

# Databázové systémy a SQL

A decorative background graphic consisting of a series of vertical, stylized bar charts or box plots. Each bar has a central circle and is connected to a horizontal line. The bars are arranged in a slightly curved line across the top right of the slide. The background is a light gray gradient.

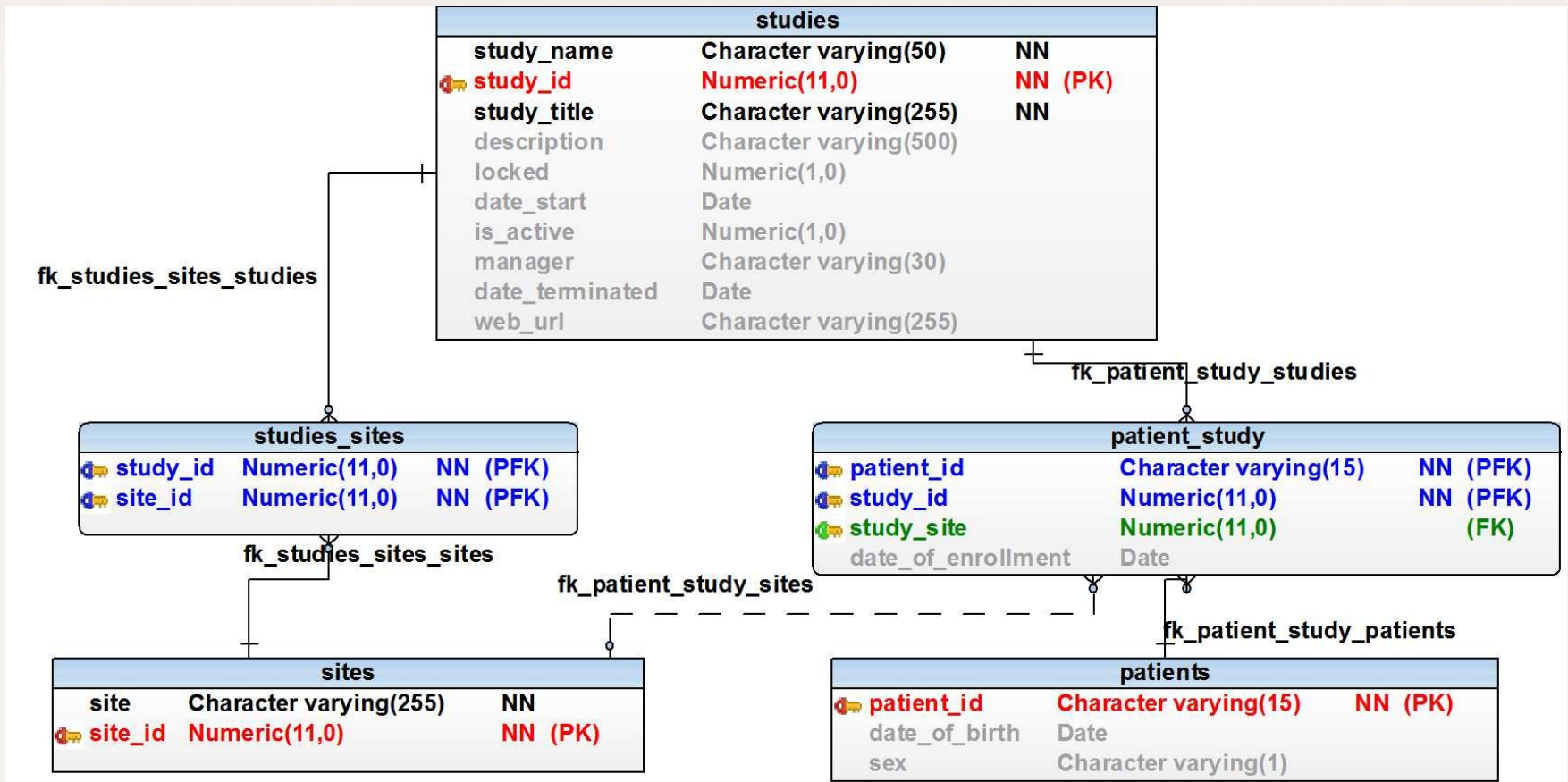
---

## Lekce 5 - Vnořené dotazy

Daniel Klimeš

# Another data model

patients – studies m-n => „mezitabulka“ PATIENT\_STUDY  
 studies – sites m-n => „mezitabulka“ STUDIES\_SITES



