Unit 7:

- KEYS AND MORE JOINS

Using SQL Server

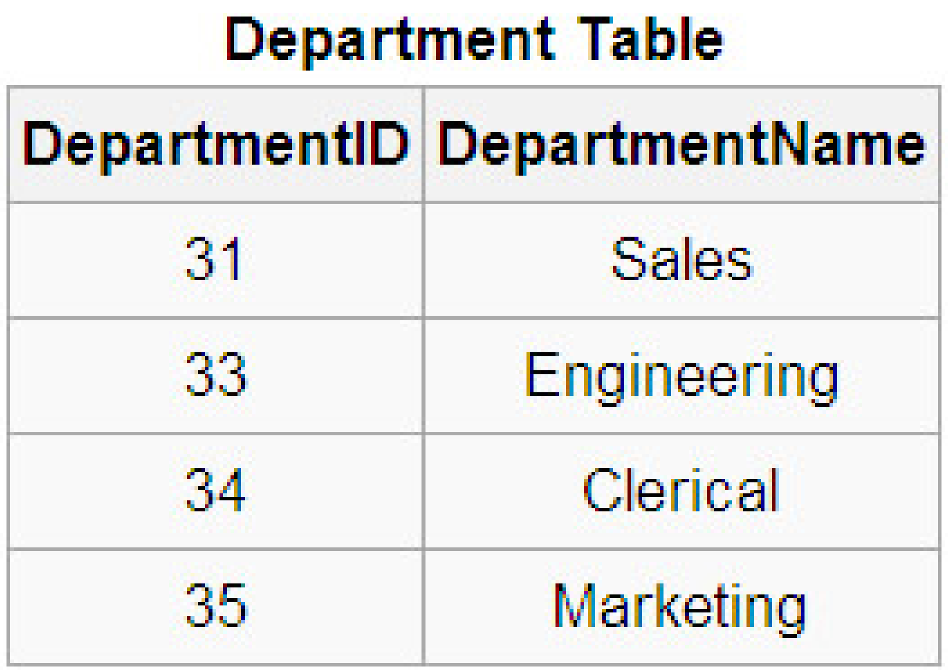
“As a student, I would like to understand how to the different types of join work so I can understand when to use them. “

•Topics of Discussion

Unit 7: Keys and More Joins

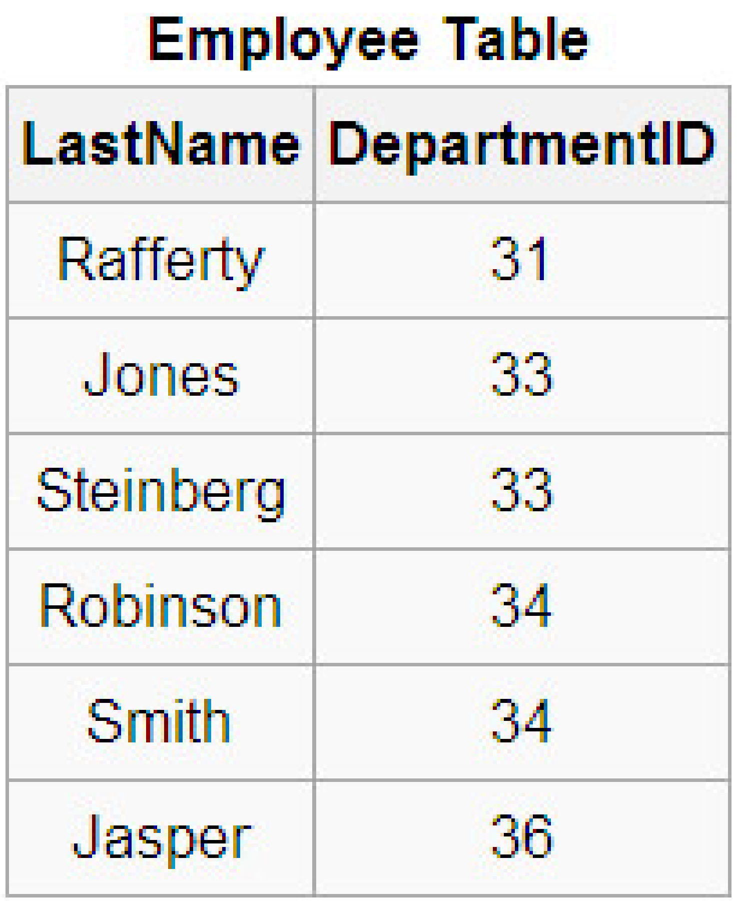
- Create SQL queries that join multiple tables
- Create nested SQL queries
- Combine query results using set operators
- Create and use database views

