Databázové systémy

Dotazovací jazyk SQL - II

SELECT VIII

Vestavěné (BUILT-IN) agregační funkce

COUNT(*sloupec*) **COUNT(*)**	Počet řádků vyhovujících podmínce WHERE. Protože výsledek nezávisí na jménu sloupce uvedeného v argumentu, lze místo sloupce uvést znak *.
COUNT(DISTINCT *sloupec*)	Počet **různých** hodnot uvedeného sloupce vyskytujících se ve všech řádcích vyhovujících podmínce WHERE.
SUM(*sloupec*)	Součet hodnot ve sloupci přes všechny řádky vyhovující podmínce WHERE. Sloupec musí být některého numerického typu.
AVG(*sloupec*)	Průměr hodnot ve sloupci přes všechny řádky splňující WHERE. Sloupec musí být některého numerického typu.
MAX(*sloupec*)	Největší hodnota ve sloupci přes všechny řádky splňující WHERE. U numerických číselně, u stringu podle abecedy, ...
MIN(*sloupec*)	Nejmenší hodnota ve sloupci přes všechny řádky splňující WHERE. U numerických číselně, u stringu podle abecedy, ...

SELECT IX

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

Count:

Počet řádků nezávisí na atributu uvedeném v argumentu funkce COUNT. Stejného výsledku tedy dosáhneme, uvedeme-li na místě argumentu jméno libovolného sloupce nebo znak '*'.

```
SELECT COUNT(*)
   FROM PACKAGE
   WHERE PACKTYPE = 'Database'
```

```
SELECT COUNT(PACKID)
   FROM PACKAGE
   WHERE PACKTYPE = 'Database'
```

Výsledek:

Název sloupce zvolil databázový systém sám.

COUNT1
2

SELECT X

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

SELECT COUNT(DISTINCT *PACKNAME* **)**
FROM *PACKAGE*
WHERE *PACKTYPE* **=** 'Database'

Výsledek:

COUNT1
1

Podmínce WHERE vyhovují dva řádky. Oba však mají atribut *PACKNAME* roven řetězci ‘Manta’. Výsledkem je tudíž číslo 1, neboť ve všech řádcích vyhovujících podmínce WHERE je pouze jedna rozdílná hodnota.

SELECT XI

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

SELECT COUNT(*PACKID* **), SUM(** *PACKCOST* **)**
FROM *PACKAGE*

Výsledek:

COUNT1	SUM2
6	1968.96

SELECT XII

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

SELECT COUNT(*PACKID* **), AVG(** *PACKCOST* **)**
FROM *PACKAGE*

Výsledek:

COUNT1	AVG2
6	**328.16**

SELECT XIII

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

SELECT COUNT(*PACKID* **), MAX(** *PACKCOST* **)**
FROM *PACKAGE*

Výsledek:

COUNT1	MAX2
6	725.83

SELECT XIV

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

SELECT COUNT(*PACKID* **), MIN(** *PACKCOST* **)**
FROM *PACKAGE*

Výsledek:

COUNT1	MIN2
6	30.00

Poznámka:

1. Věty s NULL value v příslušném sloupci jsou u SUM, AVG, MAX, MIN ignorovány
2. Může nastat situace, kdy COUNT(*) a COUNT(atribut) vrátí rozdílné hodnoty a to tehdy, když pro některé věty má atribut **atribut** nepřiřazené hodnoty (NULL).

SELECT XV

Tabulka PC

TAGNUM	COMPID	EMPNUM	LOCATION
32808	M759	611	Accounting
37691	B121	124	Sales
57772	C007	567	Info Systems
59836	B221	124	Home
77740	M759	567	Home

Použitím klíčového slova **DISTINCT** v sekci SELECT zabráníme vícenásobnému uvedení téže věty ve výsledku dotazu.

