VSB – Technical University of Ostrava
Faculty of Electrical Engineering
Department of Computer Science
Database Research Group

Introduction to Database Systems

Tutorial 4

Ing. Petr Lukáš
petr.lukas@vsb.cz
EA440
Ostrava, 2017

- **What is the purpose of the following operators?**
  1. **IN**
  2. **EXISTS**
  3. **ALL**
  4. **ANY**

- **What is the type of result of IN and EXISTS ?**

- **What is the purpose of the following operators?**

1.	**IN**	Tests whether a value falls into a set
2.	**EXISTS**	Tests whether a subquery returns a non-empty result.
3.	**ALL**	Tests, whether a comparison is valid for all values returned by a subquery.
4.	**ANY**	Tests, whether a comparison is valid for at least one value returned by a subquery.

- **What is the type of result of IN and EXISTS ?**

  Boolean value, i.e., IN and EXISTS are always used in a boolean expression (e.g., in the WHERE clause).

SELECT Clauses

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

SELECT Clauses

SELECT	*list of columns to be output*
FROM	*input tables*
WHERE	*restricting ondition*
GROUP BY	*aggregation grouping*
HAVING	*group condition*
ORDER BY	*sorting of the result*

Not all clauses are mandatory, but their order is given strictly.

Content of the Tutorial

- **Joining tables**
- **Aggregation functions**
- **Subqueries**

Joining tables

Joining tables

Student

login	f_name	univerzity
bla005	Adam	1
whi007	Bob	1
gra065	John	2

Univerzity

id	name
1	VSB – TUO
2	University of Ostrava

Select all names of students and the universities they study.

```
SELECT f_name, name
FROM Student, University
WHERE Student.university = University.id
```

f_name	name
Adam	VSB – TUO
Bob	VSB – TUO
John	University of Ostrava

Example 1 – inner join

Student

login	f_name	univerzity
bla005	Adam	1
whi007	Bob	1
gra065	John	2

Univerzity

id	name
1	VSB – TUO
2	University of Ostrava

Select all names of students and the universities they study.

```
SELECT f_name, name
FROM Student, University
WHERE
  Student.university =
  University.id
```

=

```
SELECT f_name, name
FROM
  Student
  JOIN University ON
  Student.university = University.id
```

f_name	name
Adam	VSB – TUO
Bob	VSB – TUO
John	University of Ostrava

Example 2 – outer join

Student

login	f_name	univerzity
bla005	Adam	1
whi007	Bob	1
gra065	John	2

Univerzity

id	name
1	VSB – TUO
2	University of Ostrava

Select names of all universities and fist names of their students.

```
SELECT name, f_name
FROM University JOIN Student ON University.id = Student.university
```

name	f_name
VSB – TUO	Adam
VSB – TUO	Bob
University of Ostrava	John

- **Is the result correct?** – Yes, but …

Example 2 – outer join

Student

login	f_name	univerzity
bla005	Adam	1
whi007	Bob	1
gra065	John	2

Univerzity

id	name
1	VSB – TUO
2	University of Ostrava
3	CTU in Prague

Select names of all universities and fist names of their students.

```
SELECT name, f_name
FROM University LEFT JOIN Student ON University.id = Student.university
```

name	f_name
VSB – TUO	Adam
VSB – TUO	Bob
University of Ostrava	John
CTU in Prague	**NULL**

NULL Value

- Represents an empty value.
- If we want to test, whether a value is **NULL**, we need to use a special operator **IS NULL**.
- Any other comparison with **NULL** (< > != =) returns false.

Select all persons without an address.

```
~~SELECT *~~
~~FROM Person~~
~~WHERE address = NULL~~
```

```
SELECT *
FROM Person
WHERE address IS NULL
```

Join types

INNER JOIN

Equivalent to using of more tables after FROM and connecting the tables in WHERE.

OUTER JOIN

We have left or right outer join. Works the same as inner join, but retains all records from the left or right joined table, even there are no matching records in the other table.

Aggregation functions

Aggregation functions

Example 3 – minimum

Products

product_id	name	manufacturer	price
1	Acer TravelMate P253-E	Acer	10 490 CZK
2	HP 650	HP	8 949 CZK
3	HP ProBook 4540s	HP	11 990 CZK
4	Acer Aspire V7-581G-53334G52akk	Acer	19 990 CZK
5	Apple MacBook Air 13"	Apple	33 836 CZK

⬇

Select the price of the cheapest laptop.

```
SELECT MIN(price) AS [min_price]
FROM Products
```

⬇

min_price
8 990 CZK

Example 4 – minimum

Products

product_id	name	manufacturer	price
1	Acer TravelMate P253-E	Acer	10 490 CZK
2	HP 650	HP	8 949 CZK
3	HP ProBook 4540s	HP	11 990 CZK
4	Acer Aspire V7-581G-53334G52akk	Acer	19 990 CZK
5	Apple MacBook Air 13"	Apple	33 836 CZK

⬇

Select prices of the cheapest laptops for each manufacturer.

```
SELECT manufacturer, MIN(price) AS [min_price]
FROM Products
GROUP BY manufacturer
```

⬇

manufacturer	min_price
Acer	10 490 CZK
HP	8 949 CZK
Apple	33 836 CZK

Using of GROUP BY

Any attribute in SELECT that is not contained in any aggregation function must be in GROUP BY.