## Zanořené dotazy / Subqueries

- uzavřené v kulatých závorkách () / *enclosed in brackets*
- poddotazem je myšlen příkaz SELECT

**SELECT**      column  
**FROM**        table  
**WHERE**        condition

- místo názvu sloupce

- místo názvu tabulky

- v sekci WHERE

GROUP BY  
HAVING  
ORDER BY

Vnořený dotaz na pozici sloupce musí vrátit právě jeden řádek a právě jeden sloupec!

*This type a subquery must return just one row and one column*

```
SELECT COUNT(student_uco),
      (SELECT COUNT (*) FROM student)
```

```
FROM vyuka;
```

```
SELECT COUNT(patient_id),
      (SELECT COUNT (*) FROM patients)
```

```
FROM patient_study;
```

## CVIČENÍ:

Napište dotaz, který vrátí seznam všech studentů,  
počet jejich registrovaných předmětů  
a kolik je to procent ze všech dostupných předmětů.

*Create a query, which return a list of all students with number their registered subjects and compute percent from all subjects (= all rows from table předmět)*

## CVIČENÍ:

Napište dotaz, který vrátí seznam všech studentů,  
počet jejich registrovaných předmětů  
a kolik je to procent ze všech dostupných předmětů

```
SELECT s.uco, COUNT(v.predmet_id),
       ROUND(100.0 * (COUNT(v.predmet_id)) /
             (SELECT COUNT(*) FROM predmet) )
FROM student s LEFT JOIN vyuka v
ON s.uco=v.student_uco
GROUP BY s.uco;
```

Poddotaz na pozici FROM nahrazuje tabulku.  
V PostgreSQL musí být poddotaz na pozici tabulky **VŽDY** pojmenován!

*Subquery instead of a name of the table must have a **name/acronym***

```
SELECT COUNT(*) FROM (
    SELECT study_id, COUNT(patient_id)
    FROM patient_study GROUP BY study_id
) sub
```

## CVIČENÍ:

Napište dotaz, který vrátí seznam studentů, kteří jsou registrováni do více než jednoho předmětu.

*Create a query, which select list of students registered in more than 1 subject*



## CVIČENÍ:

### Varianta 1 (variant with subquery)

```
SELECT * FROM (
    SELECT s.firstname, s.lastname, s.uco, COUNT(v.predmet_id) pocet
    FROM student s JOIN vyuka v ON s.uco=v.student_uco
    GROUP BY s.firstname, s.lastname, s.uco) sub
WHERE pocet>1;
```

### Varianta 2 (variant with HAVING)

```
SELECT s.firstname, s.lastname, s.uco, COUNT(v.predmet_id)
FROM student s JOIN vyuka v ON s.uco=v.student_uco
GROUP BY s.firstname, s.lastname, s.uco
HAVING COUNT(v.predmet_id)>1
ORDER BY s.firstname;
```

## Varianty:

### A)

- WHERE sloupec = (SELECT sloupec FROM...  
zanořený dotaz musí vrátit **právě 1 řádek a 1 sloupec**  
*subquery must return just 1 row and 1 column*

### B)

- WHERE sloupec = **ANY** (SELECT sloupec FROM...
- WHERE sloupec **IN** (SELECT sloupec FROM ...
- WHERE sloupec > **ALL** (SELECT sloupec FROM ...  
zanořený dotaz musí vrátit **1 sloupec a libovolný počet řádků**  
*subquery must return just 1 column and 0 – n rows*

## Varianty:

C)

- WHERE **EXISTS** (SELECT \* FROM....
- WHERE **NOT EXISTS** (SELECT \* FROM...  
 zanořený dotaz může vrátet **libovolný počet řádků i sloupců**  
*subquery can return 0 – n rows, columns are irrelevant*

Zanořené dotazy se obvykle propojují s nadřazeným dotazem pomocí podmínky v sekci WHERE

*Subqueries usually contain a parent-related condition after WHERE*

## Varianty:

### A)

- WHERE sloupec = (SELECT sloupec FROM...