Outer Join Example



Department Table

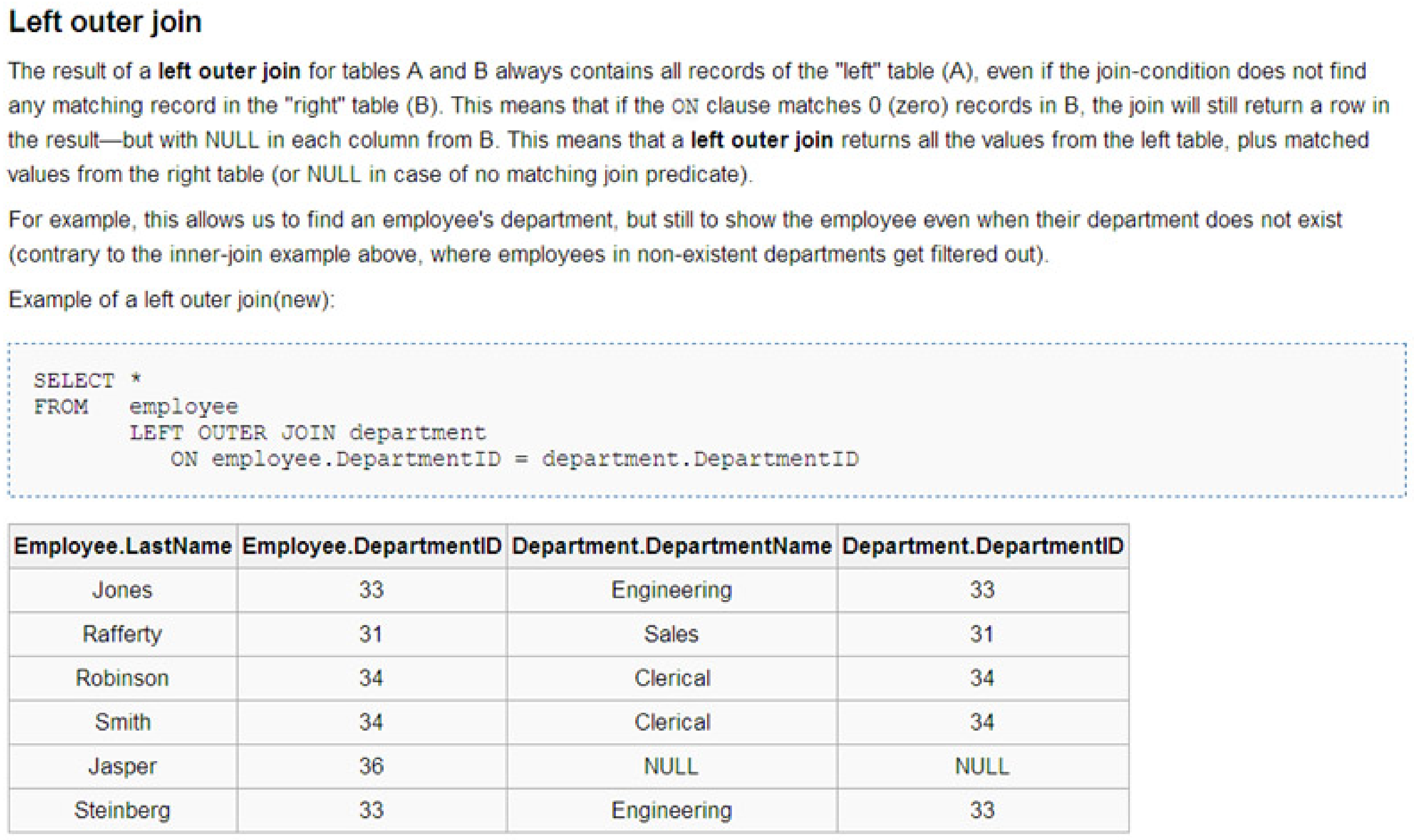
DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing



Employee Table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
Jasper	36

Left Outer Join



Left outer join

The result of a **left outer join** for tables A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B). This means that if the `ON` clause matches 0 (zero) records in B, the join will still return a row in the result—but with NULL in each column from B. This means that a **left outer join** returns all the values from the left table, plus matched values from the right table (or NULL in case of no matching join predicate).

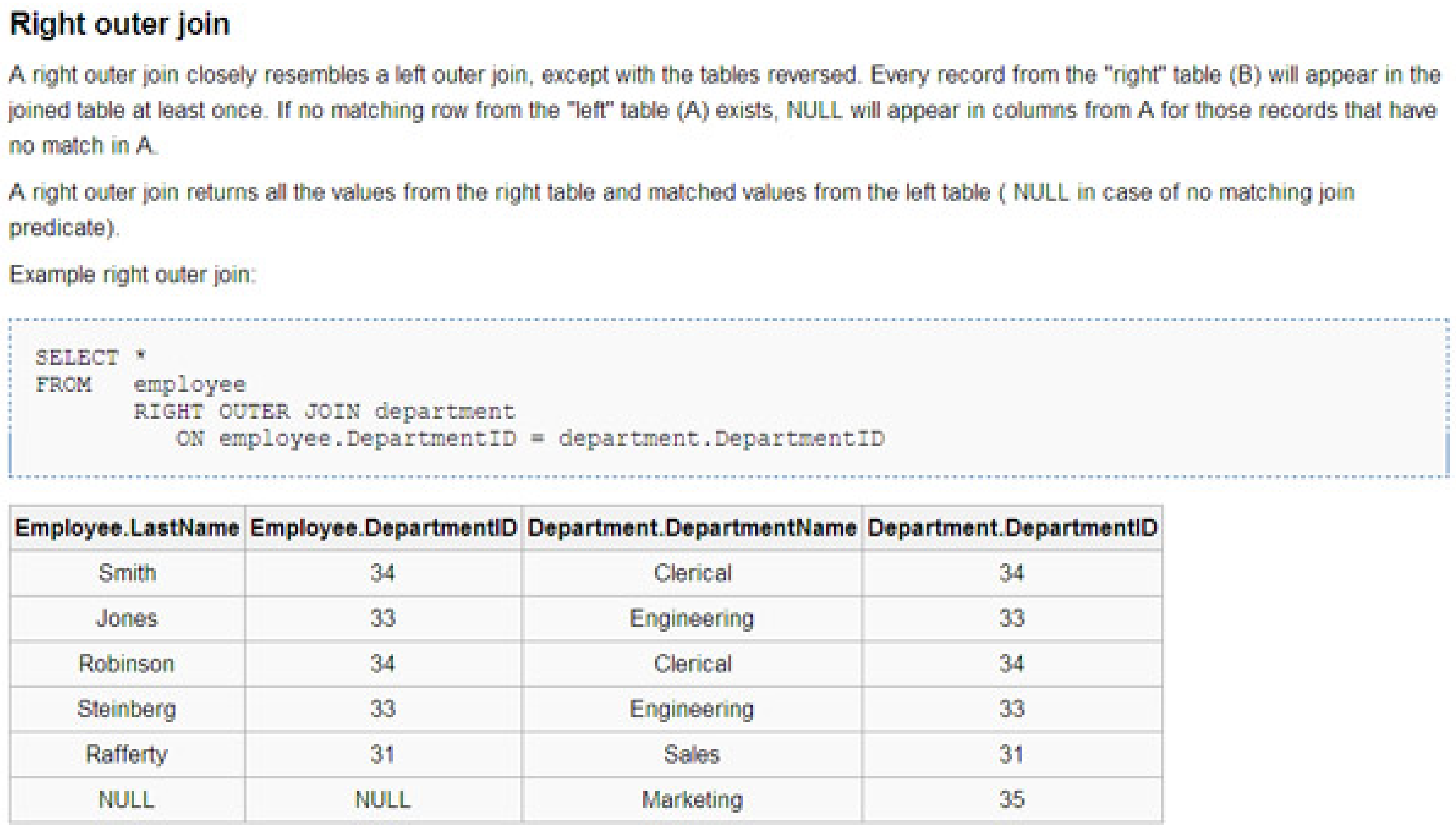
For example, this allows us to find an employee's department, but still to show the employee even when their department does not exist (contrary to the inner-join example above, where employees in non-existent departments get filtered out).

