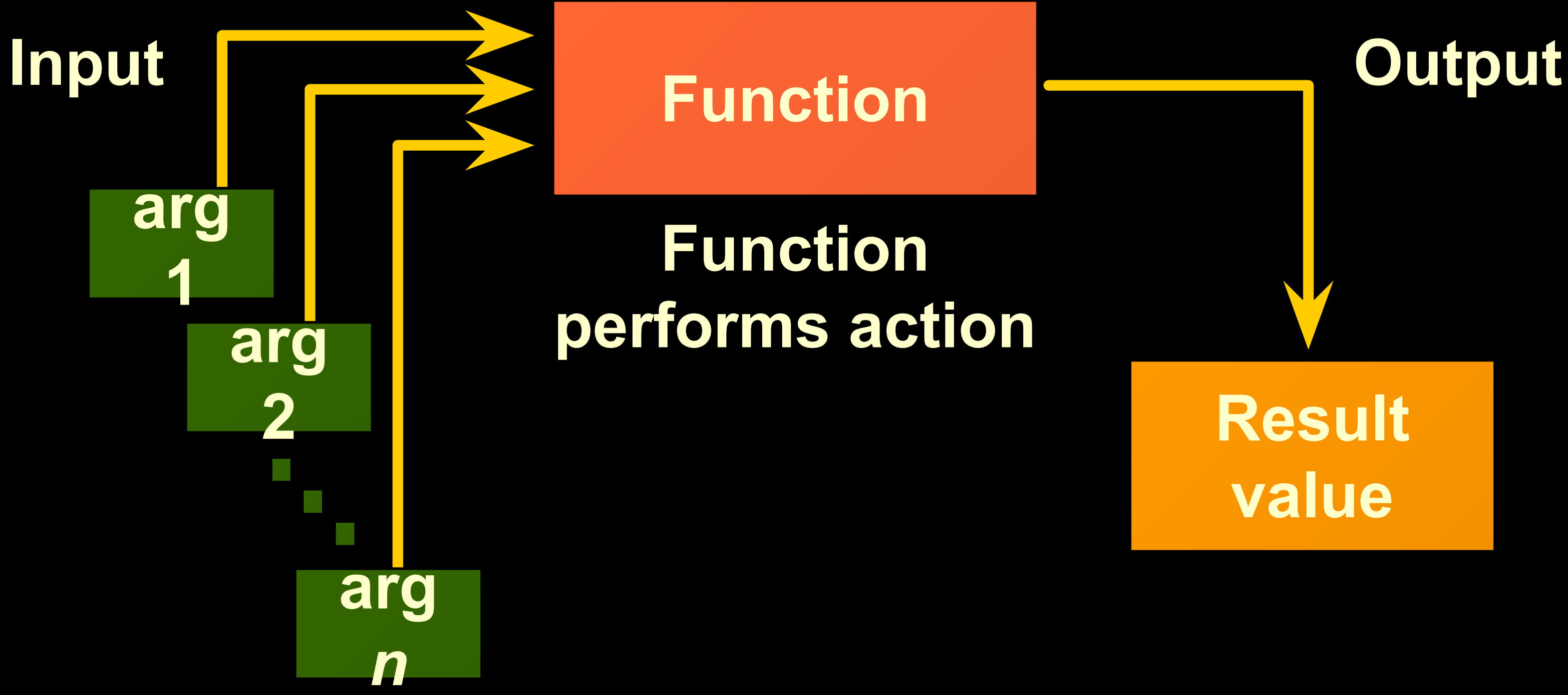
Single-Row Functions

Objectives

- After completing this lesson, you should be able to do the following:
  - Describe various types of functions available in SQL
  - Use character, number, and date functions in SELECT statements
  - Describe the use of conversion functions

