

5. SELECT with table JOIN

- **JOIN of tables is mainly used to get data from more than one table**

5.1 Cartesian Join – when in the FROM clause more than one table is given, then the result set of query will contain all rows from the Cartesian product of the tables:

SELECT ... FROM table1, table2, ...

Product

No	Product	Type_id	Quantity
1	mouse	IT	100
2	plotter	IT	120
3	fridge	AGD	15
4	tape	IT	1000
5	mixer	AGD	30

Type

Type_id	Description
IT	IT devices
AGD	household goods
Other	other products

SELECT Product, Quantity, Description

FROM Product, Type

Product	Quantity	Description
mouse	100	IT devices
mouse	100	household goods
mouse	100	other products
plotter	120	IT devices
plotter	120	household goods
plotter	120	other products
fridge	15	IT devices
fridge	15	household goods
fridge	15	other products
tape	1000	IT devices
tape	1000	household goods
tape	1000	other products
mixer	30	IT devices
mixer	30	household goods
mixer	30	other products

5.2 **INNER JOIN** – the join condition is given, in the **WHERE** clause

- **SELECT ... FROM table1, table 2**

WHERE table1.kol_a = table2.kol_b

or in the **JOIN** clause

- **SELECT ... FROM table1 JOIN table2**

ON (table1.kol_a = table2.kol_b)

- the join condition usually uses the relations between tables (defined as the **referential constraints** using primary and foreign keys)

SELECT full_name, department
FROM employee e, department d
WHERE **e.dept_no=d.dept_no**

SELECT full_name, department
FROM employee e JOIN department d
ON (e.dept_no=d.dept_no)

//join condition

SELECT Product, Quantity, Description
FROM Product p JOIN Type t
ON (p.Type_id=t.Type_id)

Product	Quantity	Description
mouse	100	IT devices
plotter	120	IT devices
fridge	15	household goods
tape	1000	IT devices
mixer	30	household goods

- **INNER JOIN** can only select those rows (from joined tables) which satisfy the join condition, i.e., that have their counterpart in the other table; they do not take into account the **NULL** values

- we can join more than two tables (more join conditions should be given then, for example, if one has to join three tables, then two join conditions are necessary for an inner join)

```
SELECT    full_name, department, job_title
FROM      (employee e JOIN department d ON (e.dept_no=d.dept_no))
          JOIN job j ON (e.job_code=j.job_code)
```

- it is possible to join a table with itself; this is called **self-join**; in such a case aliases for table name are necessary to distinguish between the copies of the table

```
SELECT e1.full_name, e1.job_country
FROM employee e1 JOIN employee e2 ON(e1.job_country=e2.job_country)
WHERE e2.emp_no=2
```

- mostly, joins use columns that are primary or unique keys;
- the join condition can be more involved and based on more than one column;

```
SELECT e.full_name, e.job_country, j.job_title, j.max_salary
FROM employee e JOIN job j
      ON(e.job_country=j.job_country AND e.job_code=j.job_code AND e.job_grade=j.job_grade)
WHERE e.full_name='Nelson, Robert'
```

- it is possible to make joins based on other criteria than equality, for example

```
SELECT e1.full_name, e1.salary
FROM employee e1 JOIN employee e2 ON(e1.salary>e2.salary)
WHERE e2.emp_no=2
```

5.3 OUTER JOIN – we use the clause **LEFT JOIN, RIGHT JOIN**

- We use outer joins in case when in one of joined tables there are rows having no counterparts in the other, but we want to include such rows in the join; this may also happened if there are rows having NULL values in columns mentioned in the join condition; in order to include such rows in the join, we should use outer join.
- The existing rows of one of our tables are then joined with NULL rows from the other
- **LEFT JOIN** (**RIGHT JOIN**) – **left** (**right**) outer join is used when we want to include in the join **all rows from the left hand side table** (**right hand side table**, resp.), even if these rows do not have counterparts in the other table

For example, the next query display only the departments having at least one employer:

```
SELECT d.department, d.budget, e.full_name, e.salary  
FROM department d JOIN employee e  
ON(d.dept_no=e.dept_no)
```

If one want to include in the result all departments,
an outer query should be used:

```
SELECT d.department, d.budget, e.full_name, e.salary  
FROM department d LEFT JOIN employee e ON (d.dept_no=e.dept_no)  
(left table) (right table)
```

This query selects all types and join them with products

```
SELECT Product, Quantity, Description  
FROM Product p RIGHT JOIN Type t  
ON (p.Type_id=t.Type_id)
```

Product	Quantity	Description
mouse	100	IT devices
plotter	120	IT devices
fridge	15	household goods
tape	1000	IT devices
mixer	30	household goods
NULL	NULL	other products

