Databázové systémy

Jazyk SQL - Príkaz SELECT

Ing. Ján Perháč, PhD.

Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Košice, Slovensko

Príkaz SELECT

Základná syntax

- Najdôležitejší príkaz v SQL.
- Získa (vyfiltruje) údaje z tabuľky.
  - Vybrané riadky (WHERE).
  - Vybrané stĺpce (SELECT).
- Syntax:

```
SELECT co
FROM odkial
WHERE co_nas_zaujima
ORDER BY podla_coho_triedit
DESC;
```

Príkaz SELECT

Príkaz SELECT - Príklad

```
SELECT * FROM student;

SELECT meno, priezvisko
FROM student;

SELECT meno, priezvisko
FROM student
ORDER BY priezvisko, meno;

SELECT meno, priezvisko
FROM student
WHERE meno = 'Janko';
```

Základné (logické) operátory a funkcie

Základné operátory

- Relačné:

$$=, <, >, <>, != , <=, >=$$

- Aritmetické:

$$+, -, *, /, \text{MOD(a,b)}$$

- Spojenie dvoch reťazcov (konkatenácia):

'string1' || 'string2'

- Test na hodnotu *NULL*:

IS [NOT] NULL

Výraz môže byť:

- V klauzule *WHERE* pri formulácii podmienky selekcie.
- V zozname stĺpcov projekcie príkazu *SELECT*.
- V klauzule *ORDER BY*.

Základné (logické) operátory a funkcie

Základné logické operátory

- **AND** - "a zároveň platí"
  meno='Janko' AND priezvisko='Hrasko'
- **OR** - "alebo"
  predmet='SQL' OR predmet='DBS'
  predmet IN ('SQL', 'DBS')
- **NOT** - negácia
  predmet NOT IN ('SQL', 'DBS')

Základné (logické) operátory a funkcie

Základné funkcie na reťazcoch

- *upper(string)* - prevod na veľké písmená.
- *lower(string)* - prevod na malé písmená.
- *substr(string, from [, count])* - podreťazec.
- *initcap(string)* - kapitálky.
- *replace(string text, from text, to text)* - náhrada reťazca.
- *length(string)* - dĺžka reťazca.
- *md5(string)* - vypočíta MD5 hash parametra.
- *ascii(string)* - ASCI kód prvého symbolu reťazca.
- *chr(int)* - ASCI kód na symbol.

Základné (logické) operátory a funkcie

Základné funkcie na reťazcoch - príklad

Príklad	**Výsledok**
upper('tom')	TOM
lower('TOM')	tom
substr('alphabet', 3, 2)	ph
initcap('hi THOMAS')	Hi Thomas
replace('abcdefabcdef', 'cd', 'XX')	abXXefabXXef
length('jose')	4
md5('gop')	1c6b57c90...
ascii('a')	97
chr(97)	a

Základné (logické) operátory a funkcie

Základné funkcie na dátume a čase

- *age(timestamp, timestamp)* - odčíta argumenty.
- *age(timestamp)* - odčíta argument od current_date.
- *current_date* - vráti aktuálny dátum.
- *current_time* - vráti aktuálny čas.
- *current_timestamp* - vráti aktuálny čas a dátum (ekvivalent *now()*).
- *extract(field FROM source)* - vráti napr. rok, hodinu, minútu a pod. zo *source*, pričom *source* musí byť typu *timestamp*, *time*, *interval*, *date*.

Základné (logické) operátory a funkcie

Základné funkcie na dátume a čase - príklad

Príklad	**Výsledok**
age(timestamp '1957-06-13')	63 years 8 mons 24 days ...
date '2001-10-01' - date '2001-09-28'	3
time '05:00' - time '03:00'	02:00:00
time '05:00' - interval '2 hours'	03:00:00
900 * interval '1 second'	00:15:00

- SELECT EXTRACT(CENTURY FROM TIMESTAMP '2000-12-16 12:21:13');
  Result: 20
- SELECT EXTRACT(ISOYEAR FROM DATE '2006-01-02');
  Result: 2006
- SELECT EXTRACT(MONTH FROM TIMESTAMP '2001-02-16 20:38:40');
  Result: 2

Základné (logické) operátory a funkcie

Základné funkcie na číslach

- *abs(-5)* - absolútna hodnota.
- *ceil(15.7)* resp. *floor(15.7)* - najbližšie väčšie resp. menšie celé číslo.
- *mod(123,5)* - zvyšok po delení.
- *round(15.7)* - zaokrúhlenie.
- *sign(-5)* - znamienko.
- *trunc(165.54, 1)* - orezanie desatinných miest.
- *greatest(1,5,7,33)* - najväčší prvok.
- *least(1,5,7,33)* - najmenší prvok.
- *pi()* - konštanta π.

Klauzuly

SELECT DISTINCT

- Tabuľka v relačných DBS je multimnožina - umožňuje mať viacero rovnakých záznamov.
- **SELECT DISTINCT** - odstráni duplikáty vo výsledku.
- Príklad:

```
SELECT DISTINCT meno
FROM student;
```

Klauzuly

LIKE

- LIKE - slúži na vyhľadávanie prostredníctvom vzorov:
  - % - 0 až N ľubovoľných znakov,
  - _ - 1 ľubovoľný znak.
- Príklad:

```
SELECT priezvisko, meno
FROM student
WHERE priezvisko LIKE '%ova'
AND meno LIKE '_ana'
```

Klauzuly

BETWEEN

- Slúži na definovanie rozsahu hodnôt od-do (vrátane) alebo časových období (dátumov).
- Príklad:

```
SELECT priezvisko, meno
FROM student
WHERE rocnik BETWEEN 1 AND 5;
```

```
SELECT * FROM faktura
WHERE datum_vystavenia
NOT BETWEEN '2013-12-01' AND '2013-12-31';
```

