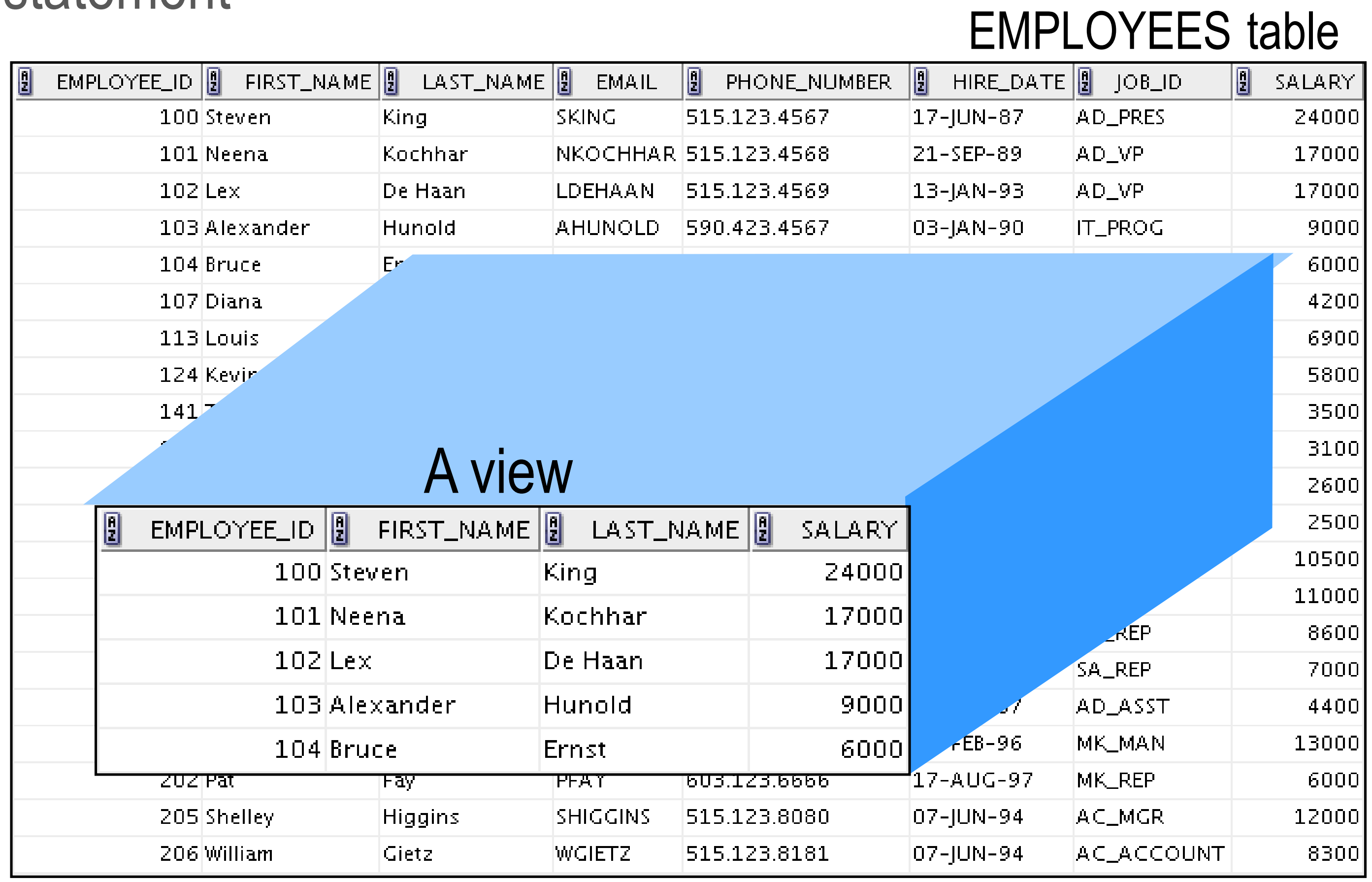
greatlearning
Learning for Life

Creating Other Schema Objects

VIEWS

- A database view is a whole or part of one or more table(s)
- Created using SELECT statement
- Also referred as

  VIRTUAL TABLE.



EMPLOYEES table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Bruce	Er					6000
107	Diana						4200
113	Louis						6900
124	Kevin						5800
141	[illegible]						3500
[illegible]							3100
							2600
							2500
							10500
							11000
						[illegible]	8600
						SA_REP	7000
					[illegible]	AD_ASST	4400
					[illegible]	MK_MAN	13000
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

A view

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
103	Alexander	Hunold	9000
104	Bruce	Ernst	6000

ADVANTAGES OF VIEWS

To restrict data access

To make complex queries easy

To provide data independence

To present different views of the same data

CREATING A VIEW

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

```
CREATE VIEW         dept80_emp
 AS SELECT  employee_id, first_name, last_name, salary
    FROM    employees
    WHERE   department_id = 80;
```

CREATING A VIEW

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

```
CREATE VIEW          salvu50
 AS SELECT  employee_id, first_name || ' ' last_name AS     fullname, salary*12
annual_sal
    FROM    employees
    WHERE   department_id = 50;
```

- Select the columns from this view by the given alias names.
- This view definition can be modified using:

```
CREATE OR REPLACE VIEW salvu50
 AS SELECT ...
```

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

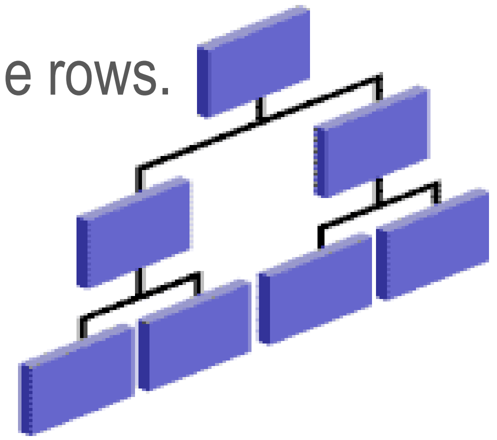
- You can ensure that DML operations performed on the view stay in the domain of the view by using the WITH CHECK OPTION clause:

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

- 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.

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 input/output (I/O) by using a rapid path access method to locate data quickly
  - Is dependent on the table that it indexes
  - Is used and maintained automatically by the Oracle server
- A unique index is created automatically when a PRIMARY KEY or UNIQUE constraint defined in a table definition.
- 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 [UNIQUE]INDEX index_name
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);
```

- Indexes get used when a query searches for a involving indexed column:

```
SELECT * FROM Employees WHERE last_name LIKE 'Boo%';
```

- Index can be dropped using DROP INDEX statement:

```
DROP INDEX emp_last_name_idx;
```

INDEX CREATION GUIDLINES

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

SUMMARY

- Use group or aggregate functions in SQL
- Retrieving data from multiple tables using joins
- Use of sub-queries
- Set operations on tables
- Manipulating data
- Creating different database objects

greatlearning
Learning for Life

THANK YOU