Example of a left outer join(new):

```
SELECT *
FROM   employee
       LEFT OUTER JOIN department
          ON employee.DepartmentID = department.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
Jasper	36	NULL	NULL
Steinberg	33	Engineering	33

Right Outer Join



Right outer join

A right outer join closely resembles a left outer join, except with the tables reversed. Every record from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in A.

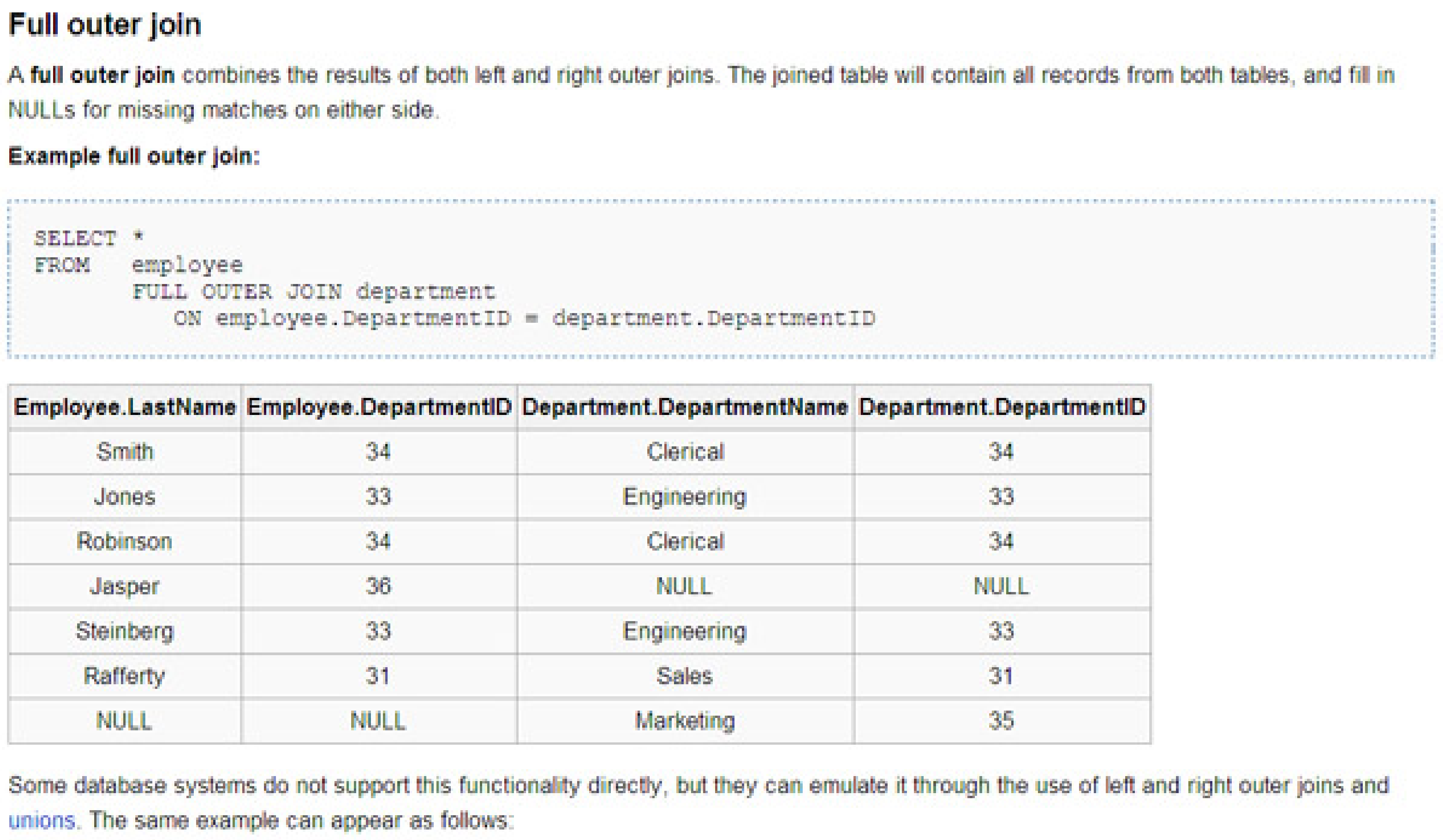
A right outer join returns all the values from the right table and matched values from the left table (NULL in case of no matching join predicate).

Example right outer join:

