Practical class 5

# Practical class uses data from Word file “Practical class Nr 3 2022”.

# Index

Index creates additional data structure that is used to effectively search data.

SELECT \* FROM orders WHERE ID = 10010 – look at primary key index  
SELECT \* FROM orders WHERE courier = 5 – search through all data

CREATE INDEX arbitrary\_name ON orders(courier)

Huge impact on speed. If data is searched, filtered or joined without index than server browses through all data. Browse through all data takes time O(n), with index server can search data in time O(log(n)). Compare time increase in case of 1 000 rows vs 1 000 000 rows. On the other hand index is additional data structure that takes some place on disk and in should be updated in every INSERT, UPDATE and DELETE operation.

Typical use:

* Columns with foreign key;
* Columns often searched by value;
* To optimize “important” query

Index can be used to require unique values (like primary key).

CREATE UNIQUE INDEX arbitrary)index\_name ON products(description)

"Clustered" index (possible only one per table) actualy store table rows in required order. Primary key by default creates clustered index.

CREATE CLUSTERED INDEX index\_name ON orders(order\_time)

# Data aggregation

Aggregation functions are used to get aggregated value of some data. For example, get summary value of all orders.

SELECT SUM(order\_amount)

FROM orders

SQL has following aggregation functions:

AVG – average value  
SUM – summary value  
MIN – minimal value   
MAX – maximal value  
COUNT – count

Aggregate functions ignore NULL values!

Specific syntax to get row count, for example, select order count:

SELECT COUNT(\*)

FROM orders

Next query selects count of orders that has address (address is not null)

SELECT COUNT(delivery\_address)

FROM orders

DISTINCT is used to select count of different addresses:

SELECT COUNT(DISTINCT delivery\_address)

FROM orders

Previous select will ignore rows where address is null and count only different addresses. Different DISTINCT can be combined.

SELECT COUNT(DISTINCT delivery\_address), COUNT(DISTINCT courier), sum(order\_amount)

FROM orders

AVG and SUM work only on numeric data. MIN, MAX, COUNT work on every data type.

SELECT can have number of aggregate functions. Aggregate functions can be combined with other features of SELECT, for example, SELECT can getnumber of deliveries, sum of deliveries and average time of delivery for each employee.

SELECT COUNT(\*), SUM(order\_amount), AVG(delivery\_minutes)

FROM orders WHERE courier=4

If column data type is integer, then AVG also will be integer and result will be rounded. If precise value is necessary, then column inside AVG can be multiplied by 1.0. This is true for Microsoft SQL server, Oracle calculate precise value by default.

SELECT AVG(delivery\_minutes\*1.0)

FROM orders WHERE courier=4

Any expression can be written inside aggregate function.

# Aggregate functions and JOIN

Aggregate functions aggregate data and speed is affected by increase of row count!

SELECT SUM(order\_amount)

FROM orders

JOIN order\_items on order\_items.order\_id = orders.id

# Result grouping

Data can be selected and grouped on single SELECT, for example, select number of orders for each address:

SELECT delivery\_address, count(\*)

FROM orders

GROUP BY delivery\_address

Data can be grouped by more than one field. Than in one group will be all data with the same values for all grouped fields. By default, all rows are in one group for which aggregate function is calculated. If GROUP BY is used, than aggregated value is calculated for all rows with the same values for combination of GROUP BY fields.

SELECT delivery\_address, courier, count(\*)

FROM orders

GROUP BY delivery\_address, courier

ORDER BY can use aggregate functions:

SELECT delivery\_address, count(\*), sum(order\_amount)

FROM orders

GROUP BY delivery\_address

ORDER BY sum(order\_amount) DESC

Each column in SELECT should be in GROUP BY or in aggregate function.

SELECT delivery\_address, **courier,** count(\*)

FROM orders

GROUP BY delivery\_address

This select makes no sense. If we group by delivery\_address, but some address has more than one courier, than which courier value should be displayed. SQL interpreter will give an error.

# Filtering of aggregated results

Keywor HAVING is used to filter by aggregated values, for example, select addresses with more than 3 orders.

SELECT delivery\_address, count(\*), sum(order\_amount)

FROM orders

GROUP BY delivery\_address

HAVING count(\*) > 3

HAVING works even without HAVING statement, but that is very inefficiently. WHERE should be used instead of HAVING to filter by non-aggregated values.

WHERE works when data is readed from table (before grouping), HAVING works after aggregating. (If SELECT has no GROUP BY, than all data is selected (read from table), put in memory and then HAVING is used.)

# Usage of only some (not all) data

WHERE is used to select only some part of data, if only some part of data is used for grouping and aggregating.

Count number of “slow” orders for each address:

SELECT delivery\_address, count(\*)

FROM orders

WHERE delivery\_minutes >= 40

GROUP BY delivery\_address

# Self-check excercises

* Which courier delivered most times?
* Which was financially best month ?
* Which couriers deliver pizzas slowly?
* How much money we earn by pizzas?
* How many pizzas are not delivered?
* Which products are delivered most?
* Which customers order the most beer?
* How many pizzas has each courier delivered? Count pizzas exactly, not orders or drinks.
* How much has been paid (Table Products field cost) for all delivered products?
* Show all deliveries with profit - order\_amount minus product costs

# Answers for first five excercises

-- Which courier delivered most times?

select employees.id, name, count(\*) piegādes from orders

join employees on employees.id = courier

where status = 'DONE'

group by employees.id, name

order by count(\*) desc

-- Which was financially best month?

select cast( year(order\_time) as nvarchar) + ' ' + cast (month(order\_time) as nvarchar) mēnesis,

sum(order\_amount) summa

from orders

group by year(order\_time), month(order\_time)

order by sum(order\_amount) desc

-- Which couriers deliver pizzas slowly?

select name, count(\*) [lēnās piegādes], max(delivery\_minutes) [vislēnākā piegāde] from orders

join employees on courier = employees.id

where status = 'DONE' and delivery\_minutes > 30

group by name

having count(\*) > 1

-- How much money we earn by pizzas?

select sum(order\_amount) from orders where status = 'DONE'

-- How many pizzas are not delivered?

select count(\*) from orders where status <> 'DONE'