Klauzuly

Triedenie záznamov

- Triedenie podľa hodnôt atribútov:
  - vzostupne (ASC - predvolené),
  - zostupne (DESC).
- Príklad:

```
SELECT priezvisko, meno
FROM student
ORDER BY priezvisko;
```

```
SELECT priezvisko, meno
FROM student
ORDER BY priezvisko, meno
DESC;
```

Klauzuly

LIMIT, OFFSET

- *limit* - voliteľná klauzula príkazu *select*. Obmedzí počet riadkov, ktoré vráti dotaz.
- *offset* - definuje, koľko riadkov má dotaz vynechať pred začatím počítania riadkov definovaných v *limit*.
- Syntax:

```
SELECT select_list
    FROM table_expression
    [ ORDER BY ... ]
    [ LIMIT { number | ALL } ]
    [ OFFSET number ];
```

- Príklad:

```
SELECT priezvisko, meno
FROM student
ORDER BY priezvisko, meno
LIMIT 10 OFFSET 2;
```

Klauzuly

FETCH

- *fetch* - ekvivalent *limit*, avšak je súčasťou SQL štandardu (zavedený v SQL:2008).
- Syntax:

```
OFFSET start { ROW | ROWS }
FETCH { FIRST | NEXT } [ row_count ]
{ ROW | ROWS } ONLY
```

- Príklad:

```
SELECT priezvisko, meno
FROM student
ORDER BY priezvisko, meno
OFFSET 5 ROWS
FETCH FIRST 5 ROW ONLY;
```

NULL hodnota

NULL hodnota

- Špeciálna hodnota pre neznámu alebo nedefinovanú hodnotu.
- Špecifické správanie pri WHERE klauzule, agregačných funkciách, atď.
- Na vyhodnotenie pravdivosti SQL používa trojhodnotovú logiku:

NOT(A)

A	¬A
F	T
U	U
T	F

AND(A, B)

A ∧ B	B = F	B = U	B = T
A = F	F	F	F
A = U	F	U	U
A = T	F	U	T

OR(A, B)

A ∨ B	B = F	B = U	B = T
A = F	F	U	T
A = U	U	U	T
A = T	T	T	T

Aliasy

Alias

- Umožňuje dočasne premenovať stĺpec (kľúčové slovo *AS*) alebo tabuľku (bez kľúčového slova, niekedy sa označuje ako tabuľkové premenné).
- Používa sa napríklad na:
  - skrátený zápis (čitateľnosť),
  - používa sa na rozlíšenie stĺpcov s rovnakým menom,
  - "pekné" pomenovanie stĺpca.
- Príklad:

```
SELECT d.surname || ' ' || d.name AS meno
FROM dbuser d;
```

Vytváranie entít pomocou SELECT

Vytvorenie pohľadu z výsledkov príkazu SELECT

- Pohľad je pomenovaný SELECT uložený v databáze.
- Virtuálna (odvodená) tabuľka.
- Príklad:
  CREATE VIEW nazov_pohladu **AS**
  SELECT stlpce
  FROM tabulky
  WHERE podmienky;

Vytváranie entít pomocou SELECT

Vytvorenie novej tabuľky z výsledkov príkazu SELECT

- S takto vytvorenou tabuľkou je možné robiť všetky DML príkazy jazyka SQL.
- Úpravy v novej tabuľke sa nedotknú pôvodných údajov.
- Príklad:
  CREATE TABLE názov **AS**
  SELECT stĺpce
  FROM tabuľky
  WHERE podmienky;

Vytváranie entít pomocou SELECT

SELECT v príkaze INSERT

- Vloží do tabuľky výsledok dopytu.
- Príklad:
  INSERT INTO tabuľka
  select príkaz;
- Príklad:
  INSERT INTO follows
  SELECT * FROM friendsWith;

Zdroje a použitá literatúra

- Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ, *Databázové systémy*- prednášky, FEI, TUKE, 2011 - 2016.
- `https://www.postgresql.org/docs/`.
- `https://www.postgresqltutorial.com/`.
- Ramez Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems*, Addison Wesley, 5 edition, 2006, 1168 p. ISBN 0321369572.
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan: *Database System Concepts*, The McGraw-Hill Companies, 2011, 6th edition, ISBN 978-0-07-352332-3.
- Ярцев В.П. *Організація баз даних та знань*: навчальний посібник, Державний університет телекомунікацій, 214с, 2018.

Otázky?

Databázové systémy

Jazyk SQL - Spájanie tabuliek

Ing. Ján Perháč, PhD.

Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Košice, Slovensko

Spájanie tabuliek

Úvod

- Pri normalizácii rozdeľujeme databázu na viacero tabuliek prepojených cudzími kľúčmi.
- SQL umožňuje tabuľky opäť spojiť pomocou *Joins*.
- Spojenie prostredníctvom rovnakých hodnôt v stĺpci.
- Typy spájania:
  - CROSS JOIN
  - NATURAL JOIN
  - INNER JOIN ON
  - INNER JOIN USING(attrs)
  - LEFT | RIGHT | FULL OUTER JOIN

Spájanie tabuliek

Karteziánsky súčin

- **Karteziánsky súčin** množiny A a množiny B je množina všetkých usporiadaných dvojíc:

$$(a, b),$$

ktorých prvé prvky sú prvkami množiny A a ktorých druhé prvky sú prvkami množiny B (*kombinácie "každý s každým"*).

