

# Creating a View

- You embed a subquery in the CREATE VIEW statement:

```
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 employees in department 80:

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

- Describe the structure of the view by using the *DESCRIBE* command:

```
DESCRIBE empvu80
```

# Creating a View

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

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

- Select the columns from this view by the given alias names:

# Retrieving Data from a View

```
SELECT *  
FROM salvu50;
```

# Modifying a View

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

```
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:

```
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;
```

# 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:

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

- Any attempt to change 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

```
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 ;
```

# 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;
```

# Sequences

**A sequence:**

- **Can automatically generate unique numbers**
- **Is a sharable 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 / NOMINVALUE}]  
[{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;
```



# **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:

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

- View the current value for the DEPT\_DEPTID\_SEQ sequence:

```
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:

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

# 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;
```

# 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 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:

```
CREATE 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 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

# Removing an Index

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

```
DROP INDEX index;
```

- Remove the UPPER\_LAST\_NAME\_IDX index from the data dictionary:

```
DROP INDEX emp_last_name_idx;
```

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

# Synonyms

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;
```

# Creating and Removing Synonyms

- Create a shortened name for the DEPT\_SUM\_VU view:

```
CREATE SYNONYM d_sum  
FOR dept_sum_vu;
```

- Drop a synonym:

```
DROP SYNONYM d_sum;
```