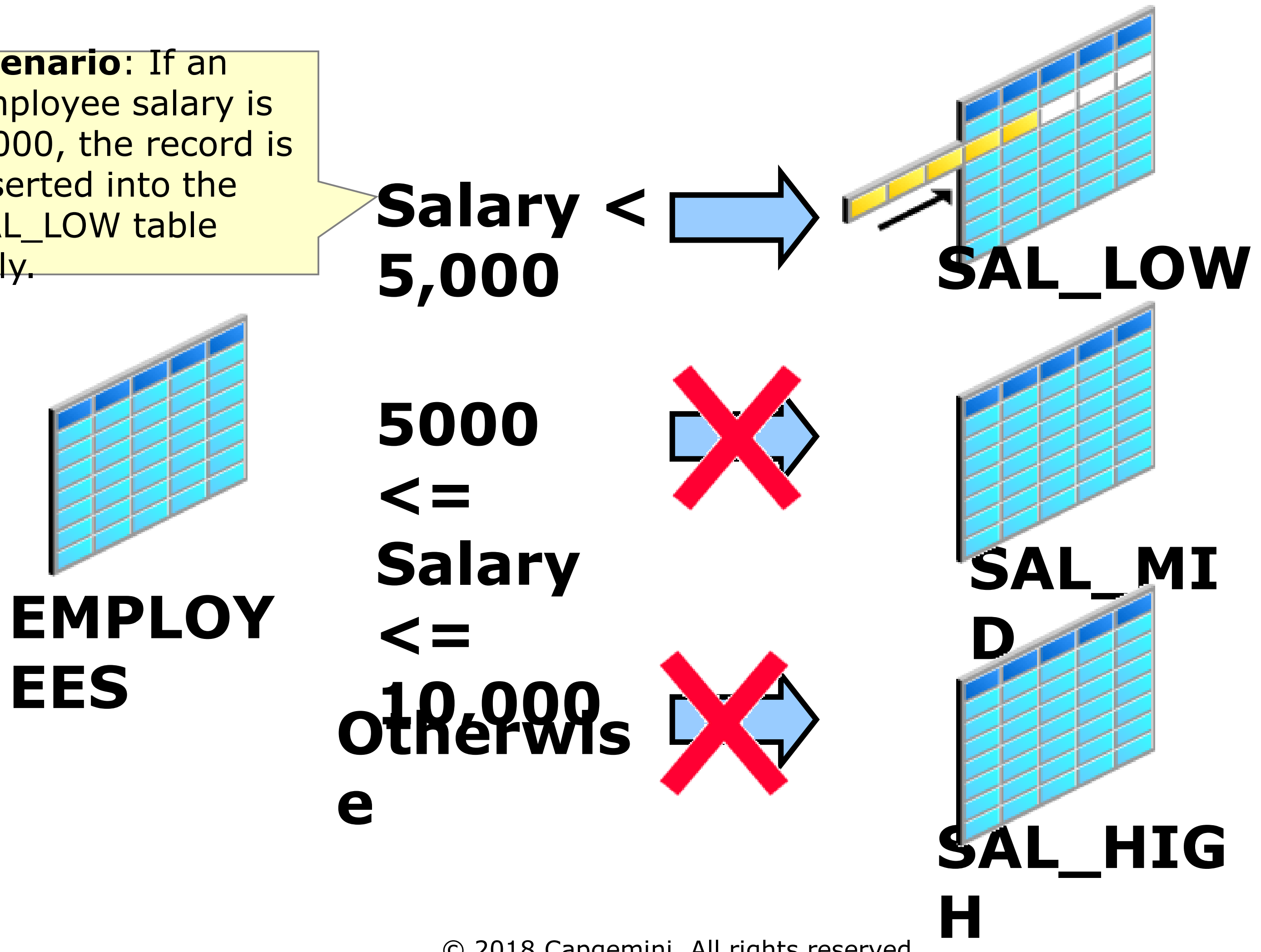
1.2: Multitable Inserts
Conditional `INSERT ALL`

```
INSERT  ALL
 WHEN HIREDATE < '01-JAN-95' THEN
   INTO emp_history VALUES(EMPID,HIREDATE,SAL)
 WHEN COMM IS NOT NULL THEN
   INTO emp_sales VALUES(EMPID,COMM,SAL)
   SELECT employee_id EMPID, hire_date HIREDATE,
          salary SAL, commission_pct COMM
   FROM  employees
```

```
48 rows inserted
```

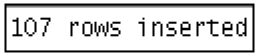
Conditional INSERT FIRST: Example

Scenario: If an employee salary is 2,000, the record is inserted into the SAL_LOW table only.



