Creating Other Schema Objects
Objectives

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

- Create simple and complex views
- Retrieve data from views
- Create, maintain, and use sequences
- Create and maintain indexes
- Create private and public synonyms
Lesson Agenda

• Overview of views:
  – Creating, modifying, and retrieving data from a view
  – Data manipulation language (DML) operations on a view
  – Dropping a view

• Overview of sequences:
  – Creating, using, and modifying a sequence
  – Cache sequence values
  – NEXTVAL and CURRVAL pseudocolumns

• Overview of indexes
  – Creating, dropping indexes

• Overview of synonyms
  – Creating, dropping synonyms
# Database Objects

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Basic unit of storage; composed of rows</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Generates numeric values</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of data retrieval queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Gives alternative names to objects</td>
</tr>
</tbody>
</table>
What Is a View?

**EMPLOYEES table**

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>PHONE_NUMBER</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Steven</td>
<td>King</td>
<td>SKING</td>
<td>515.123.4567</td>
<td>17-JUN-87</td>
<td>AD_PRES</td>
<td>24000</td>
</tr>
<tr>
<td>2</td>
<td>Neena</td>
<td>Kochhar</td>
<td>NKOC...</td>
<td>515.123.4568</td>
<td>21-SEP-89</td>
<td>AD_VP</td>
<td>17000</td>
</tr>
<tr>
<td>3</td>
<td>Lex</td>
<td>De Haan</td>
<td>LDEHA...</td>
<td>515.123.4569</td>
<td>13-JAN-93</td>
<td>AD_VP</td>
<td>17000</td>
</tr>
<tr>
<td>4</td>
<td>Alexander</td>
<td>Hunold</td>
<td>AHUNOLD</td>
<td>590.423.4567</td>
<td>03-JAN-90</td>
<td>IT_PROG</td>
<td>9000</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Steven</td>
<td>King</td>
<td>24000</td>
</tr>
<tr>
<td>101</td>
<td>Neena</td>
<td>Kochhar</td>
<td>17000</td>
</tr>
<tr>
<td>102</td>
<td>Lex</td>
<td>De Haan</td>
<td>17000</td>
</tr>
<tr>
<td>103</td>
<td>Alexander</td>
<td>Hunold</td>
<td>9000</td>
</tr>
<tr>
<td>104</td>
<td>Bruce</td>
<td>Ernst</td>
<td>6000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>PHONE_NUMBER</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Shelley</td>
<td>Higgins</td>
<td>SHIG...</td>
<td>515.123.8080</td>
<td>07-JUN-94</td>
<td>AC_MGR</td>
<td>12000</td>
</tr>
<tr>
<td>20</td>
<td>William</td>
<td>Getz</td>
<td>WGETZ</td>
<td>515.123.8181</td>
<td>07-JUN-94</td>
<td>AC_ACC...</td>
<td>8300</td>
</tr>
</tbody>
</table>
Advantages of Views

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data
## Simple Views and Complex Views

<table>
<thead>
<tr>
<th>Feature</th>
<th>Simple Views</th>
<th>Complex Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tables</td>
<td>One</td>
<td>One or more</td>
</tr>
<tr>
<td>Contain functions</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Contain groups of data</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DML operations through a view</td>
<td>Yes</td>
<td>Not always</td>
</tr>
</tbody>
</table>
Creating a View

- You embed a subquery in the `CREATE VIEW` statement:

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

- The subquery can contain complex `SELECT` syntax.
Creating a View

• Create the EMPVU80 view, which contains details of the employees in department 80:

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
```

CREATE VIEW succeeded.

• Describe the structure of the view by using the iSQL*Plus DESCRIBE command:

```
DESCRIBE empvu80
```
Creating a View

• Create a view by using column aliases in the subquery:

```sql
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME, salary*12 ANN_SALARY
FROM employees
WHERE department_id = 50;
```

CREATE VIEW succeeded.

• Select the columns from this view by the given alias names.
Retrieving Data from a View