SELECT *EMPNUM*
FROM *PC*

Výsledek:

EMPNUM
611
124
567
124
567

SELECT DISTINCT *EMPNUM*
FROM *PC*

Výsledek:

EMPNUM
124
567
611

GROUP BY I

Tabulka SOFTWARE

PACKID	TAGNUM	INSTDATE	SOFTCOST
AC01	32808	09/13/95	754.95
DB32	32808	12/03/95	380.00
DB32	37691	06/15/95	380.00
DB33	57772	05/27/95	412.77
WP08	32808	01/12/96	185.00
WP08	37691	06/15/95	227.50
WP08	57772	05/27/95	170.24
WP09	59836	10/30/95	35.00
WP09	77740	05/27/95	35.00

Věty, které projdou případnou podmínkou WHERE, se seskupí do skupin se stejnou hodnotou atributu *TAGNUM.* Pro každou takovou skupinu se určí suma hodnot atributu *SOFTCOST,* která se objeví ve výsledku – viz sloupec *SUM1* výsledku dotazu.
Bez uvedení sekce ORDER BY by nebylo pořadí skupin ve výsledku definováno.
Podmínka HAVING se vztahuje na celou skupinu a pomocí ní lze některé skupiny z výsledku dotazu vyfiltrovat.

SELECT *TAGNUM*, **SUM(** *SOFTCOST* **)**
FROM *SOFTWARE*
GROUP BY *TAGNUM*
ORDER BY *TAGNUM*

Výsledek:

G_TAGNUM	SUM1
32808	1319.95
37691	607.50
57772	583.01
59836	35.00
77740	35.00

SELECT *TAGNUM*, **SUM(** *SOFTCOST* **)**
FROM *SOFTWARE*
GROUP BY *TAGNUM*
HAVING SUM(*SOFTCOST* **) >** 600
ORDER BY *TAGNUM*

Výsledek:

G_TAGNUM	SUM1
32808	1319.95
37691	607.50

GROUP BY II

Tabulka SOFTWARE

PACKID	TAGNUM	INSTDATE	SOFTCOST
AC01	32808	09/13/95	754.95
DB32	32808	12/03/95	380.00
DB32	37691	06/15/95	380.00
DB33	57772	05/27/95	412.77
WP08	32808	01/12/96	185.00
WP08	37691	06/15/95	227.50
WP08	57772	05/27/95	170.24
WP09	59836	10/30/95	35.00
WP09	77740	05/27/95	35.00

```
SELECT TAGNUM, SUM( SOFTCOST )
  FROM SOFTWARE
  GROUP BY TAGNUM
  ORDER BY TAGNUM
```

Výsledek:

G_TAGNUM	SUM1
32808	1319.95
37691	607.50
57772	583.01
59836	35.00
77740	35.00

```
SELECT TAGNUM, SUM( SOFTCOST )
  FROM SOFTWARE
  GROUP BY TAGNUM
    HAVING SUM( SOFTCOST ) > 600
  ORDER BY TAGNUM
```

Výsledek:

G_TAGNUM	SUM1
32808	1319.95
37691	607.50

Další tři skupiny se do výsledku (narozdíl od příkladu vlevo) nedostaly, neboť v jejich případě nebyla hodnota sloupce SUM1 větší než 600, jak požaduje podmínka HAVING.

HAVING versus WHERE

- Podmínka **WHERE** se vyhodnocuje pro jednotlivou větu a buď pro danou hodnotu nabývá hodnoty **true** nebo **false**.
- Věty, pro něž je podmínka **WHERE** vyhodnocena jako **true** jsou vybrány do výstupu dotazu. Podmínka **WHERE** se tedy vyhodnocuje pro jednu větu a postupně se aplikuje na všechny věty vstupní tabulky (nebo joinu vstupních tabulek). V podmínce **WHERE** se tedy nemohou vyskytnout agregační funkce, protože aplikace agregační funkce na jedinou větu nemá smysl.
- Podmínka **HAVING** se vyhodnocuje pro všechny věty dané skupiny najednou. Pro danou skupinu vět nabývá hodnody **true** nabo **false**. Neaplikuje se tedy větu po větě, ale na všechny věty dané skupiny najednou.
- Skupiny s podmínkou **HAVING** vyhodnocenou jako **true** jsou vybrány do výsledku dotazu.
- Protože se podmínka **HAVING** vyhodnocuje nad několika větami současně, má smysl, aby (narozdíl od podmínky WHERE) obsahovala agregační funkce.
- Kromě agregačních funkcí může obsahovat i atributy vyjmenované v sekci **GROUP BY**.
- Jiné atributy než ty, které jsou vyjmenovány v sekci **GROUP BY**, se nemouhou v podmínce **HAVING** vyskytnout (s výjimkou výskytu na místě argumentu nějaké agregační funkce), protože mohou mít pro různé věty téže skupiny různé hodnoty a nebylo by tudíž možné určit jednoznačnou hodnotu takového atributu pro celou skupinu.