1.2: Multitable Inserts
Conditional INSERT FIRST

```
INSERT FIRST
WHEN salary < 5000 THEN
  INTO sal_low VALUES (employee_id, last_name, salary)
WHEN salary between 5000 and 10000 THEN
  INTO sal_mid VALUES (employee_id, last_name, salary)
ELSE
  INTO sal_high VALUES (employee_id, last_name, salary)
SELECT employee_id, last_name, salary
FROM employees
```



```
107 rows inserted
```

1.2: Multitable Inserts
Pivoting INSERT

Convert the set of sales records from the nonrelational database table to relational format.

Emp_ID	Week_ID	MON	TUES	WED	THUR	FRI
176	6	2000	3000	4000	5000	6000

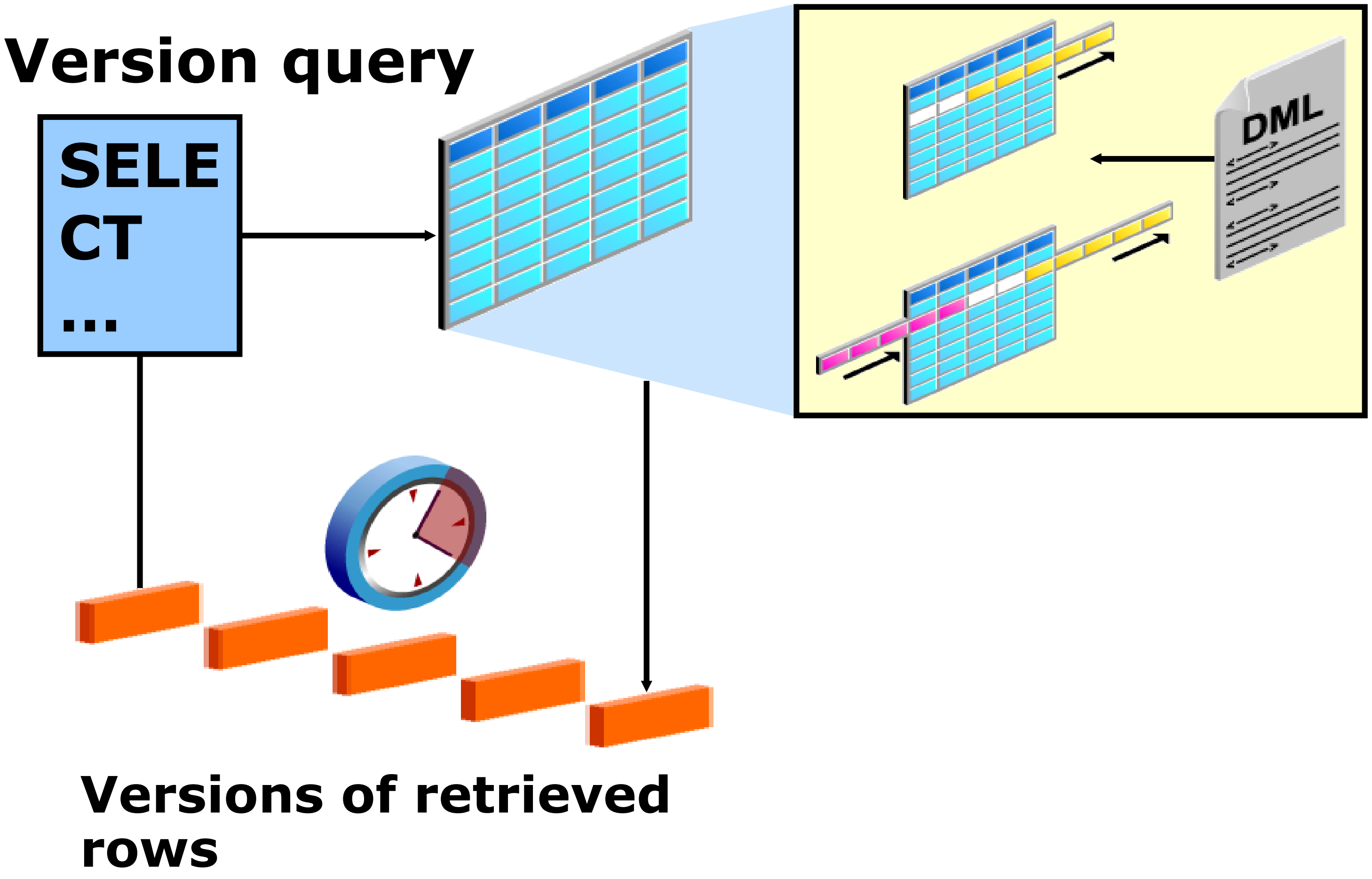
Employee_ID	WEEK	SALES
176	6	2000
176	6	3000
176	6	4000
176	6	5000
176	6	6000

