**27TH JANUARY**

**JOINED RELATIONS**

* + 1. **Types of JOIN**

Cross join

inner join

equi-join

left outer join

right outer join

full outer join

* + 1. **Join Conditions**

natural

on < predicate>

using (A1, A1, . . ., An)

* The **join condition** defines **which tuples** in the **two relations match** and what **attributes are present** in the result of the join.
* The **join type** defines **how tuples** in each **relation that do not match any tuple** in the other relation (based on the join condition) are treated.
* The **use of a join condition** is mandatory for **outer joins**, but is **optional** for **inner joins**.
* SQL left outer join is also known as SQL left join.
* Suppose, we want to join two tables: A and B. SQL **left outer join returns all rows in the left table (A) and all the matching rows found in the right table (B)**.
* It means the result of the SQL left join always contains the rows in the left table.
* A **right outer join returns all the values from the right table and matched values from the left table**
* The meaning of the join condition **natural**, in terms of **which tuples from the two relations match,** is straightforward.
* Example of **combining the natural join condition with the right outer join type**:

loan **natural right outer join** borrower

|  |  |
| --- | --- |
| Employee table | |
| **LastName** | **DepartmentID** |
| Rafferty | 31 |
| Jones | 33 |
| Heisenberg | 33 |
| Robinson | 34 |
| Smith | 34 |
| Williams | NULL |

|  |  |
| --- | --- |
| Department table | |
| **DepartmentID** | **DepartmentName** |
| 31 | Sales |
| 33 | Engineering |
| 34 | Clerical |
| 35 | Marketing |

* + 1. **CROSS JOIN**
* CROSS JOIN returns the [Cartesian product](https://en.wikipedia.org/wiki/Cartesian_product) of rows from tables in the join. In other words, it will produce rows which combine each row from the first table with each row from the second table.

**SELECT** \*

**FROM** employee **CROSS JOIN** department;

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee.LastName** | **Employee.DepartmentID** | **Department.DepartmentName** | **Department.DepartmentID** |
| Rafferty | 31 | Sales | 31 |
| Jones | 33 | Sales | 31 |
| Heisenberg | 33 | Sales | 31 |
| Smith | 34 | Sales | 31 |
| Robinson | 34 | Sales | 31 |
| Williams | NULL | Sales | 31 |
| Rafferty | 31 | Engineering | 33 |
| Jones | 33 | Engineering | 33 |
| Heisenberg | 33 | Engineering | 33 |
| Smith | 34 | Engineering | 33 |
| Robinson | 34 | Engineering | 33 |
| Williams | NULL | Engineering | 33 |
| Rafferty | 31 | Clerical | 34 |
| Jones | 33 | Clerical | 34 |
| Heisenberg | 33 | Clerical | 34 |
| Smith | 34 | Clerical | 34 |
| Robinson | 34 | Clerical | 34 |
| Williams | NULL | Clerical | 34 |
| Rafferty | 31 | Marketing | 35 |
| Jones | 33 | Marketing | 35 |
| Heisenberg | 33 | Marketing | 35 |
| Smith | 34 | Marketing | 35 |
| Robinson | 34 | Marketing | 35 |
| Williams | NULL | Marketing | 35 |

* The results of a cross join can be filtered by using a [WHERE](https://en.wikipedia.org/wiki/Where_(SQL)) clause which may then produce the equivalent of an inner join.
  + 1. **INNER JOIN**

1. EXPLICIT JOIN NOTATION

**SELECT** employee.LastName, employee.DepartmentID, department.DepartmentName

**FROM** employee

**INNER JOIN** department **ON**

employee.DepartmentID = department.DepartmentID;

|  |  |  |
| --- | --- | --- |
| **Employee.LastName** | **Employee.DepartmentID** | **Department.DepartmentName** |
| Robinson | 34 | Clerical |
| Jones | 33 | Engineering |
| Smith | 34 | Clerical |
| Heisenberg | 33 | Engineering |
| Rafferty | 31 | Sales |

1. IMPLICIT JOIN NOTATION

**SELECT** employee.LastName, employee.DepartmentID, department.DepartmentName

**FROM** employee, department

**WHERE** employee.DepartmentID = department.DepartmentID;

* SAME OUTPUT
  + 1. **EQUI-JOIN**
* An **equi-join** is a specific type of comparator-based join, that uses only [equality](https://en.wikipedia.org/wiki/Equality_(mathematics)) comparisons in the join-predicate.

**SELECT** \*

**FROM** employee **JOIN** department

**ON** employee.DepartmentID = department.DepartmentID;

**OR**

**SELECT** \*

**FROM** employee, department

**WHERE** employee.DepartmentID = department.DepartmentID;

* If columns in an equi-join have the same name, then equi-join uses the **USING** construct

**SELECT** \*

**FROM** employee **INNER JOIN** department **USING** (DepartmentID);

* + 1. **NATURAL JOIN**
* The natural join is a special case of equi-join.
* The result of the natural join is the set of all combinations of [tuples](https://en.wikipedia.org/wiki/Tuples) in *R* and *S* that are equal on their common attribute names.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | *Employee* | | | | **Name** | **EmpId** | **DeptName** | | Harry | 3415 | Finance | | Sally | 2241 | Sales | | George | 3401 | Finance | | Harriet | 2202 | Sales | | |  |  | | --- | --- | | *Dept* | | | **DeptName** | **Manager** | | Finance | George | | Sales | Harriet | | Production | Charles | |

**Name EmpId DeptName Manager**

|  |  |  |  |
| --- | --- | --- | --- |
| Harry | 3415 | Finance | George |
| Sally | 2241 | Sales | Harriet |
| George | 3401 | Finance | George |
| Harriet | 2202 | Sales | Harriet |

* The above sample query for inner joins can be expressed as a natural join in the following way:
* **SELECT** \* **FROM** employee **NATURAL JOIN** department;

As with the explicit USING clause, only one DepartmentID column occurs in the joined table, with no qualifier:

|  |  |  |
| --- | --- | --- |
| **DepartmentID** | **Employee.LastName** | **Department.DepartmentName** |
| 34 | Smith | Clerical |
| 33 | Jones | Engineering |
| 34 | Robinson | Clerical |
| 33 | Heisenberg | Engineering |
| 31 | Rafferty | Sales |

## **OUTER JOIN**

* Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table's rows are retained.

1. **LEFT OUTER JOIN**

* The result of a **left outer join** (or simply **left join**) for tables A and B always contains all rows of the "left" table (A), even if the join-condition does not find any matching row in the "right" table (B).

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 |
| *Williams* | NULL | NULL | NULL |
| Heisenberg | 33 | Engineering | 33 |

### **RIGHT OUTER JOIN**

* A **right outer join** (or **right join**) closely resembles a left outer join, except with the treatment of the tables reversed.

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 |
| Heisenberg | 33 | Engineering | 33 |
| Rafferty | 31 | Sales | 31 |
| NULL | NULL | *Marketing* | *35* |

1. **FULL OUTER JOIN**

* A **full outer join** combines the effect of applying both left and right outer joins.

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 |
| *Williams* | NULL | NULL | NULL |
| Heisenberg | 33 | Engineering | 33 |
| Rafferty | 31 | Sales | 31 |
| NULL | NULL | *Marketing* | *35* |

* Some database systems do not support the full outer join functionality directly, but they can emulate it through the use of an inner join and **UNION ALL** selects of the "single table rows" from left and right tables respectively.