Example:

```
SELECT * FROM patients
```

```
WHERE date_of_birth = (SELECT MAX(date_of_birth) FROM patients);
```

## Varianty:

### B)

- WHERE sloupec = **ANY** (SELECT sloupec FROM...
- WHERE sloupec **IN** (SELECT sloupec FROM ...
- WHERE sloupec > **ALL** (SELECT sloupec FROM ...

Example:

```
SELECT * FROM student
```

```
WHERE uco = ANY (SELECT student_uco FROM vyuka WHERE  
predmet_id=10);
```

```
SELECT * FROM student
```

```
WHERE uco IN (SELECT student_uco FROM vyuka WHERE  
predmet_id=10);
```

## Varianty:

### C)

- WHERE **EXISTS** (SELECT \* FROM....
- WHERE **NOT EXISTS** (SELECT \* FROM...

### Example:

```
SELECT * FROM student s
```

```
WHERE EXISTS (SELECT * FROM vyuka v
```

```
WHERE predmet_id=10
```

```
AND s.uco=v.student_uco);
```

## Nejmladší student/ youngest student:

```
SELECT * FROM student WHERE birthdate = (
    SELECT MAX(birthdate) FROM student);
```

```
SELECT * FROM student WHERE
    birthdate >= ALL (
    SELECT birthdate FROM student);
```

```
SELECT * FROM student tab1 WHERE NOT EXISTS (
    SELECT * FROM student tab2
    WHERE tab2. birthdate > tab1. birthdate);
```

**Pozor na NULL hodnoty !**  
***Beware of NULLs in data***

**Task: Přepište na nejstarší studenty / *rewrite queries to oldest students***

## Task:

Vypište seznam studentů, kteří nemají registrovaný žádný předmět.

*Select all students, who have no registered subject*



## CVIČENÍ:

Vypište seznam studentů, kteří nemají registrovaný žádný předmět.

```
SELECT * FROM student s
WHERE NOT EXISTS (
    SELECT * FROM vyuka v
    WHERE s.uco=v.student_uco
);
```

## CVIČENÍ

Najděte všechny učitele, kteří nevyučují žádný předmět.

*SELECT all teachers, who teach no subject*

Najděte všechny učitele, kteří nevyučují žádný předmět.

```
SELECT * FROM teacher u
WHERE NOT EXISTS (
    SELECT * FROM predmet p
    WHERE u.teacher_ucod=p.teacher_ucod);
```

Vypište všechny studenty, kteří mají zapsaný předmět Databáze v biomedicíně i Černou magii. (predmet\_id 1 a 10)

*Select all students, who have registered subjects predmet\_id = 1 and 10 - both*

Vypište všechny studenty, kteří mají zapsaný předmět Databáze v biomedicíně i Černou magii. (predmet\_id 1 a 10)

Varianta 1

```
SELECT * FROM student s
WHERE EXISTS (SELECT * FROM vyuka v
              WHERE s.uco=v.student_uco AND predmet_id=1)
INTERSECT
SELECT * FROM student s
WHERE EXISTS (SELECT * FROM vyuka v
              WHERE s.uco=v.student_uco AND predmet_id=10);
```

Vypište všechny studenty, kteří mají zapsaný předmět Databáze v biomedicíně i Černou magii. (predmet\_id 1 a 10)

Varianta 2)

```
SELECT * FROM student s
```

```
WHERE EXISTS (SELECT * FROM vyuka v
```

```
                WHERE s.uco=v.student_uco AND predmet_id=1)
```

```
AND EXISTS (SELECT * FROM vyuka v
```

```
                WHERE s.uco=v.student_uco AND predmet_id=10);
```

Vypište všechny studenty, kteří mají zapsaný předmět Databáze v biomedicíně (predmet\_id 1), ale nemají zapsanou Černou magii (predmet\_id 10).