```
SELECT * 
FROM salyu50; 
```

<table>
<thead>
<tr>
<th>ID_NUMBER</th>
<th>NAME</th>
<th>ANN_SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>124 Mourgos</td>
<td>69600</td>
</tr>
<tr>
<td>2</td>
<td>141 Rajs</td>
<td>42000</td>
</tr>
<tr>
<td>3</td>
<td>142 Davies</td>
<td>37200</td>
</tr>
<tr>
<td>4</td>
<td>143 Matos</td>
<td>31200</td>
</tr>
<tr>
<td>5</td>
<td>144 Vargas</td>
<td>30000</td>
</tr>
</tbody>
</table>
Modifying a View

- Modify the `EMPVU80` view by using a `CREATE OR REPLACE VIEW` clause. Add an alias for each column name:

```sql
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT employee_id, first_name || ' ' || last_name, salary, department_id
  FROM employees
  WHERE department_id = 80;
```

- Column aliases in the `CREATE OR REPLACE VIEW` clause are listed in the same order as the columns in the subquery.
Creating a Complex View

Create a complex view that contains group functions to display values from two tables:

```sql
CREATE OR REPLACE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT d.department_name, MIN(e.salary),
         MAX(e.salary), AVG(e.salary)
FROM employees e JOIN departments d
ON (e.department_id = d.department_id)
GROUP BY d.department_name;
```

CREATE OR REPLACE VIEW succeeded.
Rules for Performing DML Operations on a View

• You can usually perform DML operations on simple views.
• You cannot remove a row if the view contains the following:
  – Group functions
  – A GROUP BY clause
  – The DISTINCT keyword
  – The pseudocolumn ROWNUM keyword
Rules for Performing DML Operations on a View

You cannot modify data in a view if it contains:

- Group functions
- A GROUP BY clause
- The DISTINCT keyword
- The pseudocolumn ROWNUM keyword
- Columns defined by expressions
Rules for Performing DML Operations on a View

You cannot add data through a view if the view includes:

- Group functions
- A GROUP BY clause
- The DISTINCT keyword
- The pseudocolumn ROWNUM keyword
- Columns defined by expressions
- NOT NULL columns in the base tables that are not selected by the view
Using the **WITH CHECK OPTION** Clause

- You can ensure that DML operations performed on the view stay in the domain of the view by using the **WITH CHECK OPTION** clause:

  ```sql
  CREATE OR REPLACE VIEW empvu20
  AS SELECT * FROM employees
  WHERE department_id = 20
  WITH CHECK OPTION CONSTRAINT empvu20_ck;
  ```

- Any attempt to **INSERT** a row with a `department_id` other than 20, or to **UPDATE** the department number for any row in the view fails because it violates the **WITH CHECK OPTION** constraint.
Denying DML Operations

• You can ensure that no DML operations occur by adding the `WITH READ ONLY` option to your view definition.
• Any attempt to perform a DML operation on any row in the view results in an Oracle server error.
Denying DML Operations

```sql
CREATE OR REPLACE VIEW empvu10
  (employee_number, employee_name, job_title)
AS SELECT employee_id, last_name, job_id
  FROM employees
  WHERE department_id = 10
  WITH READ ONLY ;
```

CREATE OR REPLACE VIEW succeeded.
Removing a View

You can remove a view without losing data because a view is based on underlying tables in the database.

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
```

DROP VIEW empvu80 succeeded.
Practice 11: Overview of Part 1

This practice covers the following topics:

• Creating a simple view
• Creating a complex view
• Creating a view with a check constraint
• Attempting to modify data in the view
• Removing views
Lesson Agenda

• Overview of views:
  – Creating, modifying, and retrieving data from a view
  – DML operations on a view
  – Dropping a view

• Overview of sequences:
  – Creating, using, and modifying a sequence
  – Cache sequence values
  – `NEXTVAL` and `CURRVAL` pseudocolumns

• Overview of indexes
  – Creating, dropping indexes

• Overview of synonyms
  – Creating, dropping synonyms
Sequences

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Basic unit of storage; composed of rows</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Generates numeric values</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of some queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Gives alternative names to objects</td>
</tr>
</tbody>
</table>
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
CREATE SEQUENCE Statement: Syntax

Define a sequence to generate sequential numbers automatically:

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

Creating a Sequence

- Create a sequence named `DEPT_DEPTID_SEQ` to be used for the primary key of the `DEPARTMENTS` table.
- Do not use the `CYCLE` option.

```
CREATE SEQUENCE dept_deptid_seq
    INCREMENT BY 10
    START WITH 120
    MAXVALUE 9999
    NOCACHE
    NOCYCLE;
```

