# Practical class 3

**Content**

* Short reharshing of previous clases
* SELECT from many tables
* JOIN
* OUTER JOIN
* UNION

**Database CREATE script**

create table jobs (

ID char(4) primary key,

title nvarchar(50),

does\_deliveries bit not null default 0

)

create table employees (

ID int identity(1,1) not null primary key,

name nvarchar(50),

job\_id char(4) null foreign key references jobs

)

create table employment (

ID int identity(1,1) primary key,

employee\_id int not null references employees,

job\_id char(4) not null references jobs,

salary smallmoney not null,

start\_date date not null,

end\_date date null

)

create table products (

ID int identity(100,1) primary key,

description nvarchar(max),

list\_price smallmoney not null,

cost smallmoney null,

needs\_cooking bit default 1

)

create table order\_status (

ID char(4) primary key,

title nvarchar(50)

)

create table orders (

ID int identity(10001,1) primary key,

order\_amount smallmoney not null,

order\_time datetime2 default getUTCdate(),

status char(4) not null references order\_status default 'COOK',

courier int null references employees,

delivery\_address nvarchar(max),

notes nvarchar(max)

)

create table order\_items (

ID int identity (1,1) primary key,

order\_id int not null foreign key references orders,

product\_id int not null foreign key references products

)

Create and analyse database diagramm..

Database contains order\_items that is one of possible implementation of many-to-many relationship. Other possible implementation – table with complex primary key.

create table order\_item\_alternate (

order\_id int not null foreign key references orders,

product\_id int not null foreign key references products,

number int,

constraint PK\_order\_item\_alternate primary key (order\_id, product\_id)

)

**Database content**

INSERT INTO jobs VALUES (‘TEST’, ‘Good person’, 1)

INSERT INTO jobs VALUES ('BOSS', 'Manager', 0)

INSERT INTO jobs VALUES ('COOK', 'Cooks', 0)

INSERT INTO jobs VALUES ('DRVR', 'Supplier', 1)

INSERT INTO employees (name, job\_id) VALUES ('Lisa Apple', NULL)

INSERT INTO employment (employee\_id, job\_id, salary, start\_date, end\_date)

VALUES (1, 'BOSS', 10, '2020-01-01', '2020-12-31')

INSERT INTO employees (name, job\_ID) VALUES ('Anna Gold', 'BOSS')

INSERT INTO employment (employee\_id, job\_id, salary, start\_date, end\_date)

VALUES (2, 'COOK', 5, '2020-01-01', '2020-12-31')

INSERT INTO employment (employee\_id, job\_id, salary, start\_date)

VALUES (2, 'BOSS', 10, '2021-01-01')

INSERT INTO employees (name, job\_ID) VALUES ('John Smith', 'COOK')

INSERT INTO employment (employee\_id, job\_id, salary, start\_date)

VALUES (3, 'COOK', 5, '2021-01-01')

INSERT INTO employees (name, job\_ID) VALUES ('Peter Johnson', 'DRVR')

INSERT INTO employment (employee\_id, job\_id, salary, start\_date)

VALUES (4, 'DRVR', 5, '2020-06-01')

INSERT INTO employees (name, job\_ID) VALUES ('William Black', 'DRVR')

INSERT INTO employment (employee\_id, job\_id, salary, start\_date)

VALUES (5, 'DRVR', 4, '2020-12-01')

INSERT INTO employees (name, job\_ID) VALUES ('Charles River', 'DRVR')

INSERT INTO employment (employee\_id, job\_id, salary, start\_date)

VALUES (6, 'DRVR', 4, '2021-03-01')

select \* from employees

select \* from employment

where end\_date is null

order by start\_date

INSERT INTO products (description, list\_price) VALUES

('Meat pizza', 5)

INSERT INTO products (description, list\_price) VALUES

('Pineapple pizza with tuna', 6)

INSERT INTO products (description, list\_price) VALUES

(‘Chicken pizza', 10)

INSERT INTO products (description, list\_price, cost, needs\_cooking) VALUES

('Soft drink', 1, 0.5, 0)

select \* from products

INSERT INTO order\_status VALUES ('COOK', 'Cooking')