Example 5 – group condition

Products

product_id	name	manufacturer	price
1	Acer TravelMate P253-E	Acer	10 490 CZK
2	HP 650	HP	8 949 CZK
3	HP ProBook 4540s	HP	11 990 CZK
4	Acer Aspire V7-581G-53334G52akk	Acer	19 990 CZK
5	Apple MacBook Air 13"	Apple	33 836 CZK

⬇

Select manufacturers selling their cheapest laptop for more than 30 000 CZK.

```
SELECT manufacturer
FROM Products
GROUP BY manufacturer
HAVING MIN(price) > 30000
```

⬇

manufacturer
Apple

Aggregation functions

COUNT (*attr*)	Number of values
COUNT (DISTINCT *attr*)	Number of unique values
SUM (*attr*)	Sum of values
AVG (*attr*)	Arithmetic mean
MIN (*attr*)	Minimum value
MAX (*attr*)	Maximum value

Aggregation functions skip NULL values.

Subqueries

Example 6

Products

product_id	name	manufacturer	price
1	Acer TravelMate P253-E	Acer	10 490 CZK
2	HP 650	HP	8 949 CZK
3	HP ProBook 4540s	HP	11 990 CZK
4	Acer Aspire V7-581G-53334G52akk	Acer	19 990 CZK
5	Apple MacBook Air 13"	Apple	33 836 CZK

⬇

Select the maximum average price of laptops of different maufacturers.

Select the averate price of each manufacturer.

```
SELECT manufacturer, AVG(price) AS [average]
FROM Products
GROUP BY manufacturer
```

➡

manufacturer	average
Acer	15 240 CZK
HP	10 469 CZK
Apple	33 836 CZK

Example 6

Products

product_id	name	manufacturer	price
1	Acer TravelMate P253-E	Acer	10 490 CZK
2	HP 650	HP	8 949 CZK
3	HP ProBook 4540s	HP	11 990 CZK
4	Acer Aspire V7-581G-53334G52akk	Acer	19 990 CZK
5	Apple MacBook Air 13"	Apple	33 836 CZK

⬇

Select the maximum average price of laptops of different maufacturers.

Select the averate price of each manufacturer.

```
SELECT manufacturer, AVG(price) AS [average]
FROM Products
GROUP BY manufacturer
```

➡

manufacturer	average
Acer	15 240 CZK
HP	10 469 CZK
Apple	33 836 CZK

Select the maximum average price of laptops of different maufacturers.

```
SELECT MAX(average) AS [maximum]
FROM 
```

➡

maximum
33 836 CZK

Example 6

Products

product_id	name	manufacturer	price
1	Acer TravelMate P253-E	Acer	10 490 CZK
2	HP 650	HP	8 949 CZK
3	HP ProBook 4540s	HP	11 990 CZK
4	Acer Aspire V7-581G-53334G52akk	Acer	19 990 CZK
5	Apple MacBook Air 13"	Apple	33 836 CZK

⬇

Select the maximum average price of laptops of different maufacturers.

```
SELECT MAX(average) AS [maximum]
FROM
(
  SELECT manufacturer, AVG(price) AS [average]
  FROM Products
  GROUP BY manufacturer
) averages
```

⬇

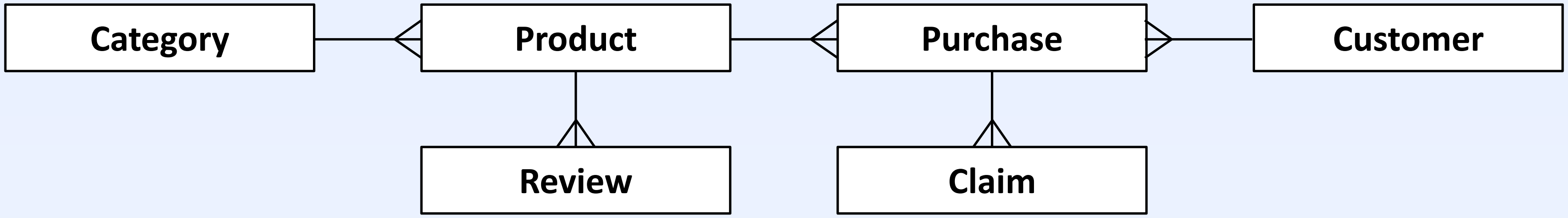
maximum
33 836 Kč

Subqueries

- We can subquery after FROM instead of a regular table. The result of the subquery acts just as it is a regular table.