SQL Functions



Input

arg 1

arg 2

arg *n*

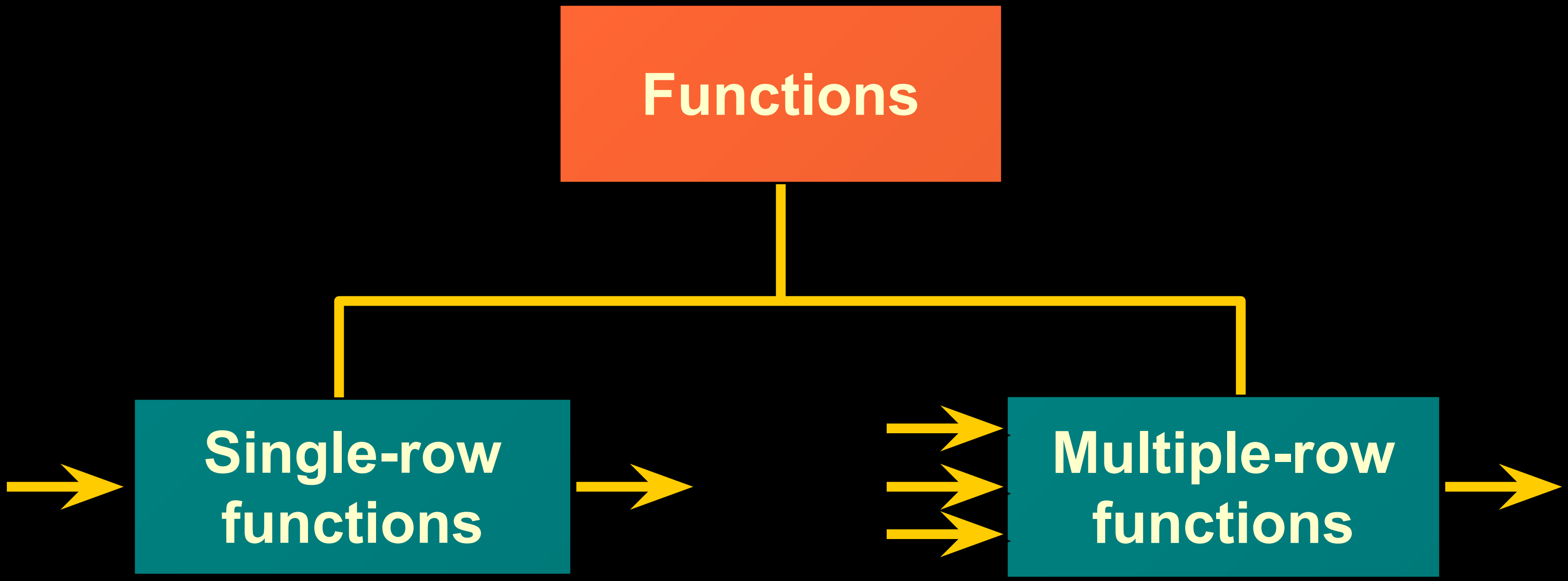
Function

Function performs action

Output

Result value

Two Types of SQL Functions

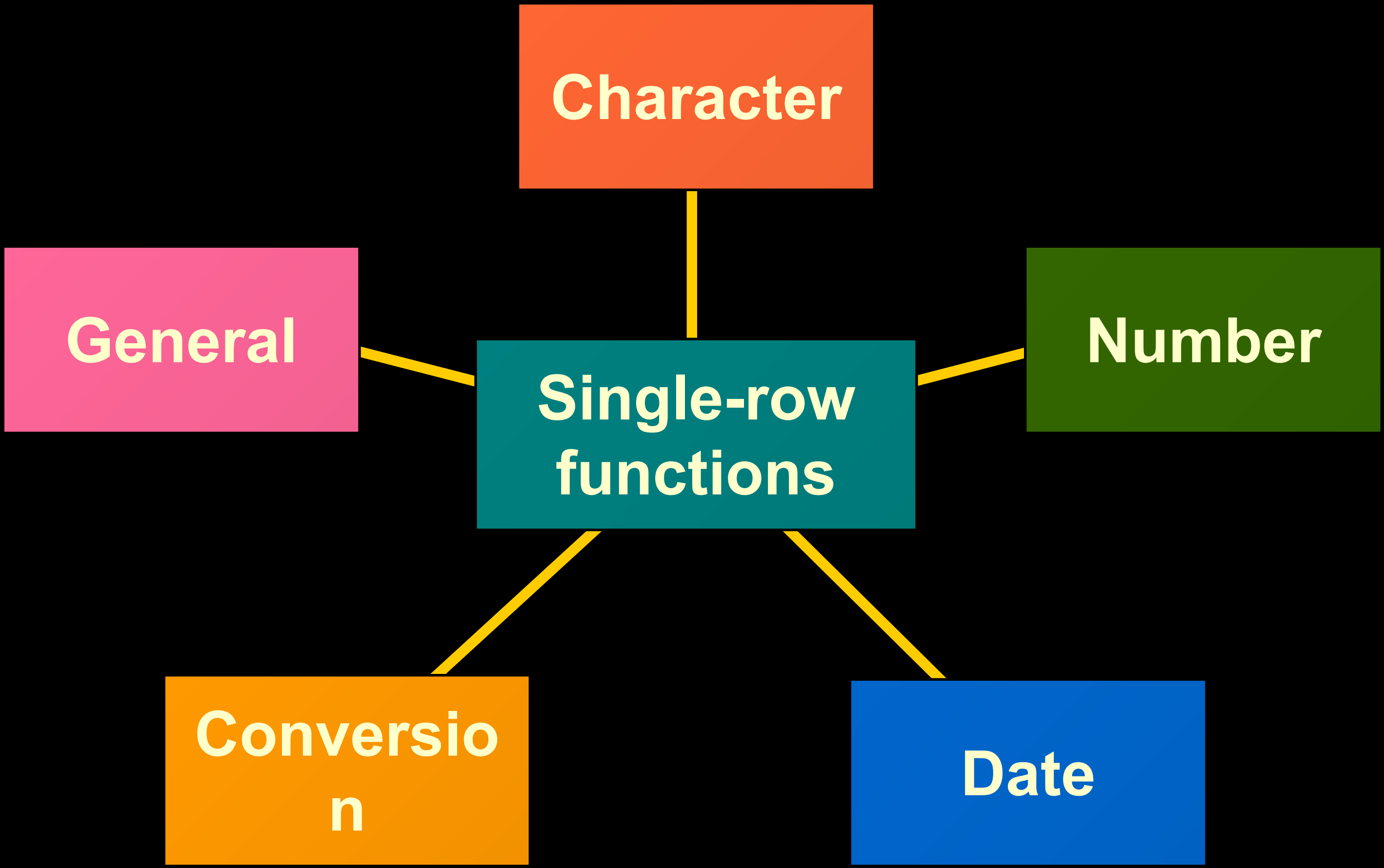


Single-Row Functions

- Manipulate data items
- Accept arguments and return one value
- Act on each row returned
- Return one result per row
- May modify the datatype
- Can be nested

```
function_name (column|expression, [arg1, arg2,...])
```

Single-Row Functions



Character

General

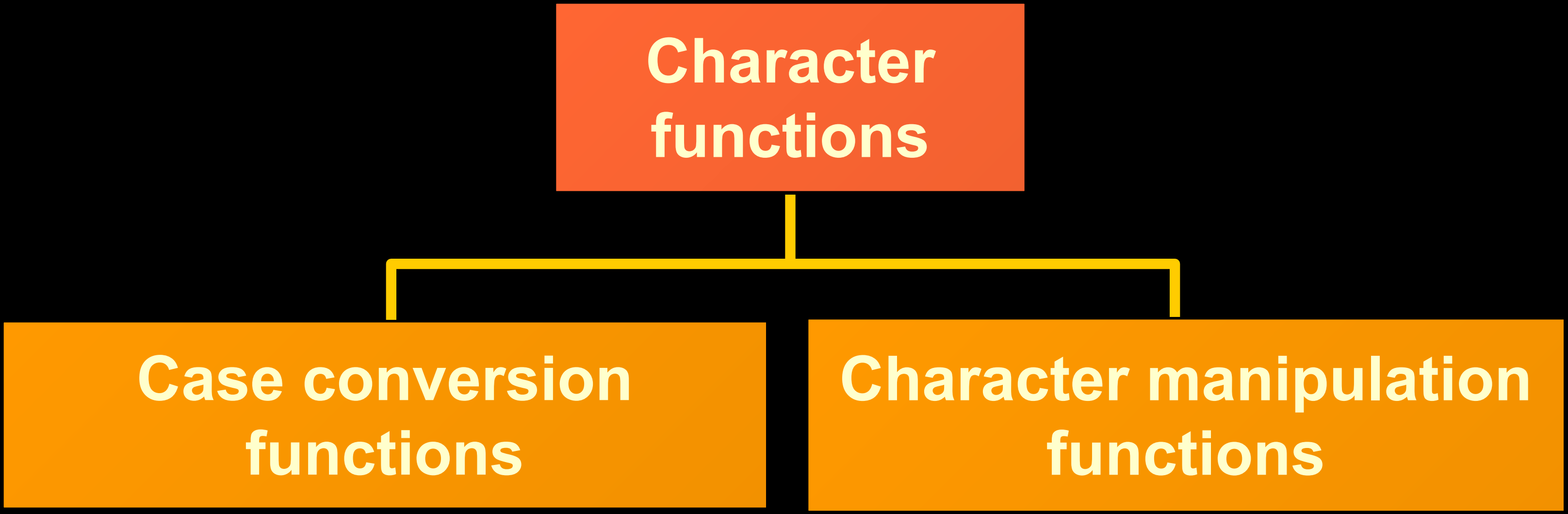
Single-row functions

Number

Conversio
n

Date

Character Functions



Character functions

Case conversion functions

Character manipulation functions

LOWER
UPPER
INITCAP

CONCAT
SUBSTR
LENGTH
INSTR
LPAD

Case Conversion Functions

- Convert case for character strings

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

Using Case Conversion Functions

- Display the employee number, name, and department number for employee Blake.

```
SQL> SELECT   empno, ename, deptno
  2  FROM emp
  3  WHERE    ename = 'blake';
no rows selected
```

```
SQL> SELECT   empno, ename, deptno
  2  FROM emp
  3  WHERE    LOWER(ename) = 'blake';
```

```
    EMPNO ENAME            DEPTNO
--------- ---------- ---------
     7698 BLAKE              30
```

Character Manipulation Functions

- Manipulate character strings

Function	Result
CONCAT('Good', 'String')	GoodString
SUBSTR('String',1,3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	******5000

Using the Character Manipulation Functions

```
SQL> SELECT ename, CONCAT (ename, job),
LENGTH(ename),
    2    INSTR(ename, 'A')
    3 FROM    emp
    4 WHERE   SUBSTR(job,1,5) = 'SALES';
```

```
 ENAME       CONCAT(ENAME,JOB)    LENGTH(ENAME) INSTR(ENAME,'A')
 ----------  -------------------- ------------- ----------------
 MARTIN      MARTINSALESMAN                   6                2
 ALLEN       ALLENSALESMAN                    5                1
 TURNER      TURNERSALESMAN                   6                0
 WARD        WARDSALESMAN                     4                2
```