1.2: Multitable Inserts
Pivoting INSERT

```
INSERT ALL
  INTO sales_info VALUES
(employee_id,week_id,sales_MON)
  INTO sales_info VALUES
(employee_id,week_id,sales_TUE)
  INTO sales_info VALUES
(employee_id,week_id,sales_WED)
  INTO sales_info VALUES
(employee_id,week_id,sales_THUR)
  INTO sales_info VALUES (employee_id,week_id,
sales_FRI)
  SELECT EMPLOYEE_ID, week_id, sales_MON, sales_TUE,
         sales_WED, sales_THUR,sales_FRI
  FROM sales_source_data;
```

```
5 rows inserted
```

1.2: Multitable Inserts
Tracking Changes in Data



1.2: Multitable Inserts
Example of the Flashback Version Query

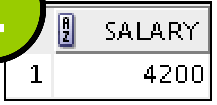
1
```
SELECT salary FROM employees3
WHERE  employee_id = 107;
```

2
```
UPDATE employees3 SET salary = salary * 1.30
WHERE  employee_id = 107;

COMMIT;
```

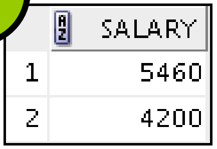
3
```
SELECT salary FROM employees3
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 107;
```

1



	SALARY
1	4200

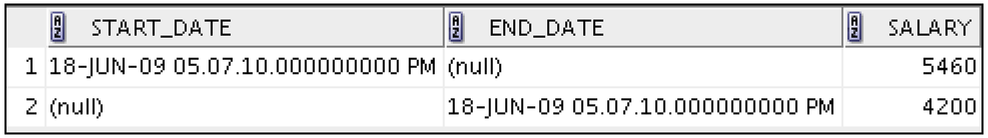
3



	SALARY
1	5460
2	4200

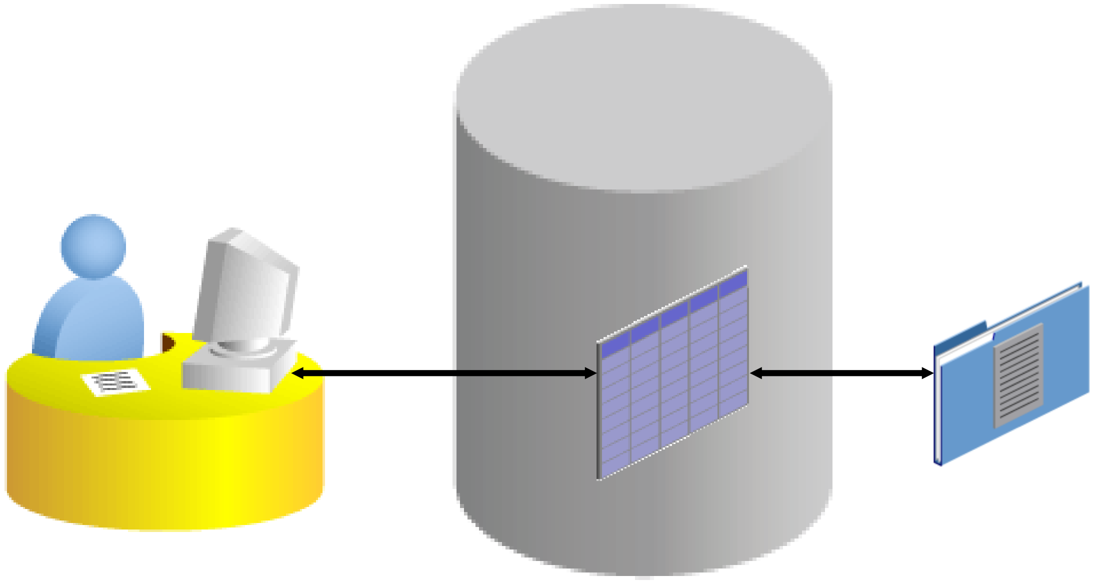
1.2: Multitable Inserts
VERSIONS BETWEEN Clause

```
SELECT versions_starttime "START_DATE",
       versions_endtime   "END_DATE",
       salary
FROM   employees
    VERSIONS BETWEEN SCN MINVALUE
    AND MAXVALUE
WHERE  last_name = 'Lorentz';
```



	START_DATE	END_DATE	SALARY
1	18-JUN-09 05.07.10.000000000 PM	(null)	5460
2	(null)	18-JUN-09 05.07.10.000000000 PM	4200