INSERT INTO order\_status VALUES ('MOVE', 'Delivering')

INSERT INTO order\_status VALUES ('DONE', 'Delivered')

INSERT INTO order\_status VALUES ('FAIL', 'Failed')

INSERT INTO orders (order\_amount, delivery\_address, courier)

VALUES (10, 'Apple street 1', 4)

INSERT INTO order\_items VALUES (10001, 102)

INSERT INTO orders (order\_amount, delivery\_address, courier)

VALUES (20, 'Peach street 2', 4)

INSERT INTO order\_items VALUES (10002, 100)

INSERT INTO order\_items VALUES (10002, 100)

INSERT INTO order\_items VALUES (10002, 100)

INSERT INTO order\_items VALUES (10002, 100)

INSERT INTO orders (order\_amount, delivery\_address, status, notes, courier)

VALUES (1, 'Long street 3', 'MOVE', 'I am hungry, please, bring fast', 6)

INSERT INTO order\_items VALUES (10003, 103)

INSERT INTO orders (order\_amount, delivery\_address, courier)

VALUES (15, 'Short street 4', 4)

INSERT INTO order\_items VALUES (10004, 100)

INSERT INTO order\_items VALUES (10004, 102)

INSERT INTO order\_items VALUES (10004, 103)

INSERT INTO orders (order\_amount, notes, status)

VALUES (3, 'Yesterday’s pizza with discount without delivery', 'DONE')

INSERT INTO order\_items VALUES (10005, 101)

# Data select from multiple tables

**SELECT *[columna list]*  
FROM *[tables list]***

**WHERE *[rows rules]***

SELECT \* FROM orders, order\_status

Result is Cartesian product – all possible combinations.

|  |  |  |  |
| --- | --- | --- | --- |
|  | COOK | MOVE | DONE |
| Order1 | Order1 + COOK | Order1 + MOVE | Order1 + DONE |
| Order2 | Order2 + COOK | Order2 + MOVE | Order2 + DONE |
| Order3 | Order3 + COOK | Order3 + MOVE | Order3 + DONE |

SELECT \* FROM orders, order\_status

WHERE orders.status = order\_status.ID

Only combinations that satisfy WHERE criteria.

|  |  |  |  |
| --- | --- | --- | --- |
|  | COOK | MOVE | DONE |
| Order1 | Order1 + COOK | Order1 + MOVE | Order1 + DONE |
| Order2 | Order2 + COOK | Order2 + MOVE | Order2 + DONE |
| Order3 | Order3 + COOK | Order3 + MOVE | Order3 + DONE |

SELECT \* FROM orders, order\_status

WHERE status = ID

If the same column name is in more than one table, table name should be included.

**JOIN syntax**

**… FROM *table1* JOIN *table2* ON*****[rule]***

SELECT \*

FROM orders JOIN order\_items ON orders.ID = order\_items.order\_ID

SELECT \*

FROM orders

JOIN order\_items ON orders.ID = order\_items.order\_ID

JOIN products ON products.ID = order\_items.product\_ID

Any logical rule is allowed

SELECT \* FROM orders JOIN order\_items ON 2+2=4

**Meaningfull selects**

SELECT orders.ID, orders.order\_time, description

FROM orders

JOIN order\_items ON orders.ID = order\_items.order\_ID

JOIN products ON products.ID = order\_items.product\_ID

where orders.status = 'COOK'

order by order\_time

What does this statement do?

SELECT orders.ID, order\_amount, name as Driver\_name, delivery\_address

FROM orders

JOIN employees on orders.courier = employees.id

where orders.status = 'MOVE'

And this?

**Join table with itself**

select name, new\_job.job\_id, new\_job.start\_date, old\_job.start\_date

from employment new\_job join employment old\_job

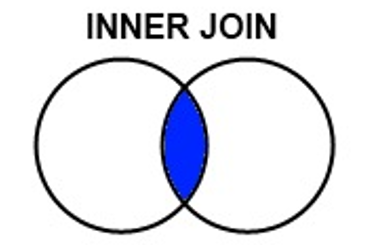
on new\_job.employee\_id = old\_job.employee\_id

and new\_job.start\_date > old\_job.end\_date

join employees on new\_job.employee\_id = employees.id

Selects where tha same table is used multiple times are not often. Usaly such construction is used in case of comparing multiple records from the same table (for example, hierachy or historical data).