Number Functions

- ROUND:Rounds value to specified decimal

  ROUND(45.926, 2) → 45.93

- TRUNC:Truncates value to specified decimal

  TRUNC(45.926, 2) → 45.92

- MOD: Returns remainder of division

  MOD(1600, 300) → 100

Using the ROUND Function

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2         ROUND(45.923,-1)
  3  FROM   DUAL;
```

```
ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
          45.92              46               50
```

Using the TRUNC Function

```
SQL> SELECT   TRUNC(45.923,2), TRUNC(45.923),
  2        TRUNC(45.923,-1)
  3  FROM    DUAL;
```

```
TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-1)
--------------- ------------- ----------------
          45.92            45               40
```

Working with Dates

- – Oracle stores dates in an internal numeric format: century, year, month, day, hours, minutes, seconds.
- – The default date format is DD-MON-YY.
- – SYSDATE is a function returning date and time.

Arithmetic with Dates

- Add or subtract a number to or from a date for a resultant *date* value.
- Subtract two dates to find the *number* of days between those dates.
- Add *hours* to a date by dividing the number of hours by 24.

Using Arithmetic Operators with Dates

```
SQL> SELECT ename, (SYSDATE-hiredate)/7 WEEKS
  2  FROM   emp
  3  WHERE  deptno = 10;
```

```
ENAME           WEEKS
---------- ----------
KING        830.93709
CLARK       853.93709
MILLER      821.36566
```

Date Functions

Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

Using Date Functions

- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') → 19.6774194

- ADD_MONTHS ('11-JAN-94',6) → '11-JUL-94'

- NEXT_DAY ('01-SEP-95','FRIDAY') → '08-SEP-95'

- LAST_DAY('01-SEP-95') → '30-SEP-95'

DECODE Function

- Facilitates conditional inquiries by doing the work of a CASE or IF-THEN-ELSE statement

```
DECODE(col/expression, search1, result1
                 [, search2, result2,...,]
                 [, default])
```

Using the DECODE Function

```
SQL> SELECT job, sal,
  2         DECODE(job, 'ANALYST',  SAL*1.1,
  3                     'CLERK',   SAL*1.15,
  4                     'MANAGER', SAL*1.20,
  5                                SAL)
  6               REVISED_SALARY
  7  FROM   emp;
```

```
JOB              SAL REVISED_SALARY
--------- ---------- --------------
PRESIDENT       5000           5000
MANAGER         2850           3420
MANAGER         2450           2940
...
14 rows selected.
```

Using the DECODE Function

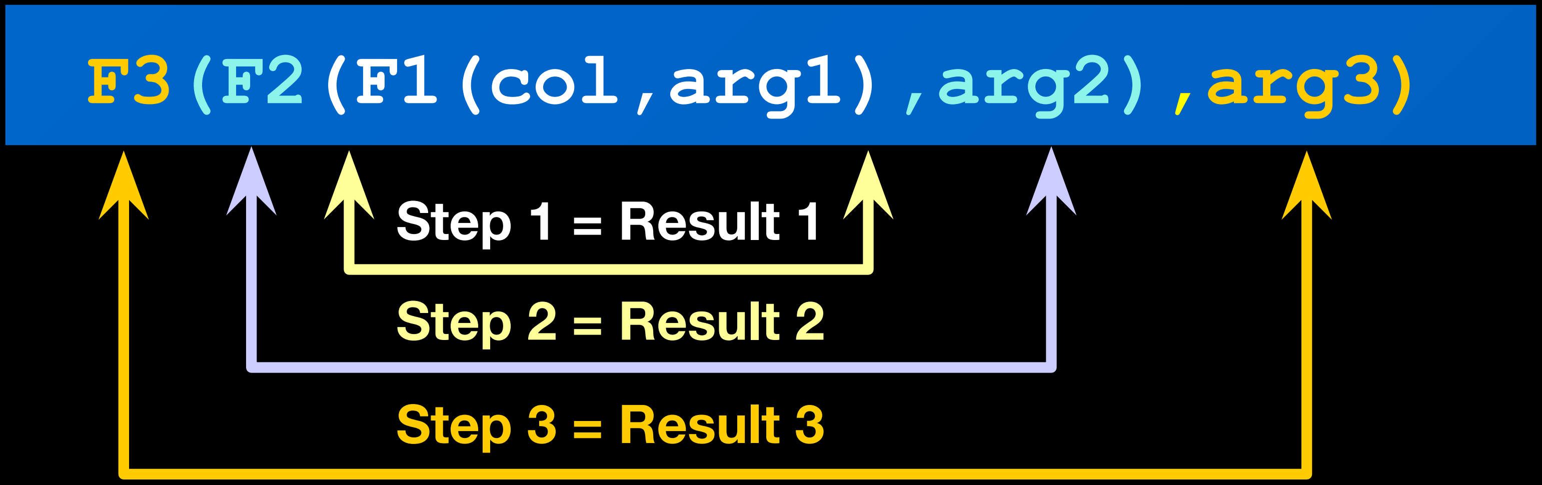
- Display the applicable tax rate for each employee in department 30.

```
SQL> SELECT ename, sal,
  2         DECODE(TRUNC(sal/1000, 0),
  3                          0, 0.00,
  4                1, 0.09,
  5                          2, 0.20,
  6                          3, 0.30,
  7                          4, 0.40,
  8                          5, 0.42,
  9                          6, 0.44,
 10                             0.45)
 11  FROM    emp
 12  WHERE   deptno = 30;
```

Nesting Functions

– Single-row functions can be nested to any level.

– Nested functions are evaluated from deepest level to the least-deep level.