JOIN I

Tabulka EMPLOYEE

EMPNUM	EMPNAME	EMPPHONE
124	Alvarez	1212
567	Feinstein	8716
611	Dinh	2963

Tabulka PC

TAGNUM	COMPID	EMPNUM	LOCATION
32808	M759	611	Accounting
37691	B121	124	Sales
57772	C007	567	Info Systems
59836	B221	124	Home
77740	M759	567	Home

Rádi bychom se dotazovali na relaci, jež vznikne spojením těchto dvou tabulek.

JOIN II

SELECT *
FROM *PC, EMPLOYEE*

Join je spojení tabulek metodou každý s každým, t.j. každá věta z „levé“ tabulky se spáruje s každou větou z „pravé“ tabulky.
To znamená, má-li tabulka *PC* 5 řádků a tabulka *EMPLOYEE* 3 řádky, má JOIN obou tabulek 15 řádků – viz níže.

Výsledek:

TAGNUM	COMPID	EMPNUM	LOCATION	EMPNUM	EMPNAME	EMPPHONE
32808	M759	611	Accounting	124	Alvarez	1212
32808	M759	611	Accounting	567	Feinstein	8716
32808	M759	611	Accounting	611	Dinh	2963
37691	B121	124	Sales	124	Alvarez	1212
37691	B121	124	Sales	567	Feinstein	8716
37691	B121	124	Sales	611	Dinh	2963
57772	C007	567	Info Systems	124	Alvarez	1212
57772	C007	567	Info Systems	567	Feinstein	8716
57772	C007	567	Info Systems	611	Dinh	2963
59836	B221	124	Home	124	Alvarez	1212
59836	B221	124	Home	567	Feinstein	8716
59836	B221	124	Home	611	Dinh	2963
77740	M759	567	Home	124	Alvarez	1212
77740	M759	567	Home	567	Feinstein	8716
77740	M759	567	Home	611	Dinh	2963

JOIN III (equijoin)

Častější operací je tzv. **equijoin**, t.j. výsledkem je join pouze těch řádků z „levé“ a „pravé“ tabulky, které se shodují v některém sloupci. Například equijoin tabulek *PC* a *EMPLOYEE*, který je definován shodou hodnoty sloupce *EMPNUM* tabulky *PC* se sloupcem *EMPNUM* tabulky *EMPLOYEE*, se realizuje příkazem:

SELECT *TAGNUM, COMPID, EMPLOYEE.EMPNUM, EMPNAME*
FROM *PC, EMPLOYEE*
WHERE ***PC.EMPNUM = EMPLOYEE.EMPNUM***

Výsledek:

TAGNUM	COMPID	EMPLOYEE.EMPNUM	EMPNAME
32808	M759	611	Dinh
37691	B121	124	Alvarez
57772	C007	567	Feinstein
59836	B221	124	Alvarez
77740	M759	567	Feinstein

JOIN IV (equijoin)

Další příklad:

SELECT *TAGNUM, COMPID, EMPLOYEE.EMPNUM, EMPNAME*
FROM *PC, EMPLOYEE*
WHERE ***PC.EMPNUM = EMPLOYEE.EMPNUM*** **AND** ***LOCATION*** **= 'Home'**

Podmínka pro equijoin může být doplněna v sekci WHERE o další selekční podmínky.