```
SELECT *
FROM   employee
       RIGHT OUTER JOIN department
          ON employee.DepartmentID = department.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35

Full Outer Join



Full outer join

A **full outer join** combines the results of both left and right outer joins. The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

Example full outer join:

```
SELECT *
FROM   employee
       FULL OUTER JOIN department
          ON employee.DepartmentID = department.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Jasper	36	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35

Some database systems do not support this functionality directly, but they can emulate it through the use of left and right outer joins and unions. The same example can appear as follows:

*** Source: Wikipedia - http://en.wikipedia.org/wiki/Join_(SQL)

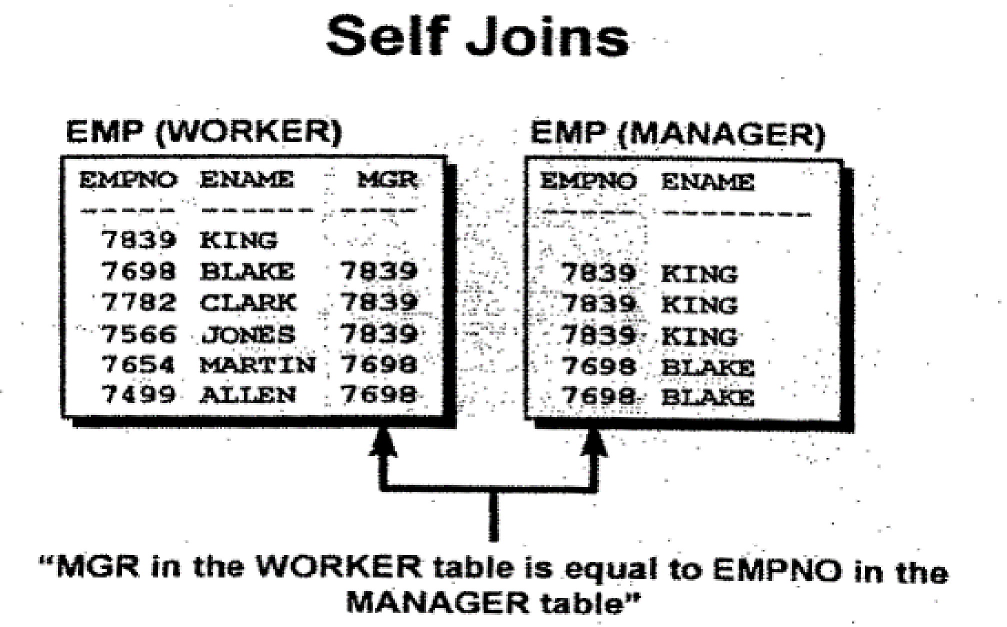
Self-joins

- Query that joins table to itself
- Must create table alias
  - Alternate name assigned to table in query's FROM clause
  - Syntax
    - `FROM` *`table1 alias1`*`, …`

    - `Employee Table`
      - `EmployeeNumber, EmployeeName, Manager`

      - `If you wanted to see the name of each employees manager, you would need to first find out the employee then find out the manager.  Look at handouts.`

Self-joins



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

Sometimes you need to join a table to itself. To find the name of each employee's manager, you need to join the EMP table to itself, or perform a self join. For example, to find the name of Blake's manager, you need to:

- Find Blake in the EMP table by looking at the ENAME column.
- Find the manager number for Blake by looking at the MGR column. Blake's manager number is 7839.
- Find the name of the manager with EMPNO 7839 by looking at the ENAME column. King's employee number is 7839, so King is Blake's manager.