```
F3(F2(F1(col,arg1),arg2),arg3)
```

Step 1 = Result 1

Step 2 = Result 2

Step 3 = Result 3

Practice Questions

- Print the following
- <employee name> earns <salary> monthly but wants <3 times salary> label the column dream salaries

Practice Questions

- Display the employee name, with the first letter capitalized and all other letter lowercase and the length of their name for all the employees whose name started with A, F or M.

Practice Questions

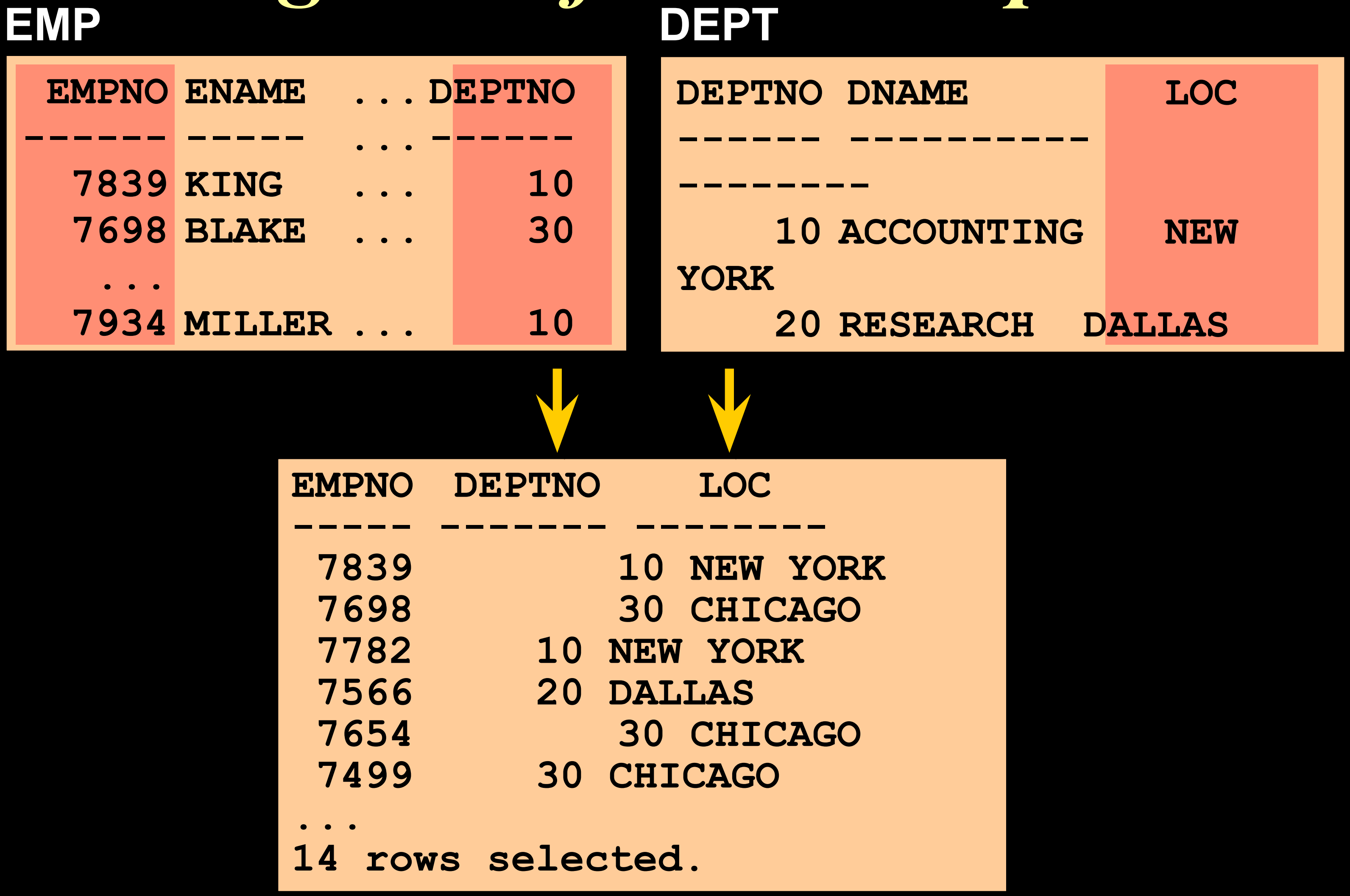
- Display the employee name and commission in percentage. If the employee does not earn commission display No commission

Displaying Data from Multiple Tables

Objectives

- After completing this lesson, you should be able to do the following:
  - Write SELECT statements to access data from more than one table using equality and nonequality joins
  - View data that generally does not meet a join condition by using outer joins
  - Join a table to itself

Obtaining Data from Multiple Tables



EMP

```
 EMPNO ENAME    ...DEPTNO
------ -----    ...------
  7839 KING     ...    10
  7698 BLAKE    ...    30
   ...
  7934 MILLER ...      10
```

DEPT

```
DEPTNO DNAME          LOC
------ ----------
---------
    10 ACCOUNTING    NEW
YORK
    20 RESEARCH   DALLAS
```

```
EMPNO  DEPTNO    LOC
----- ------- ---------
 7839         10 NEW YORK
 7698         30 CHICAGO
 7782      10 NEW YORK
 7566      20 DALLAS
 7654         30 CHICAGO
 7499      30 CHICAGO
...
14 rows selected.
```

What Is a Join?

- Use a join to query data from more than one table.

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column1 = table2.column2;
```

  - Write the join condition in the WHERE clause.
  - Prefix the column name with the table name when the same column name appears in more than one table.

Cartesian Product

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

Generating a Cartesian Product

EMP (14 rows)

```
 EMPNO ENAME   ...DEPTNO
------ -----   ...------
  7839 KING    ...     10
  7698 BLAKE   ...     30
   ...
  7934 MILLER ...      10
```

DEPT (4 rows)

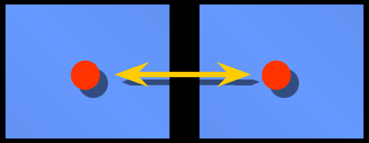
```
DEPTNO DNAME          LOC
------ ----------
---------
      10 ACCOUNTING    NEW
YORK
      20 RESEARCH   DALLAS
```

```
ENAME      DNAME
------
----------
KING
ACCOUNTING
BLAKE
ACCOUNTING
...
KING       RESEARCH
```

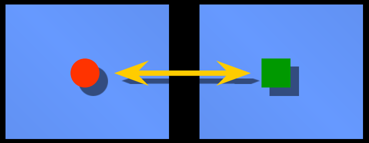
"Cartesian product: 14*4=56 rows"

Types of Joins

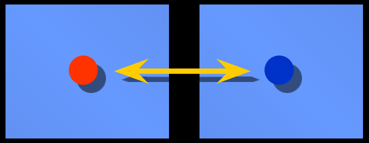
Equijoin



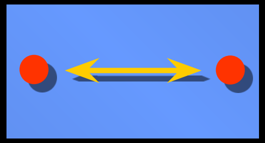
Non-equijoin



Outer join



Self join



What Is an Equijoin?

EMP

```
  EMPNO ENAME      DEPTNO
------- -------- --------
   7839 KING           10
   7698 BLAKE          30
   7782 CLARK          10
   7566 JONES          20
   7654 MARTIN         30
   7499 ALLEN          30
   7844 TURNER         30
   7900 JAMES          30
   7521 WARD           30
   7902 FORD           20
   7369 SMITH          20
...
14 rows selected.
```

DEPT

```
  DEPTNO DNAME       LOC
-------- ----------- ---------
      10 ACCOUNTING NEW YORK
      30 SALES       CHICAGO
      10 ACCOUNTING  NEW
YORK
      20 RESEARCH DALLAS
      30 SALES       CHICAGO
      30 SALES       CHICAGO
      30 SALES       CHICAGO
      30 SALES       CHICAGO
      30 SALES       CHICAGO
      20 RESEARCH DALLAS
      20 RESEARCH DALLAS
...
```

Foreign key

Primary key

Retrieving Records with Equijoins

```
SQL> SELECT   emp.empno, emp.ename, emp.deptno,
  2     dept.deptno, dept.loc
  3  FROM     emp, dept
  4  WHERE    emp.deptno=dept.deptno;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------- ------ ------ ---------
 7839 KING        10     10 NEW YORK
 7698 BLAKE       30     30 CHICAGO
 7782 CLARK       10     10 NEW YORK
 7566 JONES       20     20 DALLAS
...
14 rows selected.
```

Qualifying Ambiguous Column Names

- – Use table prefixes to qualify column names that are in multiple tables.
- – Improve performance by using table prefixes.
- – Distinguish columns that have identical names but reside in different tables by using column aliases.