- Počet všetkých **usporiadaných dvojíc** je definovaný súčin počtu prvkov množiny A s počtom prvkov množiny B, teda:

$$|A|.|B|$$

Vzorový príklad

Zamestnanci a oddelenia

- Obsah tabuliek:

Employee Table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department Table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

CROSS JOIN

CROSS JOIN - Karteziánsky súčin

- Reprezentuje karteziánsky súčin v SQL, t.j. nedochádza ku žiadnemu filtrovaniu.
- Syntax:
  - Explicitná notácia:
    SELECT *
    FROM tab1
    CROSS JOIN tab2;
  - Implicitná notácia:
    SELECT *
    FROM tab1,tab2;

CROSS JOIN

CROSS JOIN - Príklad

SELECT *
FROM employee
CROSS JOIN department;

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Rafferty	31	Sales	31
Jones	33	Sales	31
Steinberg	33	Sales	31
Smith	34	Sales	31
Robinson	34	Sales	31
John	NULL	Sales	31
Rafferty	31	Engineering	33
Jones	33	Engineering	33
Steinberg	33	Engineering	33
Smith	34	Engineering	33
Robinson	34	Engineering	33
John	NULL	Engineering	33

NATURAL JOIN

NATURAL JOIN - Prirodzené spájanie

- Spája záznamy tabuliek na základe rovnakých hodnôt stĺpcov s rovnakým pomenovaním.
- Nepoužívaný kvôli implicitnému vyjadreniu podmienky spájania.
- Syntax:
  SELECT *
  FROM tab1 **NATURAL JOIN** tab2;

NATURAL JOIN

NATURAL JOIN - Príklad

SELECT *
FROM employee e
NATURAL JOIN department d;

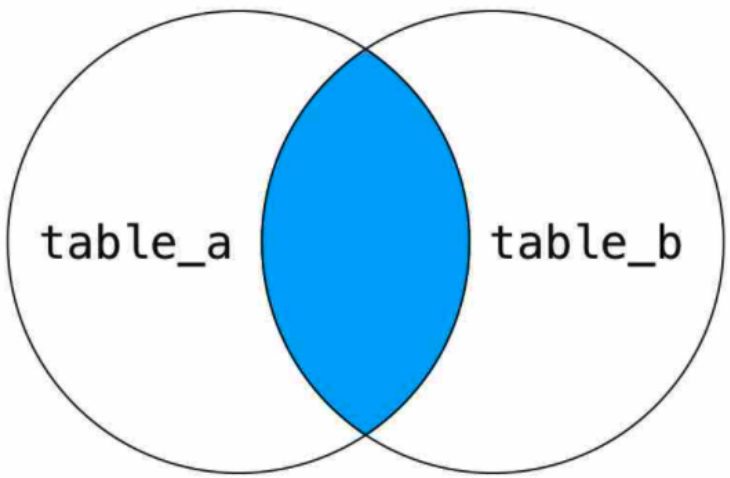
Employee.LastName	Employee.DepartmentID	Department.DepartmentName
Robinson	34	Clerical
Jones	33	Engineering
Smith	34	Clerical
Steinberg	33	Engineering
Rafferty	31	Sales

INNER JOIN

INNER JOIN - Vnútorné spájanie

- Najčastejšie používaný *JOIN*.
- Spojenie vznikne kombináciou záznamov tabuliek na základe podmienky spojenia.
- Zvyčajne spája cez cudzí a primárny kľúč.
- Syntax:
  - *Explicitná*:
    SELECT * **FROM** tab1
    INNER JOIN tab2 **ON** podmienka;
  - *Implicitná (CROSS JOIN + WHERE)*:
    SELECT * **FROM** tab1, tab2
    WHERE podmienka;

INNER JOIN - vizualizácia




```
SELECT column_name
FROM table_a
INNER JOIN table_b
ON table_a.col_name = table_b.col_name;
```