`CREATE SEQUENCE succeeded.`
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.
Using a Sequence

- Insert a new department named “Support” in location ID 2500:

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

1 row inserted

- View the current value for the `DEPT_DEPTID_SEQ` sequence:

```sql
SELECT dept_deptid_seq.CURRVAL
FROM dual;
```
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
Modifying a Sequence

Change the increment value, maximum value, minimum value, cycle option, or cache option:

```sql
ALTER SEQUENCE dept_deptid_seq
  INCREMENT BY 20
  MAXVALUE 999999
  NOCACHE
  NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq succeeded.
```
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;
DROP SEQUENCE dept_deptid_seq succeeded.
```
Lesson Agenda

• Overview of views:
  – Creating, modifying, and retrieving data from a view
  – DML operations on a view
  – Dropping a view

• Overview of sequences:
  – Creating, using, and modifying a sequence
  – Cache sequence values
  – NEXTVAL and CURRVAL pseudocolumns

• Overview of indexes
  – Creating, dropping indexes

• Overview of synonyms
  – Creating, dropping synonyms
## Indexes

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Basic unit of storage; composed of rows</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Generates numeric values</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of some queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Gives alternative names to objects</td>
</tr>
</tbody>
</table>
Indexes

An index:

- Is a schema object
- May 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 independent of the table that it indexes
- Is used and maintained automatically by the Oracle server
How Are Indexes Created?

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

• Manually: Users can create nonunique indexes on columns to speed up access to the rows.
Creating an Index

- Create an index on one or more columns:

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

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

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

  CREATE INDEX succeeded.
# Index Creation Guidelines

<table>
<thead>
<tr>
<th><strong>Create an index when:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ A column contains a wide range of values</td>
<td></td>
</tr>
<tr>
<td>✓ A column contains a large number of null values</td>
<td></td>
</tr>
<tr>
<td>✓ One or more columns are frequently used together in a <code>WHERE</code> clause or a join condition</td>
<td></td>
</tr>
<tr>
<td>✓ The table is large and most queries are expected to retrieve less than 2% to 4% of the rows in the table</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Do not create an index when:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✗ The columns are not often used as a condition in the query</td>
<td></td>
</tr>
<tr>
<td>✗ The table is small or most queries are expected to retrieve more than 2% to 4% of the rows in the table</td>
<td></td>
</tr>
<tr>
<td>✗ The table is updated frequently</td>
<td></td>
</tr>
<tr>
<td>✗ The indexed columns are referenced as part of an expression</td>
<td></td>
</tr>
</tbody>
</table>
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;

    DROP INDEX emp_last_name_idx succeeded.

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

- Overview of views:
  - Creating, modifying, and retrieving data from a view
  - DML operations on a view
  - Dropping a view
- Overview of sequences:
  - Creating, using, and modifying a sequence
  - Cache sequence values
  - \texttt{NEXTVAL} and \texttt{CURRVAL} pseudocolumns
- Overview of indexes
  - Creating, dropping indexes
- Overview of synonyms
  - Creating, dropping synonyms
## Synonyms

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Basic unit of storage; composed of rows</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Generates numeric values</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of some queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Gives alternative names to objects</td>
</tr>
</tbody>
</table>
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

```sql
CREATE [PUBLIC] SYNONYM synonym FOR object;
```
Creating and Removing Synonyms

• Create a shortened name for the `DEPT_SUM_VU` view:

```
CREATE SYNONYM d_sum
FOR dept_sum_vu;
CREATE SYNONYM succeeded.
```

• Drop a synonym:

```
DROP SYNONYM d_sum;
DROP SYNONYM d_sum succeeded.
```
Quiz

Indexes must be created manually and serve to speed up access to rows in a table.

1. True
2. False
Summary

In this lesson, you should have learned how to:

• Create, use, and remove views
• Automatically generate sequence numbers by using a sequence generator
• Create indexes to improve speed of query retrieval
• Use synonyms to provide alternative names for objects
Practice 11: Overview of Part 2

This practice covers the following topics:

• Creating sequences
• Using sequences
• Creating nonunique indexes
• Creating synonyms