Additional Search Conditions Using the AND Operator

EMP

```
 EMPNO ENAME     DEPTNO
------ -------- --------
  7839 KING          10
  7698 BLAKE         30
  7782 CLARK         10
  7566 JONES         20
  7654 MARTIN        30
  7499 ALLEN         30
  7844 TURNER        30
  7900 JAMES         30
  7521 WARD          30
  7902 FORD          20
  7369 SMITH         20
...
14 rows selected.
```

DEPT

```
DEPTNO DNAME         LOC
------ --------- --------
    10 ACCOUNTING   NEW
YORK
    30 SALES       CHICAGO
    10 ACCOUNTING   NEW
YORK
    20 RESEARCH  DALLAS
    30 SALES       CHICAGO
    30 SALES       CHICAGO
    30 SALES       CHICAGO
    30 SALES       CHICAGO
    30 SALES       CHICAGO
    20 RESEARCH  DALLAS
    20 RESEARCH  DALLAS
```

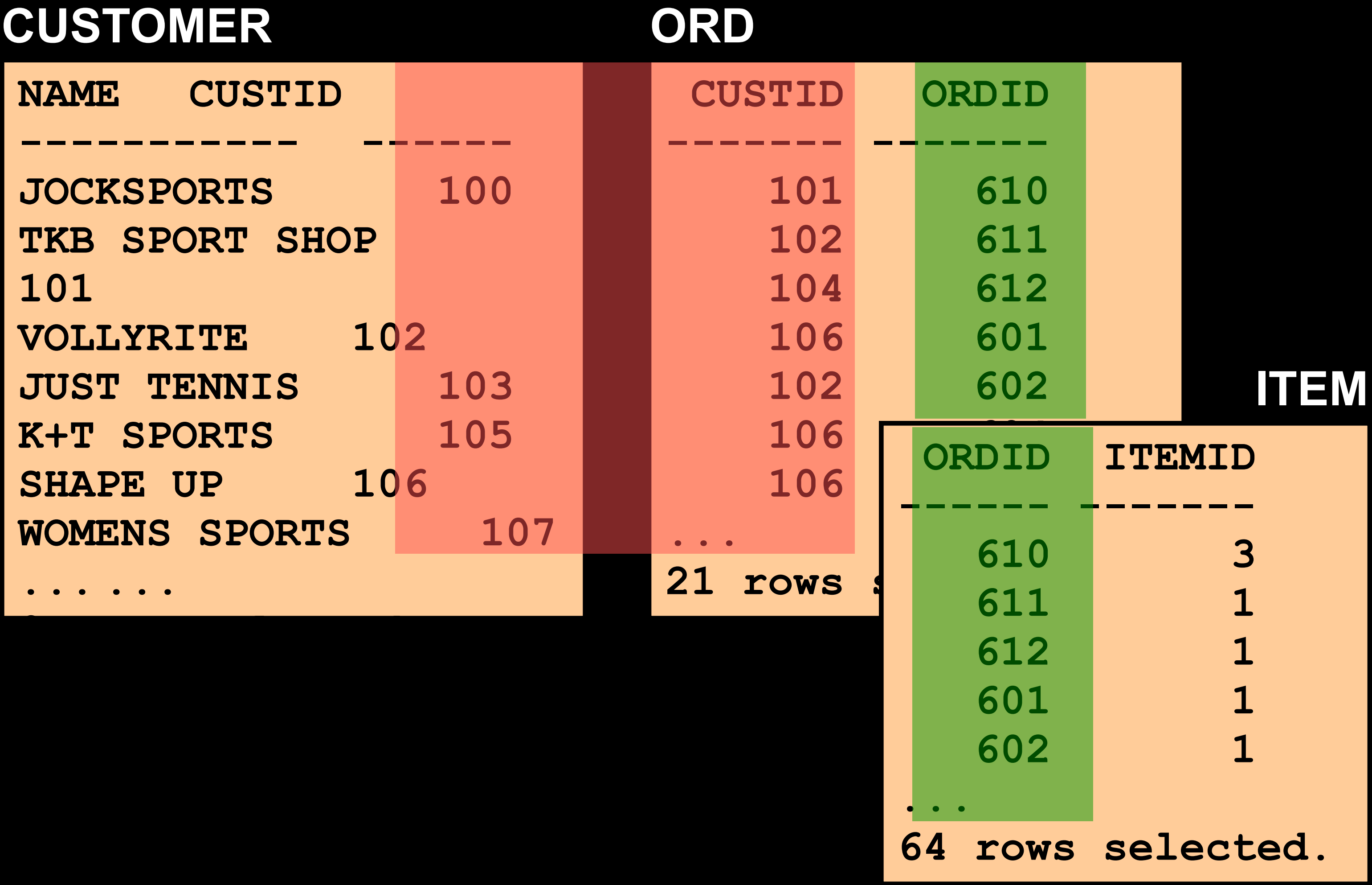
Using Table Aliases

- Simplify queries by using table aliases.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
  2    dept.deptno, dept.loc
  3  FROM   emp, dept
  4  WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e, dept d
  4  WHERE  e.deptno=d.deptno;
```

Joining More Than Two Tables



CUSTOMER

```
NAME          CUSTID
-----------   -------
JOCKSPORTS          100
TKB SPORT SHOP
101
VOLLYRITE      102
JUST TENNIS         103
K+T SPORTS          105
SHAPE UP       106
WOMENS SPORTS        107
... ...
```

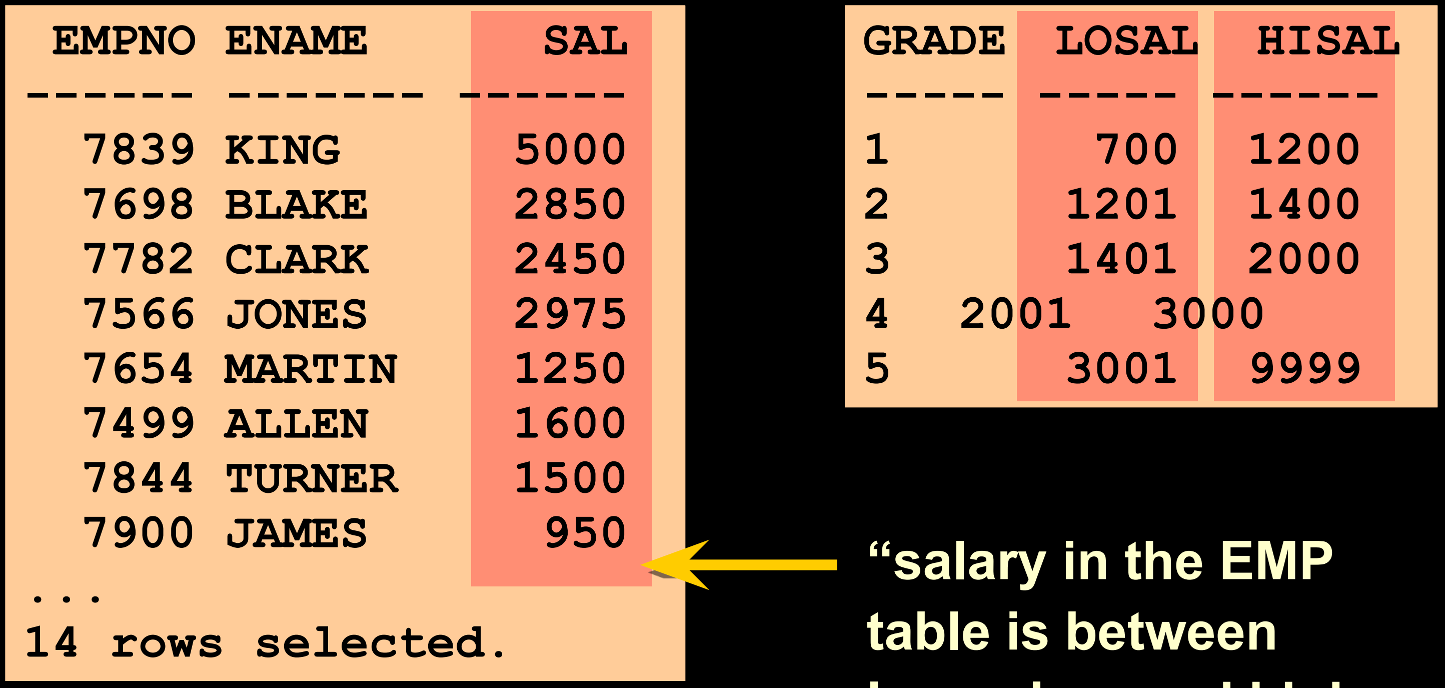
ORD