INNER JOIN

INNER JOIN - Príklad

```
SELECT * FROM employee e
INNER JOIN department d
ON e.departmentid=d.departmentid;

SELECT * FROM employee e, department d
WHERE e.departmentid=d.departmentid;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Robinson	34	Clerical	34
Jones	33	Engineering	33
Smith	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31

INNER JOIN USING

INNER JOIN USING - Vnútorné spájanie na spoločných stĺpcoch

- Explicitná alternatíva pre prirodzené spájanie.
- Spája záznamy tabuliek na základe rovnakých hodnôt explicitne vymenovaných stĺpcov s rovnakým pomenovaním.
- Syntax:
  SELECT *
  FROM tab1 **INNER JOIN** tab2
  USING(departmentid);

INNER JOIN USING

INNER JOIN USING - Príklad

SELECT * **FROM** employee e **INNER JOIN** department d **USING**(departmentid);

SELECT * **FROM** employee e **INNER JOIN** department d **ON** e.departmentid=d.departmentid;

Employee.LastName	Employee.DepartmentID	Department.DepartmentName
Robinson	34	Clerical
Jones	33	Engineering
Smith	34	Clerical
Steinberg	33	Engineering
Rafferty	31	Sales

OUTER JOIN

OUTER JOIN - Vonkajšie spájanie

- Používa sa k výpisu aj tých záznamov, ktoré nespĺňajú spojovacie kritérium (záznamov, ktoré nemajú pár z druhej tabuľky).
- Typy vonkajších spojení:
  - *Úplné vonkajšie spojenie*:
    FULL OUTER JOIN
  - *Čiastočné vonkajšie spojenie*:
    - Ľavé:
      LEFT OUTER JOIN
    - Pravé:
      RIGHT OUTER JOIN
- Umožňuje aj použitie *NATURAL* a *USING* ako alternatívu k *ON* podmienke.

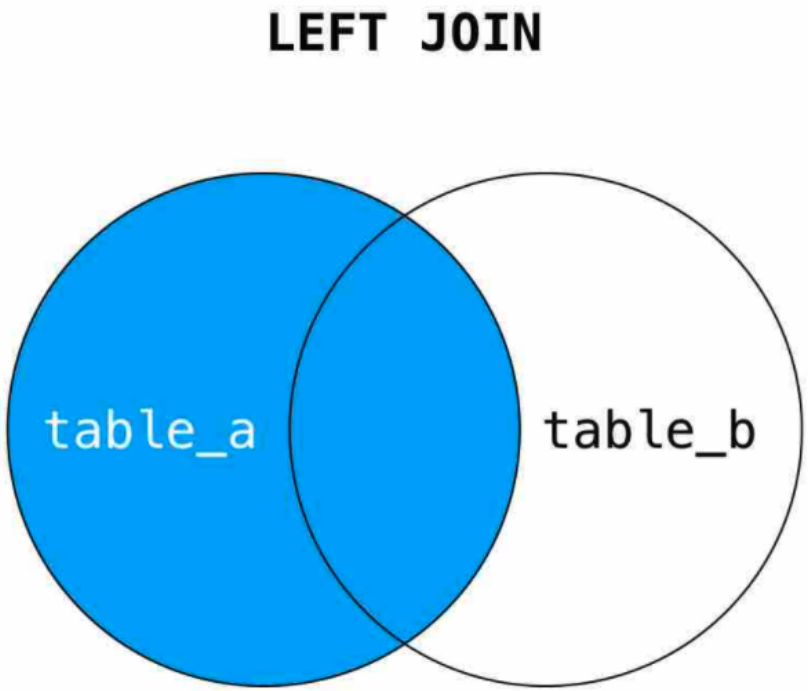
LEFT OUTER JOIN

LEFT (OUTER) JOIN - Ľavé vonkajšie spojenie

- Výsledok vždy obsahuje všetky záznamy z ľavej tabuľky.
- Každý riadok z ľavej tabuľky sa vo výsledku nachádza aspoň raz.
- Chýbajúce párové záznamy z pravej tabuľky sú nahradené hodnotou *NULL*.
- Syntax:
  SELECT * **FROM** tab1
  LEFT OUTER JOIN tab2 **ON** podm;

  SELECT * **FROM** tab1
  LEFT JOIN tab2 **ON** podm;

LEFT OUTER JOIN - vizualizácia




```
SELECT column_name
FROM table_a
LEFT JOIN table_b
ON table_a.col_name = table_b.col_name;
```

LEFT OUTER JOIN

LEFT OUTER JOIN - Príklad

SELECT * **FROM** employee e
LEFT OUTER JOIN department d
ON e.DepartmentID = d.DepartmentID;

- John nemá oddelenie (nespĺňa podmienku), vo výsledku ho však chceme.

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
John	NULL	NULL	NULL
Steinberg	33	Engineering	33

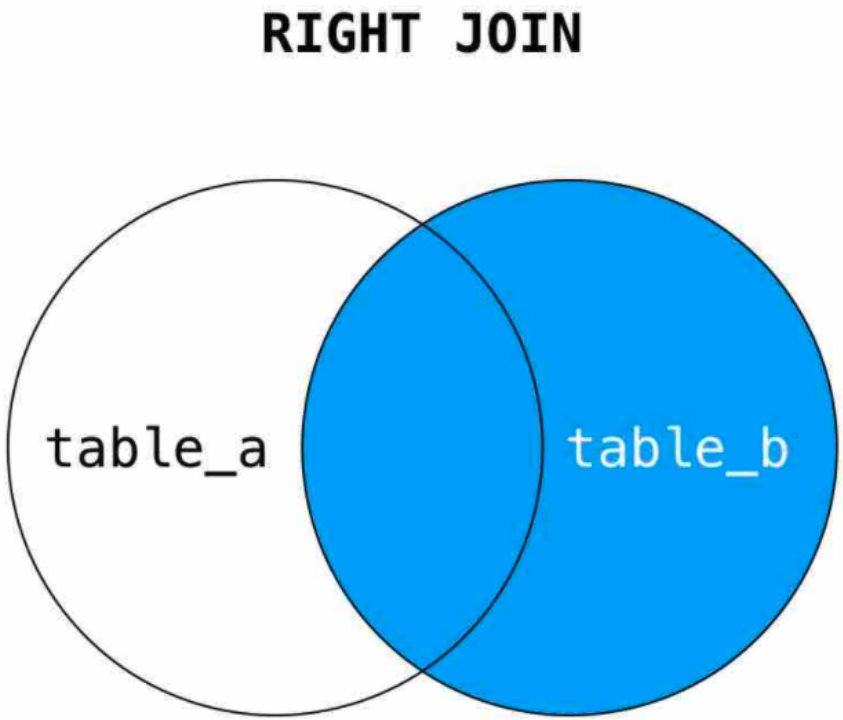
RIGHT OUTER JOIN

RIGHT (OUTER) JOIN - Pravé vonkajšie spojenie

- Výsledok vždy obsahuje všetky záznamy z pravej tabuľky.
- Každý riadok z pravej tabuľky sa vo výsledku nachádza aspoň raz.
- Chýbajúce párové záznamy z ľavej tabuľky sú nahradené hodnotou *NULL*.
- V praxi sa častejšie používa ľavé vonkajšie spojenie zámenou tabuliek.
- Syntax:
  SELECT * **FROM** tab1
  RIGHT OUTER JOIN tab2 **ON** podm;

  SELECT * **FROM** tab1
  RIGHT JOIN tab2 **ON** podm;

RIGHT OUTER JOIN - vizualizácia