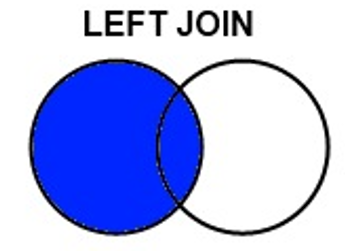
**What to do, if there is record in one table without appropriate value in other table?**



SELECT orders.ID, order\_amount, name as Driver\_name

FROM orders JOIN employees on orders.courier = employees.id

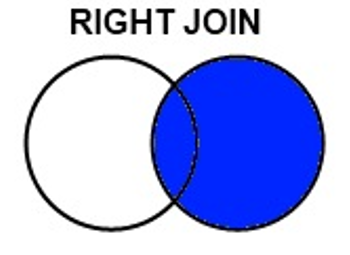
This select contains not all data!



SELECT orders.ID, order\_amount, name as Driver\_name

FROM orders LEFT JOIN employees on orders.courier = employees.id

This select takes all rows from table on left side – all orders

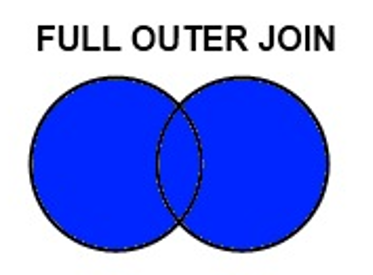


SELECT orders.ID, order\_amount, name as Driver\_name

FROM orders RIGHT JOIN employees on orders.courier = employees.id

where employees.job\_id = 'DRVR'

This select takes all rows from table on left side – all drivers



SELECT orders.ID, order\_amount, name as Driver\_name

FROM orders full OUTER JOIN employees on orders.courier = employees.id

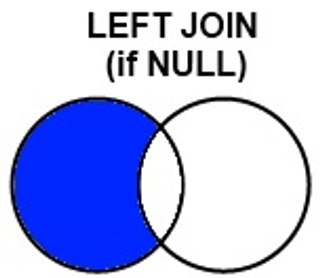
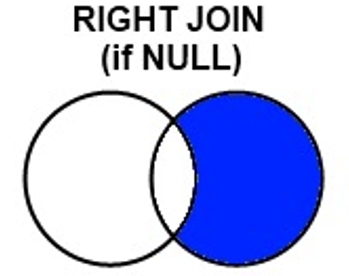
Every record from both tables.

SELECT orders.ID, order\_amount, name as Driver\_name

FROM orders full OUTER JOIN employees on orders.courier = employees.id

WHERE employees.job\_id = 'DRVR'

Be carefully with WHERE clauses – this job\_id filter “takes away” all orders without driver



SELECT orders.ID, order\_amount, name as Driver\_name

FROM orders RIGHT JOIN employees on orders.courier = employees.id

where employees.job\_id = 'DRVR' and orders.id is null

Drivers without orders

SELECT distinct name as Driver\_name

FROM orders JOIN employees on orders.courier = employees.id

where employees.job\_id = 'DRVR'

Drivers with at least one order

**UNION**

**SELECT *[…]*  UNION SELECT *[…]***

Unions rows from multiple sources

* Column number should be the same
* Column types should be the same

UNION or UNION DISTINCT – rows withour duplicates

UNION ALL – duplicates included

SELECT name from employees

UNION

SELECT description from products

SELECT name from employees

UNION

SELECT list\_price from products

SELECT name from employees

UNION

SELECT description, list\_price from products

SELECT name, null from employees

UNION

SELECT description, list\_price from products

**Excercises**

Select list of ordered products and addressses where they should be delivered

Show list with ordered products, that are delivered (DONE)

Show list with ordered products that (needs\_cooking) – if ordered multiple times, then should include multiple times (rows) in result

Show list of employees employed at this moment with their jobs and salaries

Show list with all expenses included (in single list) – employees salaries and price of delivered products