```
 CUSTID    ORDID
------- -------
    101      610
    102      611
    104      612
    106      601
    102      602
    106
    106
...
21 rows s
```

ITEM

```
 ORDID  ITEMID
------ -------
   610       3
   611       1
   612       1
   601       1
   602       1
...
64 rows selected.
```

Non-Equijoins



EMP

```
 EMPNO ENAME          SAL
------- -------- -------
   7839 KING         5000
   7698 BLAKE        2850
   7782 CLARK        2450
   7566 JONES        2975
   7654 MARTIN       1250
   7499 ALLEN        1600
   7844 TURNER       1500
   7900 JAMES         950
...
14 rows selected.
```

SALGRADE

```
GRADE  LOSAL   HISAL
----- ------ -------
1        700    1200
2       1201    1400
3       1401    2000
4   2001    3000
5       3001    9999
```

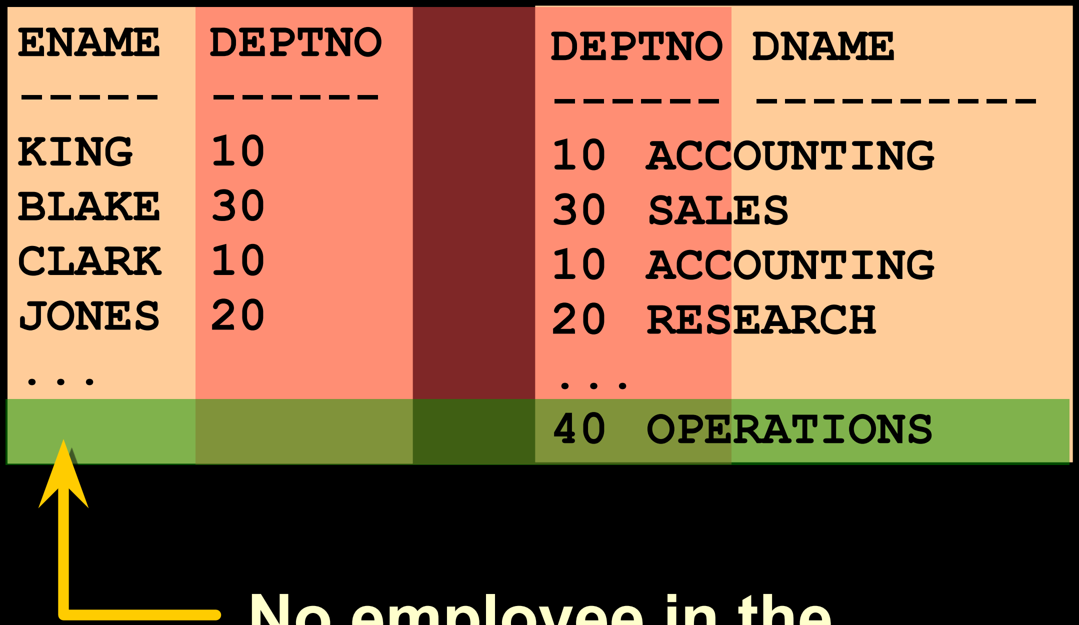
"salary in the EMP table is between low salary and high salary in the SALGRADE table"

Retrieving Records with Non-Equijoins

```
SQL>  SELECT   e.ename, e.sal, s.grade
  2   FROM   emp e, salgrade s
  3   WHERE  e.sal
  4   BETWEEN   s.losal AND s.hisal;
```

```
ENAME             SAL      GRADE
---------- ---------- ----------
JAMES             950          1
SMITH             800          1
ADAMS            1100          1
...
14 rows selected.
```

Outer Joins



EMP

```
ENAME   DEPTNO
-----   -------
KING    10
BLAKE   30
CLARK   10
JONES   20
...
```

DEPT

```
DEPTNO  DNAME
------- ----------
10   ACCOUNTING
30   SALES
10   ACCOUNTING
20   RESEARCH
...
40   OPERATIONS
```

No employee in the OPERATIONS department

Outer Joins

- – You use an outer join to also see rows that do not usually meet the join condition.
- – Outer join operator is the plus sign (+).

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

Using Outer Joins

```
SQL> SELECT   e.ename, d.deptno, d.dname
  2  FROM emp e, dept d
  3  WHERE    e.deptno(+) = d.deptno
  4  ORDER BY e.deptno;
```

```
ENAME          DEPTNO DNAME
---------- ---------- --------------
KING               10 ACCOUNTING
CLARK              10 ACCOUNTING
...
                   40 OPERATIONS
15 rows selected.
```

Left Outer Join

- SELECT d.department_id, e.last_name
  FROM
  departments d
  LEFT OUTER JOIN
  employees e
  ON
  d.department_id = e.department_id

Self Joins

EMP (WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP (MANAGER)

EMPNO	ENAME
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE

“MGR in the WORKER table is equal to EMPNO in the MANAGER table”

Joining a Table to Itself

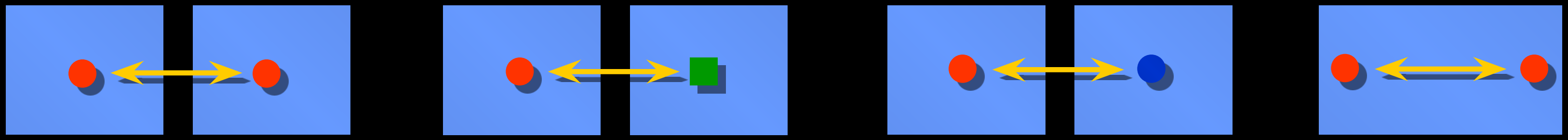
```
SQL> SELECT worker.ename||' works for '||manager.ename
  2  FROM     emp worker, emp manager
  3  WHERE    worker.mgr = manager.empno;
```

```
WORKER.ENAME||'WORKSFOR'||MANAG
-------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

Summary

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column1 = table2.column2;
```

Equijoin   Non-equijoin   Outer join   Self join



Practice Overview

- Joining tables using an equijoin
- Performing outer and self joins
- Adding conditions

Displaying Data from Multiple Tables

Objectives

- After completing this lesson, you should be able to do the following:
  - Write SELECT statements to access data from more than one table using equality and nonequality joins
  - View data that generally does not meet a join condition by using outer joins
  - Join a table to itself

Obtaining Data from Multiple Tables

EMP

```
 EMPNO ENAME    ...DEPTNO
------ -----    ...------
  7839 KING     ...    10
  7698 BLAKE    ...    30
   ...
  7934 MILLER ...      10
```

DEPT

```
DEPTNO DNAME         LOC
------ ----------
---------
    10 ACCOUNTING    NEW
YORK
    20 RESEARCH   DALLAS
```

```
EMPNO  DEPTNO    LOC
----- ------- ---------
 7839           10 NEW YORK
 7698           30 CHICAGO
 7782       10 NEW YORK
 7566       20 DALLAS
 7654           30 CHICAGO
 7499       30 CHICAGO
...
14 rows selected.
```

What Is a Join?

- Use a join to query data from more than one table.

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column1 = table2.column2;
```

  - Write the join condition in the WHERE clause.
  - Prefix the column name with the table name when the same column name appears in more than one table.

