MDA104: Tutorial 3
SQL

Vlastislav Dohnal

Online app to practice SQL

- **RelaX** - relational algebra calculator, by University of Innsbruck
  - Switch to SQL tab
  - https://dbis-uibk.github.io/relax/calc/gist/e562a4534294842027ba7f0ae3c38bd0

3

- The SQL SELECT statement can be written as:
  SELECT A_1 , …, A_k FROM r_1 , …, r_n WHERE *condition*

- Let's have relations
  course (*code*, *title*, *credits*, *type_of_completion*)
  enrollment (*učo*, *code*, *type_of_completion*)
  student (*učo*, *first_name*, *last_name*)
- In SQL, write queries that return:
  - Names of courses that have at least three credits;
  - Names of courses that students have enrolled in for credit;
  - Courses whose code begins with 'PV';
  - Courses that have the word 'English' in their title;
  - Students’ first and last names arranged alphabetically.

4

- Let's have relations

  course (*<u>code</u>*, *title*, *credits*)

  seminar (*<u>code</u>*, *<u>number</u>*, *capacity*)

- Write an expression in SQL that returns course codes that have a seminar.
  - Can the codes repeat as a result?

5

- Assume relation

product

<u>*code*</u>	*title*	*unit_quantity*	*price_wo_VAT*
LCM01	ACER LCD 19“	1	10 800
RAM23	DDR2 1024MB (2x512)	2	4 980

- Write an SQL query that returns product names and their price including 20% tax.

- Rename operator AS:
  SELECT *unit_quantity* **AS** *quantity* FROM *product*;

- Modify the example so that it returns two attributes named title and price_w_VAT.

- Consider the relation: *test* (*A*,*B*,*C*)
- *Renaming* relations *in SQL*:

  SELECT *A, B, C* FROM *test* **AS** *t* WHERE *t.A*=17;

  - Attributes can also be renamed:

    SELECT *nA, nB, nC* FROM *test* **AS** *t* (*nA*,*nB*,*nC*)
    WHERE *nA*=17;

- The result of the SELECT command is again a relation
  - i.e. SELECT can be nested in the FROM section:

```
SELECT p.title
FROM course AS p,
        (SELECT code FROM enrollment WHERE type_of_completion=‘z’) AS z
WHERE p.code=z.code;
```

7 Joining Relations

- Join operations
  - Variants: INNER JOIN, [LEFT | RIGHT | FULL] OUTER JOIN
    - SELECT … FROM *r1* NATURAL INNER JOIN *r2*
    - SELECT … FROM *r1* INNER JOIN *r2* ON *condition*
    - SELECT … FROM *r1* INNER JOIN *r2* USING (*list of attributes*)

- Let's have relations *course* (*code*, *title*, *credits*)
  seminar (*code*, *number*, *capacity*)

- Formulate SQL queries:
  - For each seminar group, write out its number, capacity and name of the corresponding course;
  - write out the pairs of course code and seminar group number.
    - The result must contain the codes of all courses that we have in database.

8 Aggregation

- Aggregation specified in GROUP BY

  SELECT G_1, …, G_n, $F_1(A_1)$, …, $F_m(A_m)$ FROM r GROUP BY G_1, …, G_n
  - The G_i and A_i attributes are from the relation schema *r*
  - F_i indicates an aggregate function, attributes G_i are optional
  - The relational schema of the result is: $(G_1,…,G_n, F_1,…,F_m)$

- Let's have relations *course* (*code*, *title*, *credits*)

  seminar (*code*, *number*, *capacity*)

- Zapište výrazy v SQL, jejichž výsledkem je:
  - celkový počet předmětů s kódem začínajícím ’MA’;
  - celková kapacita skupin předmětu ’PB154’;
  - kód předmětu a počet jeho seminárních skupin.
    - pro předměty mající alespoň jednu sem. skupinu
    - pro všechny předměty a uspořádejte sestupně podle počtu skupin

9 Aggregation (cont.)

- HAVING clause allows a condition with aggregation function
  - SELECT G_1, …, G_n, $F_1(A_1)$, …, $F_m(A_m)$ FROM r
    GROUP BY G_1, …, G_n HAVING $F_x(A_x)$ > 10

- Let's have relations *course* (*code*, *title, credits*)
  seminar (*code*, *number*, *capacity*)

- Write SQL expressions that return:
  - Pairs of course code and number of seminar groups for those courses that have a total capacity of their groups greater than 100;
  - Course codes that have less than two seminar groups.
    - i.e. also no seminar!

10

- Nested SELECT in WHERE clause:
  - Used with set operators
    - IN, NOT IN,   EXISTS,   > ANY (), = ANY (), ….

SELECT … FROM … WHERE *A* IN (SELECT *A* FROM …)

- Let's have relations

  course (*<u>code</u>*, *title*, *credits*, *type_of_completion*)

  seminar (*<u>code</u>*, *<u>number</u>*, *capacity*, *description*)

  enrollment (*<u>učo</u>*, *<u>code</u>*, *type_of_completion*)

  student (*<u>učo</u>*, *first_name*, *last_name*)

- Formulate an SQL query that selects:
  - course codes that no student is enrolled in;
  - courses with the following codes MA102, PB154, PV004;
  - names of courses with the most credits;
  - names of students who have registered in at least two courses.

11 – Hodnoty NULL

- Let's have relations

  course (*<u>code</u>*, *title*, *credits*, *type_of_completion*)

  seminar (*<u>code</u>*, *<u>number</u>*, *capacity*, *description*)

  enrollment (*<u>učo</u>*, *<u>code</u>*, *type_of_completion*)

  student (*<u>učo</u>*, *first_name*, *last_name*)

- Formulate SQL queries returning:
  - The numbers of seminar groups of the course PB154 that do not have description filled in;
  - Counts of courses for individual credit values;
  - number of enrolled students for individual types of completions of the course PB154.
    - What will be the result of this query for this table?

enrollment

<u>učo</u>	*<u>code</u>*	*type_of_completion*
10	PB154	zk
20	PB154	zk
30	PB154	NULL
40	PB154	z

12 Set operations

- EXCEPT, UNION, INTERSECT
  - Syntax: *r* UNION *s*
  - Variant with **ALL**, e.g., EXCEPT ALL, …

- Let's have relations

  course (*code*, *title*, *credits*, *type_of_completion*)

  seminar (*code*, *number*, *capacity*)

  enrollment (*učo*, *code*, *type_of_completion*)

  student (*učo*, *first_name*, *last_name*)
- Use set operations to formulate SQL queries that select:
  - course codes that no student is enrolled in;
  - učo of students who are enrolled in courses 'PB154' and 'MA102' at the same time.

13 Table modification

- Inserting INSERT INTO *r* (A_1, A_2, …) VALUES (v_1, v_2, …);
- Deleting DELETE FROM *r* WHERE p;
- Updating UPDATE *r* SET A_1=$expr_1$, … WHERE p;

- Let's have relations

  course (<u>*code*</u>, *title*, *credits*, *type_of_completion*)

  seminar (<u>*code*</u>, <u>*number*</u>, *capacity*)

  enrollment (<u>*učo*</u>, <u>*code*</u>, *type_of_completion*)

  student (<u>*učo*</u>, *first_name*, *last_name*)

- Formulate an SQL statement to:
  - insert a new course (*'IB009', 'Paralelní výpočty', 5*);
  - increase the number of credits by 1 for courses with a code beginning with 'PB';
  - delete all courses that do not have any seminar group and are not enrolled by any student.