In this process, you look in the table twice. The first time you look in the table to find Blake in the ENAME column and MGR value of 7839. The second time you look in the EMPNO column to find 7839 and the ENAME column to find King.

Results

QUERY

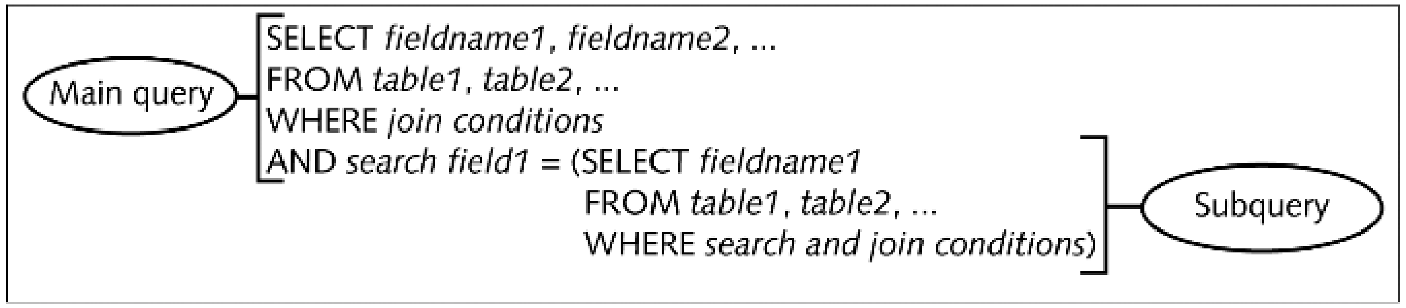
```
SELECT worker.ename ||'works for  '|| manager.ename
FROM emp worker, emp manager
WHERE worker.mgr = manager.empno
```

```
WORKER.ENAME || ' WORKS FOR' ||MANAGER.ENAME
-------------------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
```

Creating Nested Queries

- Nested query
    - Consists of main query and one or more subqueries
    - Main query
        - First query that appears in SELECT command
    - Subquery
        - Retrieves values that main query’s search condition must match

Creating Nested Queries with Subqueries that Return a Single Value



```
SELECT fieldname1, fieldname2, ...
FROM table1, table2, ...
WHERE join conditions
AND search field1 = (SELECT fieldname1
                     FROM table1, table2, ...
                     WHERE search and join conditions)
```

Figure 3-53 Syntax for nested query

UNION and UNION ALL

- UNION set operator
  - Joins output of two unrelated queries into single output result
  - Syntax
    - `query1 UNION query2;`
- UNION ALL operator
  - Same as UNION but includes duplicate rows

Minus/Intersect

EXCEPT returns distinct rows from the left input query that aren’t output by the right input query.

INTERSECT returns distinct rows that are output by both the left and right input queries operator.

The basic rules for combining the result sets of two queries that use EXCEPT or INTERSECT are the following:

The number and the order of the columns must be the same in all queries.

The data types must be compatible.

Syntax

```
        SELECT * FROM table_A
        UNION ALL  (could use Except, Intersect, Union)
        SELECT * FROM table_B;
```

Questions, comments, discussion