Cartesian Product

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

Generating a Cartesian Product

EMP (14 rows)

```
 EMPNO ENAME    ... DEPTNO
------ -----    ... ------
  7839 KING     ...     10
  7698 BLAKE    ...     30
   ...
  7934 MILLER   ...     10
```

DEPT (4 rows)

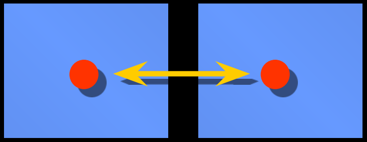
```
DEPTNO DNAME          LOC
------ ----------
---------
    10 ACCOUNTING     NEW
YORK
    20 RESEARCH   DALLAS
```

"Cartesian product: 14*4=56 rows"

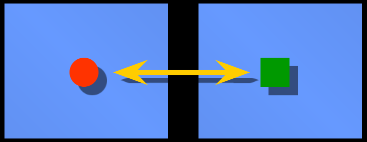
```
ENAME     DNAME
------
----------
KING
ACCOUNTING
BLAKE
ACCOUNTING
...
KING      RESEARCH
```

Types of Joins

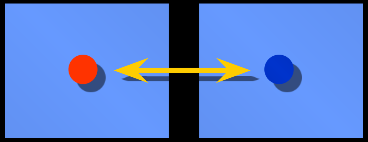
Equijoin



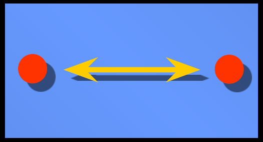
Non-equijoin



Outer join



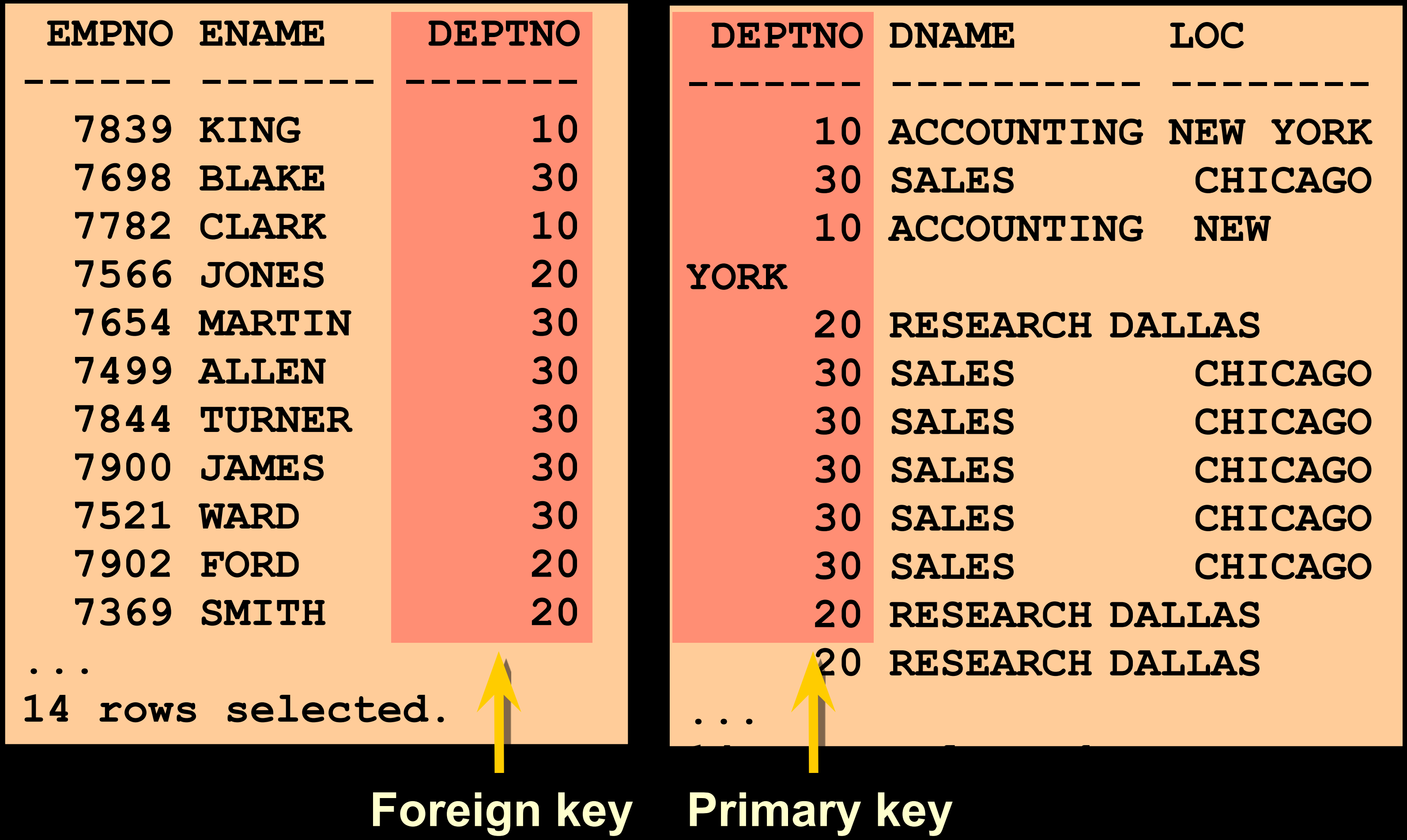
Self join



What Is an Equijoin?

EMP

DEPT



```
 EMPNO ENAME      DEPTNO
------- -------- --------
   7839 KING           10
   7698 BLAKE          30
   7782 CLARK          10
   7566 JONES          20
   7654 MARTIN         30
   7499 ALLEN          30
   7844 TURNER         30
   7900 JAMES          30
   7521 WARD           30
   7902 FORD           20
   7369 SMITH          20
...
14 rows selected.
```

```
  DEPTNO DNAME       LOC
-------- ---------- ---------
      10 ACCOUNTING NEW YORK
      30 SALES        CHICAGO
      10 ACCOUNTING   NEW
YORK
      20 RESEARCH DALLAS
      30 SALES        CHICAGO
      30 SALES        CHICAGO
      30 SALES        CHICAGO
      30 SALES        CHICAGO
      30 SALES        CHICAGO
      20 RESEARCH DALLAS
      20 RESEARCH DALLAS
...
```

Foreign key

Primary key

Retrieving Records with Equijoins

```
SQL> SELECT   emp.empno, emp.ename, emp.deptno,
  2     dept.deptno, dept.loc
  3  FROM     emp, dept
  4  WHERE    emp.deptno=dept.deptno;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------- ------ ------ ---------
 7839 KING           10     10 NEW YORK
 7698 BLAKE          30     30 CHICAGO
 7782 CLARK          10     10 NEW YORK
 7566 JONES          20     20 DALLAS
...
14 rows selected.
```

Qualifying Ambiguous Column Names

- – Use table prefixes to qualify column names that are in multiple tables.
- – Improve performance by using table prefixes.
- – Distinguish columns that have identical names but reside in different tables by using column aliases.