- The subquery has to be **in brackets** and it has to have an **alias**. All columns of the subquery have to be **named**.

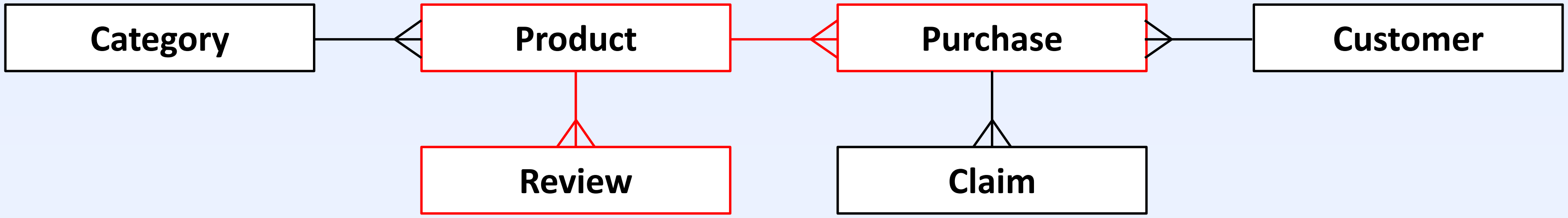
Example 7



Select all products and the number of their reviews and purchases.

```
SELECT
  Product.Name, COUNT(Review.id_review), COUNT(Purchase.id_purchase)
FROM
  Product
  LEFT JOIN Review ON Review.id_product = Product.id_product
  LEFT JOIN Purchase ON Purchase.id_product = Product.id_product
GROUP BY
  Product.Name
```

Example 7

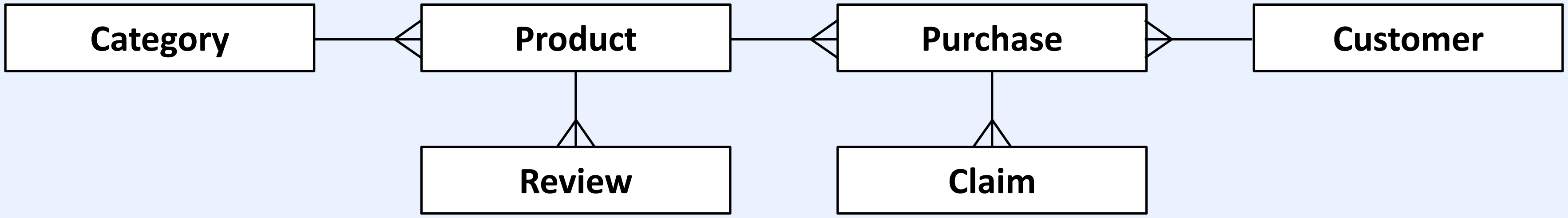


Select all products and the number of their reviews and purchases.

```
~~SELECT
   Product.Name, COUNT(Review.id_review), COUNT(Purchase.id_purchase)
FROM
   Product
   LEFT JOIN Review ON Review.id_product = Product.id_product
   LEFT JOIN Purchase ON Purchase.id_product = Product.id_product
GROUP BY
   Product.Name~~
```

One product ***p*** can have ***n*** purchases and ***m*** reviews. **The purchases and reviews are independent**, so for ***p*** the query works with **cartesian product** of the corresponding purchases and reviews.

Example 7



Select all products and the number of their reviews and purchases.

```
SELECT
  Product.Name,
  (
    SELECT COUNT(Review.id_review)
    FROM Review
    WHERE Review.id_product = Product.id_product
  ) AS [n_of_reviews],
  (
    SELECT COUNT(Purchase.id_purchase)
    FROM Purchase
    WHERE Purchase.id_product = Product.id_product
  ) AS [n_of_purchases]
FROM Product
```

Subquery for number of reviews

Subquery for number of purchases

Subqueries

- Subquery returning a single value can be used anywhere to represent a single value, i.e. in SELECT, WHERE …

- Useful if we aggregate values on multiple independent tables.

- Subqueries can be understood as functions.

Tutorial

www.dbedu.cs.vsb.cz

- LDAP **login** and **password**
- ***English Courses** -> **IDS***