```
SELECT column_name
FROM table_a
RIGHT JOIN table_b
ON table_a.col_name = table_b.col_name;
```

RIGHT OUTER JOIN

RIGHT OUTER JOIN - Príklad

SELECT * **FROM** employee e
RIGHT OUTER JOIN department d
ON e.DepartmentID = d.DepartmentID;

- Marketing nemá zamestnancov, ale chceme vidieť všetky oddelenia.

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	*Marketing*	35

FULL OUTER JOIN

FULL (OUTER) JOIN - Úplné vonkajšie spojenie

- Výsledok je zjednotením ľavého a pravého vonkajšieho spojenia.
- Vo výsledku sa nachádzajú všetky záznamy z ľavej aj všetky záznamy z pravej tabuľky.
- Tie, ktoré nemajú pár, sú doplnené *NULL* hodnotami.
- Syntax:
  SELECT * **FROM** tab1
  FULL OUTER JOIN tab2 **ON** podm;

  SELECT * **FROM** tab1
  FULL JOIN tab2 **ON** podm;

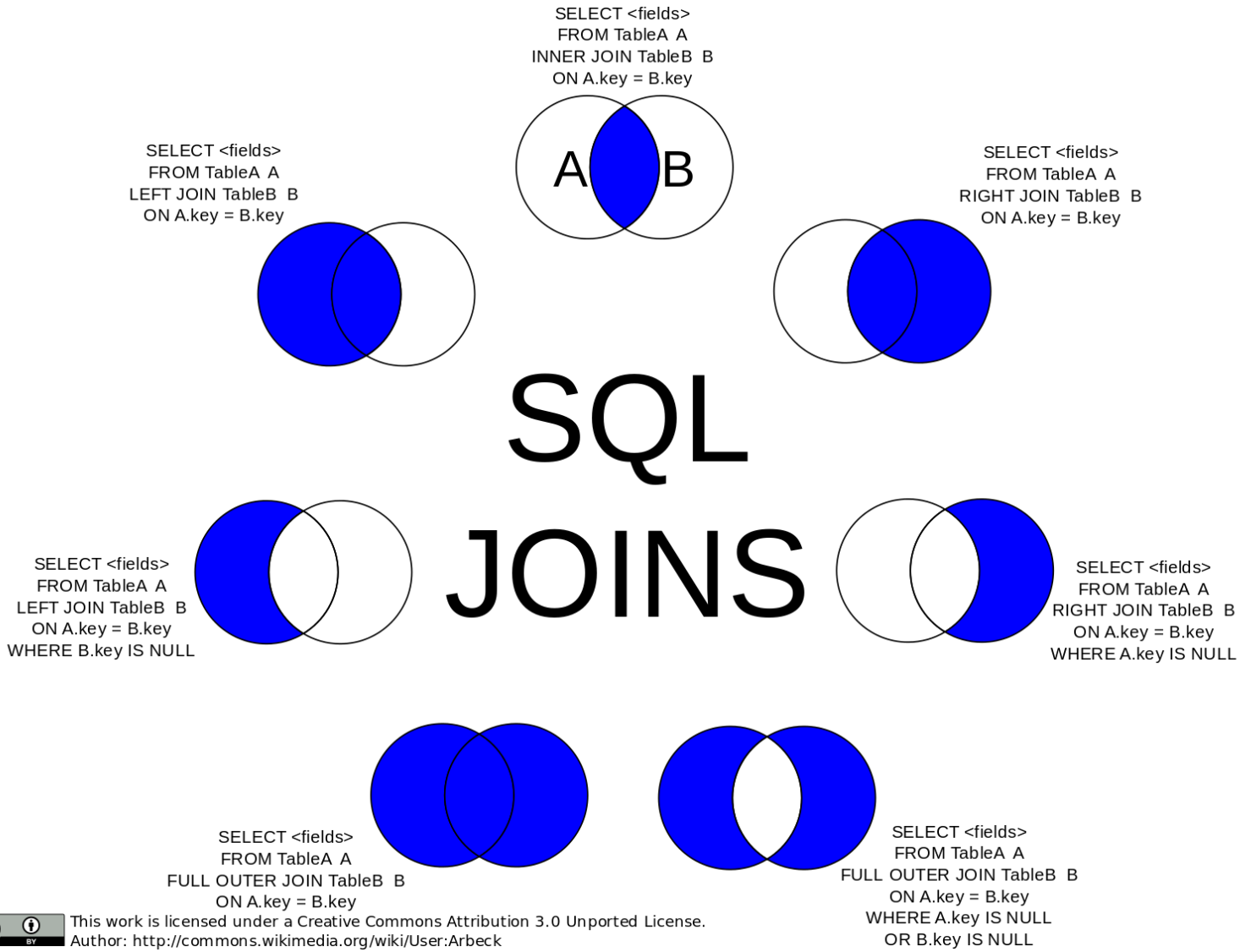
FULL OUTER JOIN

FULL OUTER JOIN - Príklad

SELECT * **FROM** employee e
FULL OUTER JOIN department d
ON e.DepartmentID = d.DepartmentID;

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
John	NULL	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	*Marketing*	35

SQL Joins - vizualizácia



ON a WHERE

Klauzuly ON a WHERE

- Klauzula **ON**:
  - Definuje podmienku spájania.
  - Vyhodnocuje sa **pred samotným spojením**.
- Klauzula **WHERE**:
  - Definuje podmienku na filtrovanie.
  - Vyhodnocuje sa **po vytvorení výsledku spájania**.
- Výber vhodnej klauzuly pre podmienku môže ovplyvniť výkon, ale aj význam SELECT-u.

ON a WHERE

ON a WHERE - Príklad

- Spojenie oddelení s ich zamestnancami a vyfiltrované sú iba tie oddelenia, ktoré nemajú zamestnanca.

SELECT * **FROM** employee e
RIGHT JOIN department d
ON e.DepartmentID = d.DepartmentID
WHERE e.lastName **IS NULL**;

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
NULL	NULL	*Marketing*	*35*

ON a WHERE

ON a WHERE - Príklad

- Test na *NULL* už počas spájania (žiadny výsledok to nespĺňa).

SELECT * **FROM** employee e
RIGHT JOIN department d
ON e.DepartmentID = d.DepartmentID
AND e.lastName **IS NULL**;

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
NULL	NULL	Sales	31
NULL	NULL	Engineering	33