Additional Search Conditions Using the AND Operator

EMP

```
 EMPNO ENAME     DEPTNO
------ -------- -------
  7839 KING          10
  7698 BLAKE         30
  7782 CLARK         10
  7566 JONES         20
  7654 MARTIN        30
  7499 ALLEN         30
  7844 TURNER        30
  7900 JAMES         30
  7521 WARD          30
  7902 FORD          20
  7369 SMITH         20
...
14 rows selected.
```

DEPT

```
DEPTNO DNAME             LOC
------ --------- ---------
    10 ACCOUNTING    NEW
YORK
    30 SALES        CHICAGO
    10 ACCOUNTING    NEW
YORK
    20 RESEARCH   DALLAS
    30 SALES        CHICAGO
    30 SALES        CHICAGO
    30 SALES        CHICAGO
    30 SALES        CHICAGO
    30 SALES        CHICAGO
    20 RESEARCH   DALLAS
    20 RESEARCH   DALLAS
```

Using Table Aliases

- Simplify queries by using table aliases.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
  2    dept.deptno, dept.loc
  3  FROM   emp, dept
  4  WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e, dept d
  4  WHERE  e.deptno=d.deptno;
```

Joining More Than Two Tables

CUSTOMER

```
NAME         CUSTID
-----------   -------
JOCKSPORTS            100
TKB SPORT SHOP
101
VOLLYRITE       102
JUST TENNIS           103
K+T SPORTS            105
SHAPE UP        106
WOMENS SPORTS           107
... ...
```

ORD

```
  CUSTID    ORDID
-------  -------
     101      610
     102      611
     104      612
     106      601
     102      602
     106
     106
...
21 rows s
```

ITEM

```
 ORDID  ITEMID
------ -------
   610        3
   611        1
   612        1
   601        1
   602        1
...
64 rows selected.
```

Non-Equijoins

EMP

```
 EMPNO ENAME       SAL
------ -------- ------
  7839 KING       5000
  7698 BLAKE      2850
  7782 CLARK      2450
  7566 JONES      2975
  7654 MARTIN     1250
  7499 ALLEN      1600
  7844 TURNER     1500
  7900 JAMES       950
...
14 rows selected.
```

SALGRADE

```
GRADE  LOSAL   HISAL
----- ----- ------
1        700   1200
2       1201   1400
3       1401   2000
4   2001    3000
5       3001   9999
```

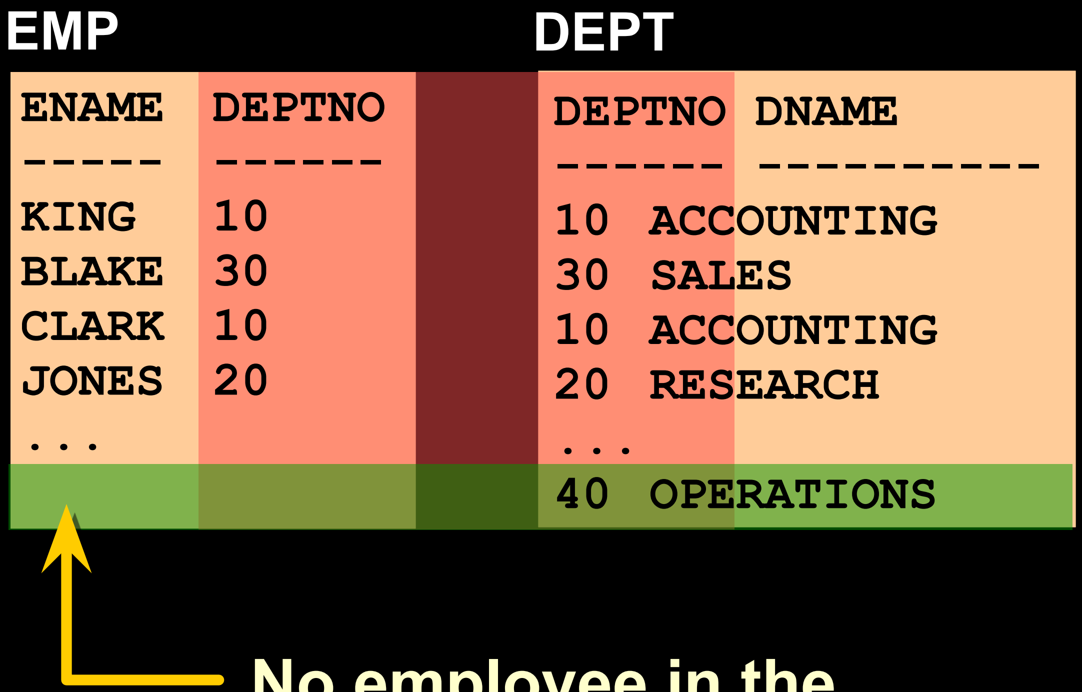
"salary in the EMP table is between low salary and high salary in the SALGRADE table"

Retrieving Records with Non-Equijoins

```
SQL>   SELECT    e.ename, e.sal, s.grade
   2   FROM   emp e, salgrade s
   3   WHERE  e.sal
   4   BETWEEN    s.losal AND s.hisal;
```

```
ENAME             SAL      GRADE
---------- ---------- ----------
JAMES             950          1
SMITH             800          1
ADAMS            1100          1
...
14 rows selected.
```

Outer Joins



EMP

```
ENAME   DEPTNO
-----   -------
KING    10
BLAKE   30
CLARK   10
JONES   20
...
```

DEPT

```
DEPTNO  DNAME
------- ----------
10  ACCOUNTING
30  SALES
10  ACCOUNTING
20  RESEARCH
...
40  OPERATIONS
```

No employee in the OPERATIONS department

Outer Joins

- – You use an outer join to also see rows that do not usually meet the join condition.
- – Outer join operator is the plus sign (+).

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

Using Outer Joins

```
SQL> SELECT   e.ename, d.deptno, d.dname
  2  FROM emp e, dept d
  3  WHERE    e.deptno(+) = d.deptno
  4  ORDER BY e.deptno;
```

```
ENAME         DEPTNO DNAME
---------- --------- -------------
KING              10 ACCOUNTING
CLARK             10 ACCOUNTING
...
                  40 OPERATIONS
15 rows selected.
```

Left Outer Join

- SELECT d.department_id, e.last_name
FROM
departments d
LEFT OUTER JOIN
employees e
ON
d.department_id = e.department_id

Self Joins

EMP (WORKER)

```
EMPNO  ENAME   MGR
-----  ------ ----
 7839  KING
 7698  BLAKE  7839
 7782  CLARK  7839
 7566  JONES  7839
 7654  MARTIN 7698
 7499  ALLEN  7698
```

EMP (MANAGER)

```
EMPNO  ENAME
-----  ---------

 7839  KING
 7839  KING
 7839  KING
 7698  BLAKE
 7698  BLAKE
```

“MGR in the WORKER table is equal to EMPNO in the MANAGER table”

Joining a Table to Itself

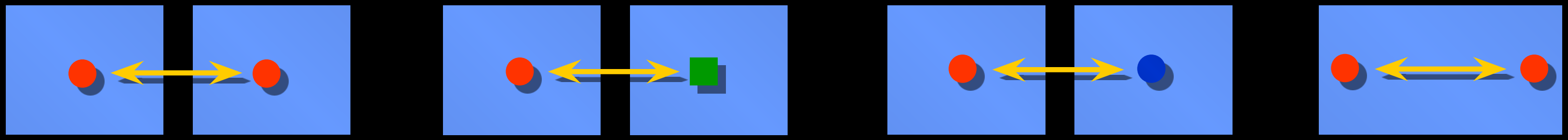
```
SQL> SELECT worker.ename||' works for '||manager.ename
  2  FROM     emp worker, emp manager
  3  WHERE    worker.mgr = manager.empno;
```

```
WORKER.ENAME||'WORKSFOR'||MANAG
-------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

Summary

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column1 = table2.column2;
```

Equijoin   Non-equijoin   Outer join   Self join



Practice Overview

- Joining tables using an equijoin
- Performing outer and self joins
- Adding conditions