1.3: External Tables
External Tables



1.3: 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 ora_21;
```

1.3: External Tables
Creating an External Table

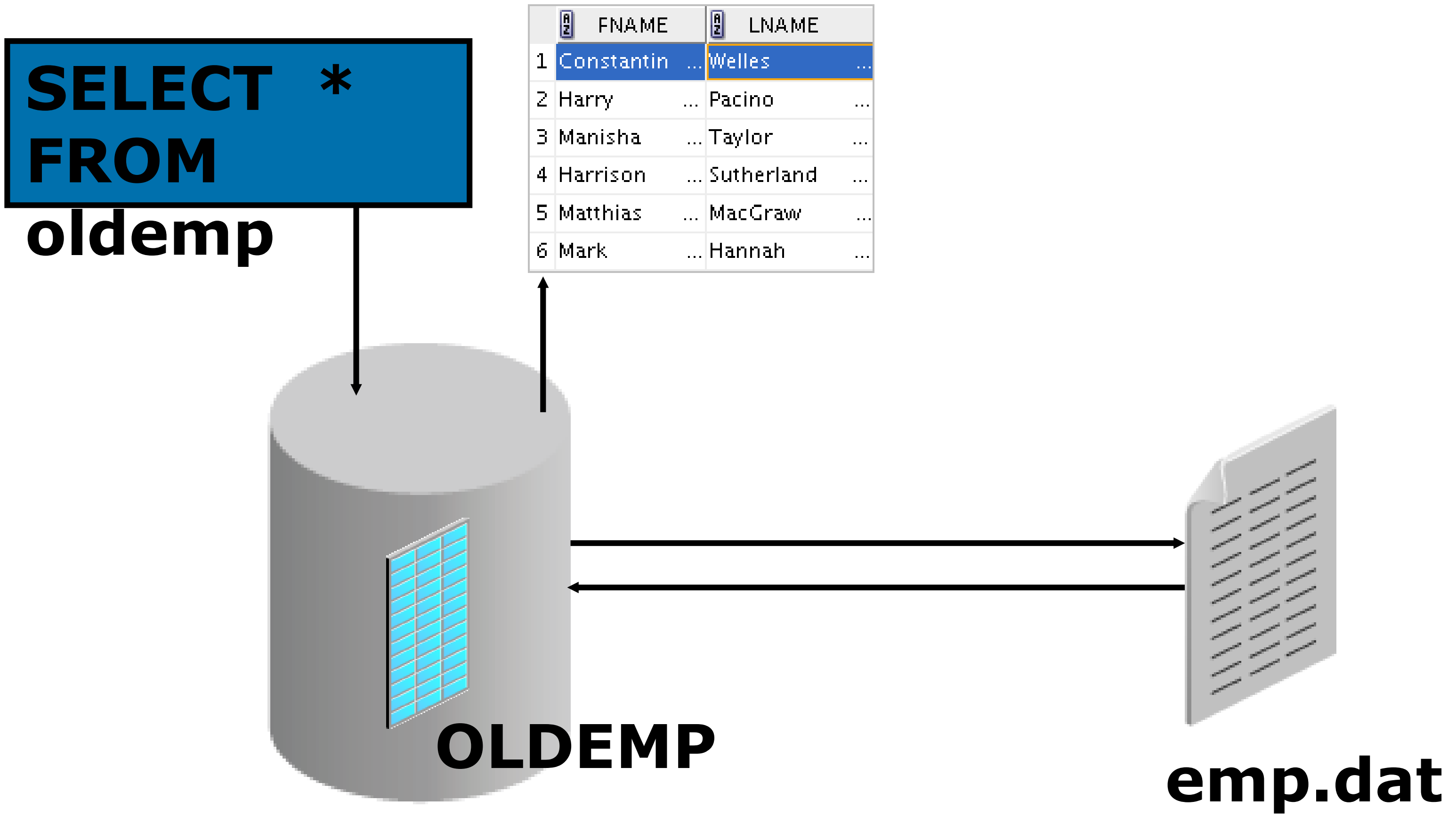
```
CREATE TABLE <table_name>
   ( <col_name> <datatype>, … )
ORGANIZATION EXTERNAL
    (TYPE <access_driver_type>
     DEFAULT DIRECTORY <directory_name>
     ACCESS PARAMETERS
      (… ) )
      LOCATION ('<location_specifier>')
REJECT LIMIT [0 | <number> | UNLIMITED];
```

1.3: External Tables
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.
```

1.3: External Tables
Querying External Tables



	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

- Differentiate system privileges from object privileges
- Grant privileges on tables
- Grant roles
- Distinguish between privileges and roles
- Use DML statements and control transactions
- Describe the features of multitable INSERTs
- Use the following types of multitable INSERTs:
- Unconditional INSERT
- Pivoting INSERT
- Conditional INSERT ALL
- Conditional INSERT FIRST
- Merge rows in a table
- Manipulate data by using subqueries
- Track the changes to data over a period of time

Review Questions

- ❖Question 1: You can use subqueries in DML statements to retrieve data by using an inline view. True/False

- ❖Question 2: _______ statement is used to create users by DBA.