*Select all students, who have registered subjects predmet\_id = 1  
but not predmet\_id = 10*



Vypište všechny studenty, kteří mají zapsaný předmět Databáze v biomedicíně (predmet\_id 1), ale nemají zapsanou Černou magii (predmet\_id 10).

Varianta 1)

```
SELECT * FROM student s
WHERE EXISTS (SELECT * FROM vyuka v
              WHERE s.uco=v.student_uco AND predmet_id=1)
INTERSECT
SELECT * FROM student s
WHERE NOT EXISTS (SELECT * FROM vyuka v
                 WHERE s.uco=v.student_uco AND predmet_id=10);
```

Vypište všechny studenty, kteří mají zapsaný předmět Databáze v biomedicíně (predmet\_id 1), ale nemají zapsanou Černou magii (predmet\_id 10).

Varianta 2)

```
SELECT * FROM student s
WHERE EXISTS (SELECT * FROM vyuka v
              WHERE s.uco=v.student_uco AND predmet_id=1)
AND NOT EXISTS (SELECT * FROM vyuka v
                WHERE s.uco=v.student_uco AND predmet_id=10);
```

Vypište všechna pracoviště, která v roce 2010 nezařadila do studie žádného pacienta.

*Select all sites, which had no enrolled patient in year 2010*

Vypište všechna pracoviště, která v roce 2010 nezařadila do studie žádného pacienta.

```
SELECT * FROM sites s
WHERE NOT EXISTS (
    SELECT * FROM patient_study ps
    WHERE EXTRACT(YEAR FROM date_of_enrollment)=2010
    AND ps.study_site=s.site_id
);
```

Vypište všechna pracoviště, která zařadila pacienta naposledy v roce 2010.

*Select all sites, which enrolled last patient in 2010*

Vypište všechna pracoviště, která zařadila pacienta naposledy v roce 2010.

```
1] SELECT * FROM sites s
WHERE EXISTS (
    SELECT * FROM patient_study ps
    WHERE EXTRACT(YEAR FROM date_of_enrollment)=2010
    AND ps.study_site=s.site_id
)
AND NOT EXISTS (
    SELECT * FROM patient_study ps
    WHERE EXTRACT(YEAR FROM date_of_enrollment)>2010
    AND ps.study_site=s.site_id
);
```

Vypište všechna pracoviště, která zařadila pacienta naposledy v roce 2010.

```
2] SELECT s.site, s.site_id,
        MAX(EXTRACT(YEAR FROM ps.date_of_enrollment)) rok
FROM sites s JOIN patient_study ps
ON s.site_id=ps.study_site
GROUP BY s.site, s.site_id
HAVING MAX(EXTRACT(YEAR FROM ps.date_of_enrollment))=2010;
```

Najděte předměty, kam se přihlásil alespoň jeden student (muž) a vypište celkový počet přihlášených studentů.

*Find and select all subjects with minimal one male as student and add column with all registered students to given subject*



Najděte předměty, kam se přihlásil alespoň jeden student (muž) a vypište celkový počet přihlášených studentů.

```
SELECT predmet_id, COUNT(*) FROM vyuka v
WHERE EXISTS (
    SELECT predmet_id FROM student s2 JOIN vyuka v2
    ON s2.uco = v2.student_uco
    WHERE s2.sex = 'žena' AND v.predmet_id=v2.predmet_id
)
GROUP BY predmet_id;
```

Zjistěte počet pacientů v jednotlivých studiích po pracovištích a dle pohlaví  
STUDY\_NAME, SITE, SEX, počet pacientů