NULL	NULL	Clerical	34
NULL	NULL	Marketing	35

Zhrnutie

Vlastnosti spájania

- V praxi môže poradie spájania **ovplyvniť rýchlosť** vyhodnotenia.
- Pravé a ľavé vonkajšie spájania **nie sú komutatívne**:
  - *(A left join B)* $\neq$ *(B left join A)*.
- Vonkajšie spájania nie sú asociatívne:
  - *(A left join B) left join C* $\neq$ *A left join (B left join C)*.

Zdroje a použitá literatúra

- Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ, *Databázové systémy*- prednášky, FEI, TUKE, 2011 - 2016.
- `https://www.postgresql.org/docs/`.
- `https://www.postgresqltutorial.com/`.
- `https://dyclassroom.com/mysql/`.
- Ramez Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems*, Addison Wesley, 5 edition, 2006, 1168 p. ISBN 0321369572.
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan: *Database System Concepts*, The McGraw-Hill Companies, 2011, 6th edition, ISBN 978-0-07-352332-3.
- Ярцев В.П. *Організація баз даних та знань*: навчальний посібник, Державний університет телекомунікацій, 214с, 2018.

Otázky?

Databázové systémy

Jazyk SQL - Množinové operácie

Ing. Ján Perháč, PhD.

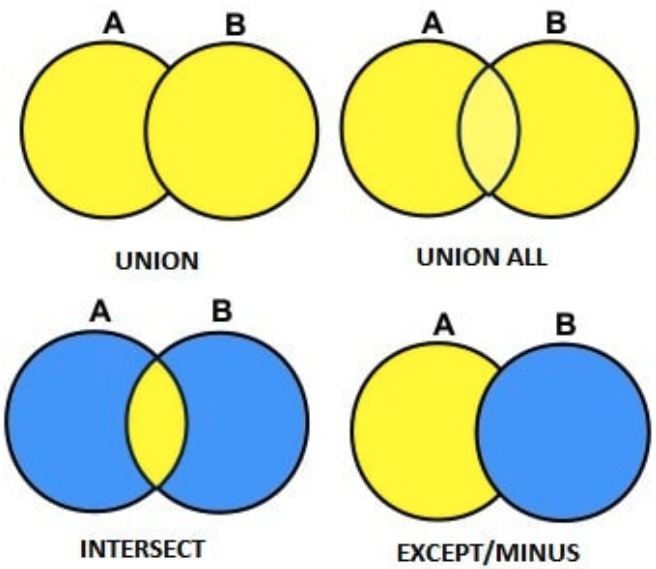
Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Košice, Slovensko

Množinové operácie

Úvod

- Operácie nad matematickými množinami.
- Množinové operácie v PostgreSQL:
  - Zjednotenie - **UNION, UNION ALL**.
  - Prienik - **INTERSECT**.
  - Rozdiel - **EXCEPT**.



UNION a UNION ALL

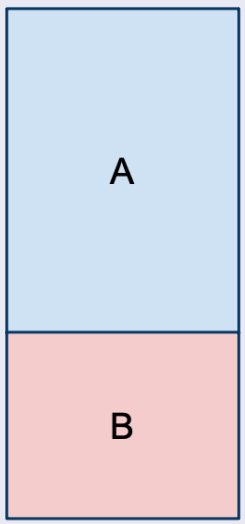
Zjednotenie tabuliek - UNION

- Vytvorí zjednotenie výsledkov dvoch (alebo viacerých) SELECT dopytov do jedného.
- Dopyty musia mať rovnakú relačnú schému, t.j. stĺpce v dopytoch:
  - Musia mať rovnaký (alebo kompatibilný) typ údajov.
  - Musí ich byť rovnaký počet.
- Vo výsledku sa nenachádzajú duplikáty.
- Syntax:

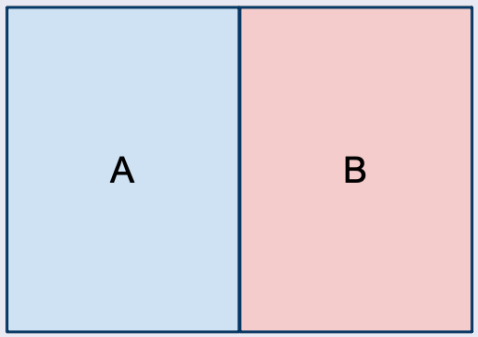
```
SELECT column(s) FROM table1
UNION
SELECT column(s) FROM table2
```

UNION a UNION ALL

Rozdiel medzi UNION a JOIN



UNION



JOIN

UNION a UNION ALL

Príklad použitia UNION

- Používateľské mená ľudí, ktorých "*user*" sleduje alebo sa s nimi priatelí:

```
SELECT followee
FROM follows
WHERE follower = 'user'
UNION
SELECT friend_with
FROM friends_with
WHERE friend = 'user';
```

UNION a UNION ALL

Zjednotenie tabuliek - UNION ALL

- Podobná funkcionalita ako *UNION*.
- Rozdiel oproti *UNION*:
  - Vo výsledkoch ostávajú všetky záznamy, **duplikáty sa neeliminujú**.
  - Výsledok nie je utriedený.
  - Rýchlejší ako *UNION*.
- Syntax:

```
SELECT column(s) FROM table1
UNION ALL
SELECT column(s) FROM table2
```

UNION a UNION ALL

Príklad použitia UNION ALL

- Používateľské mená ľudí, ktorých "*user*" sleduje alebo sa s nimi priatelí:

```
SELECT followee
FROM follows
WHERE follower = 'user'
UNION ALL
SELECT friend_with
FROM friends_with
WHERE friend = 'user';
```

UNION a UNION ALL

Vlastnosti UNION a UNION ALL

- *Alias* môže byť len v prvom *SELECT*-e.
- *ORDER BY* môže byť len jeden a musí sa nachádzať na konci príkazu.
- Ak typy stĺpcov nie sú rovnaké, ale kompatibilné, vykoná sa automatická konverzia.
- Ak typy stĺpcov sú rôzne, musia byť prekonvertované (napr. funkciou TO_CHAR()).
- Zjednotenie tabuliek s rozdielnym počtom stĺpcov (*pridaním konštanty do stĺpca navyše*).

UNION a UNION ALL

Kedy použiť UNION ALL

- Ak sa v tabuľke nachádzajú rovnaké riadky a je potrebné ich zachovať.
- Ak sa vo výsledku nemôžu nachádzať žiadne duplikáty (teda nemá zmysel ich eliminovať).
- Ak nezáleží na tom, či duplikáty existujú alebo nie.

INTERSECT

Prienik - INTERSECT

- Prienik dvoch tabuliek obsahuje všetky riadky, ktoré sú identické v oboch tabuľkách.
- Nahraditeľné pomocou *INNER JOIN*-u.
- Syntax:

```
SELECT column(s) FROM table1
INTERSECT
SELECT column(s) FROM table2;
```

INTERSECT

Príklad použitia - INTERSECT

- Používateľské mená ľudí, ktorých "*user*" sleduje a sú to zároveň jeho priatelia:

```
SELECT followee
FROM follows
WHERE follower = 'user'
INTERSECT
SELECT friend_with
FROM friends_with
WHERE friend = 'user';
```

EXCEPT

Rozdiel - EXCEPT (MINUS)

- Umožní zistiť, ktoré riadky z prvej tabuľky sa nenachádzajú v druhej tabuľke.
- Jednosmerná operácia (poradie tabuliek je dôležité):
  - A **EXCEPT** B
  - B **EXCEPT** A
- Nahraditeľné pomocou *OUTER JOIN*-u.
- Syntax:

```
SELECT column(s) FROM table1
EXCEPT
SELECT column(s) FROM table2
```

EXCEPT

Príklad použitia - EXCEPT

- Používateľské mená ľudí, ktorých "*user*" sleduje, ale nie sú to jeho priatelia:

```
SELECT followee
FROM follows
WHERE follower = 'user'
EXCEPT
SELECT friendWith
FROM friendsWith
WHERE friend = 'user';
```

Zhrnutie

Vlastnosti množinových operácií I.

- Rovnaký počet stĺpcov musí byť vybraný všetkými použitými príkazmi SELECT. Názvy stĺpcov použité pri výpise sú prevzaté z prvého dotazu.
- Dátové typy zoznamu stĺpcov musia byť kompatibilné / implicitne konvertovateľné.
  - Nebude vykonaná implicitná konverzia typov, ak zodpovedajúce stĺpce v dotazoch patria do rôznych skupín dátových typov. Napríklad ak je stĺpec v dotaze na prvý komponent dátového typu DATE a zodpovedajúci stĺpec v dotaze na druhý komponent je z dát typu CHAR, PostgreSQL nevykoná implicitnú konverziu, ale vyvolá chybu.

Zhrnutie

Vlastnosti množinových operácií II.

- Klauzula ORDER BY sa môže použiť iba raz na konci dotazu obsahujúceho zložené príkazy SELECT. To znamená, že jednotlivé príkazy SELECT nemôžu mať klauzulu ORDER BY.
- Operácie UNION a INTERSECT sú komutatívne, t. j. poradie dotazov nie je dôležité (nemení to konečný výsledok).
- Pokiaľ ide o výkon, operácia UNION ALL vykazuje lepšiu výkonnosť v porovnaní s UNION, pretože pri filtrovaní duplikátov a triedení výslednej množiny nedochádza k plytvaniu prostriedkami.
- Množinové operácie môžu byť súčasťou poddotazov.

Zdroje a použitá literatúra

- Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ, *Databázové systémy*- prednášky, FEI, TUKE, 2011 - 2016.
- `https://www.postgresql.org/docs/`.
- `https://www.postgresqltutorial.com/`.
- Ramez Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems*, Addison Wesley, 5 edition, 2006, 1168 p. ISBN 0321369572.
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan: *Database System Concepts*, The McGraw-Hill Companies, 2011, 6th edition, ISBN 978-0-07-352332-3.
- Ярцев В.П. *Організація баз даних та знань*: навчальний посібник, Державний університет телекомунікацій, 214с, 2018.

Otázky?

Databázové systémy

Jazyk SQL - Agregačné funkcie a zoskupovanie

Ing. Ján Perháč, PhD.

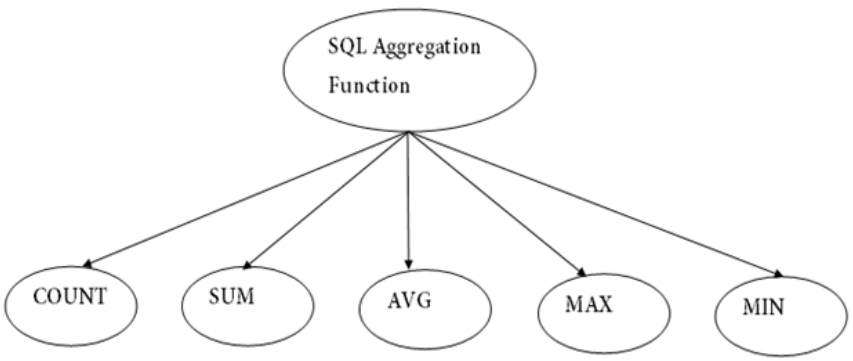
Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Košice, Slovensko

Agregačné funkcie

Úvod

- Agregačné funkcie vracajú hodnotu, ktorá je vypočítaná z hodnôt v stĺpci.
  - Agregujú kolekciu hodnôt do jednej hodnoty.
  - Hodnota *NULL* je ignorovaná.
- Pomocou agregačných funkcií je možné získať:
  - počet všetkých záznamov,
  - minimum/maximum z hodnôt,
  - priemernú hodnotu,
  - súčet.



Funkcia COUNT()

Funkcia COUNT()

- Vracia počet riadkov, ktoré vyhovujú daným kritériám.
- Použitie na: záznam, text, čísla a dátum.
- Príklady:
  SELECT COUNT(*) **FROM** table_name;

```
SELECT COUNT(collumn_name)
FROM table_name;
```

```
SELECT
COUNT(DISTINCT collumn_name)
FROM table_name;
```

Funkcia AVG()

Funkcia AVG()

- Vracia priemernú hodnotu z hodnôt nachádzajúcich sa v číselnom stĺpci.
- Použitie na: čísla.
- Príklady:
  SELECT AVG(collumn_name) **FROM** table_name;

  SELECT
  AVG(**floor**((current_date - birthday)/365))
  FROM dbuser;

Funkcie MIN() a MAX()

Funkcie MIN() a MAX()

- Vracajú najmenšiu/najväčšiu hodnotu z daného stĺpca.
- Použitie na: text, čísla a dátum.
- Príklady:
  SELECT MAX(collumn_name) **FROM** table_name;

  SELECT MIN(collumn_name) **FROM** table_name;

  SELECT MAX(population) **FROM** state;
  SELECT MIN(population) **FROM** state;

Funkcia SUM()

Funkcia SUM()

- Vráti celkový súčet hodnôt v stĺpci.
- Použitie na: čísla.
- Príklady:
  SELECT SUM(collumn_name) **FROM** table_name;

  SELECT SUM(price) **FROM** item;

Zoskupovanie záznamov

GROUP BY

- Zvyčajne sa používa s agregačnými funkciami (agregujú sa hodnoty v danej skupine).
- Určuje zoskupovanie údajov vo výstupe podľa rovnakých hodnôt určených stĺpcov.
- Syntax:

```
SELECT column_name,
       aggregate_function(column_name)
FROM table_name
WHERE exp
GROUP BY column_name;
```

Zoskupovanie záznamov

GROUP BY - príklad

- Výsledok môže obsahovať iba stĺpce prítomné v **GROUP BY** klauzule alebo agregačné funkcie.
- Príklady:

```
SELECT type, count(*)
FROM post
GROUP BY type;
```

```
SELECT type, count(*),
min(dateOfPost),
max(dateOfPost)
FROM post
GROUP BY type;
```

Zoskupovanie záznamov

Klauzula HAVING

- Používa sa v kombinácii s **GROUP BY** na filtrovanie záznamov predstavujúcich skupiny.
  - **WHERE** v dopyte filtruje záznamy **pred** ich zoskupením (môže obsahovať stĺpce, ktoré nie sú v *GROUP BY*).
  - **HAVING** filtruje záznamy **po** zoskupení.
- Príklady:

```
SELECT type, count(*)
FROM post
GROUP BY type
HAVING count(*) > 1;
```

Zoskupovanie záznamov

Klauzula ROLLUP

- Používa sa v kombinácii s **GROUP BY** na zahrnutie medzisúčtov pre zoskupovacie stĺpce sprava doľava.
- Príklad (počty používateľov s rovnakým pohlavím a vekom, plus medzisúčet pre pohlavie, a napokon pre všetkých dokopy):
  SELECT sex, **floor**((current_date - birthday)/365), **count**(*)
  FROM dbuser
  GROUP BY ROLLUP (sex, **floor**((current_date - birthday)/365));

Zoskupovanie záznamov

Klauzula CUBE

- Používa sa v kombinácii s **GROUP BY** na zahrnutie medzisúčtov pre všetky kombinácie zoskupovacích stĺpcov.
- Príklad (počty používateľov s rovnakým pohlavím a vekom, plus medzisúčet pre rovnaké pohlavie, medzisúčet pre rovnaký vek, a napokon pre všetkých dokopy): **SELECT** sex, **floor**((current_date - birthday)/365), **count**(*)
  FROM dbuser
  GROUP BY CUBE (sex, **floor**((current_date - birthday)/365));

Zhrnutie

Vlastnosti agregačných funkcií

- Agregačná funkcia produkuje jeden výsledok pre celý stĺpec alebo tabuľku.
- Agregované funkcie sa používajú na získanie súhrnných výsledkov.
- Vrátia výsledky na základe skupín riadkov.
- V predvolenom nastavení sú všetky riadky v tabuľke považované za jednu skupinu.
- Klauzula GROUP BY príkazu SELECT sa používa na rozdelenie riadkov do menších skupín.

Zdroje a použitá literatúra

- Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ, *Databázové systémy*- prednášky, FEI, TUKE, 2011 - 2016.
- `https://www.postgresql.org/docs/`.
- `https://www.postgresqltutorial.com/`.
- Ramez Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems*, Addison Wesley, 5 edition, 2006, 1168 p. ISBN 0321369572.
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan: *Database System Concepts*, The McGraw-Hill Companies, 2011, 6th edition, ISBN 978-0-07-352332-3.
- Ярцев В.П. *Організація баз даних та знань*: навчальний посібник, Державний університет телекомунікацій, 214с, 2018.

Otázky?