Výsledek:

TAGNUM	COMPID	EMPLOYEE.EMPNUM	EMPNAME
59836	B221	124	Alvarez
77740	M759	567	Feinstein

JOIN V (equijoin)

V sekci USING je seznam atributů (musí mít stejná jména v obou tabulkách), přes které se provádí equi-join.

SELECT *TAGNUM, COMPID, EMPNUM, EMPNAME*
FROM *PC* **INNER JOIN** *EMPLOYEES* **USING** (*EMPNUM)*

INNER JOIN (vnitřní join) - když se k větě z první tabulky nenajde v druhé tabulce věta splňující podmínku joinu, ona věta z první tabulky se do výsledku nepromítne.

Slovo INNER se může vynechat, je default. Opakem je OUTER JOIN.

JOIN VI (equijoin)

SELECT *TAGNUM, COMPID, EMPNUM, EMPNAME*
FROM *PC* **NATURAL JOIN** *EMPLOYEES*

Klíčové slovo **NATURAL** znamená, že se equ-join provádí přes všechny stejnojmenné atributy v obou tabulkách. Pak se nemusí uvádět sekce **USING**.

JOIN VII (equijoin)

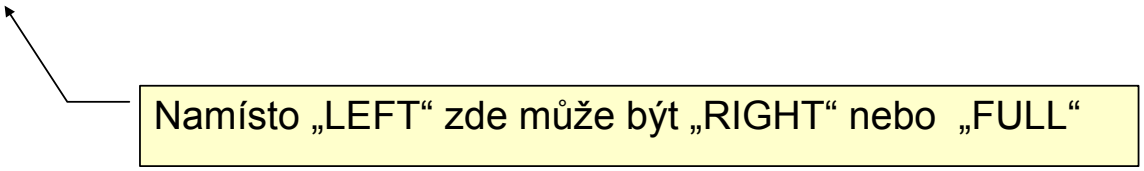
SELECT TAGNUM, COMPID, EMPNUM, EMPNAME
FROM PC **JOIN** EMPLOYEES **ON** *PC.EMPNUM = EMPLOYEES.EMPNUM*

Pokud nemají atributy, přes které se dělá equi-join, v obou tabulkách stejná jména, mohu podmínku equi-joinu vyjádřit v sekci **ON**.

JOIN VIII (OUTER JOIN)

Narozdíl od INNER JOIN se v případě OUTER JOIN do výsledku promítne věta z levé (**LEFT OUTER JOIN**), respektive z pravé **(RIGHT OUTER JOIN**), respektive z obou tabulek (**FULL OUTER JOIN**), i v případě, že nemá v druhé tabulce partnerskou větu. Atributy odpovídající chybějící partnerské větě dostanou hodnotu **NULL.**

SELECT *TAGNUM, COMPID, EMPNUM, EMPNAME*
FROM *PC* **LEFT OUTER JOIN** *EMPLOYEES*



UNION

```
SELECT COMPID, MFGNAME
    FROM COMPUTER
    WHERE PROCTYPE = '486DX'
UNION
SELECT COMPUTER.COMPID, MFGNAME
    FROM COMPUTER, PC
    WHERE COMPUTER.COMPID = PC.COMPID
          AND LOCATION = 'Home'
```

INTERSECTION

```
SELECT COMPID, MFGNAME
    FROM COMPUTER
    WHERE PROCTYPE = '486DX'
INTERSECT
SELECT COMPUTER.COMPID, MFGNAME
    FROM COMPUTER, PC
    WHERE COMPUTER.COMPID = PC.COMPID
          AND LOCATION = 'Home'
```

DIFFERENCE

```
SELECT COMPID, MFGNAME
    FROM COMPUTER
    WHERE PROCTYPE = '486DX'
EXCEPT
SELECT COMPUTER.COMPID, MFGNAME
    FROM COMPUTER, PC
    WHERE COMPUTER.COMPID = PC.COMPID
            AND LOCATION = 'Home'
```

Vnořené dotazy, subquery I

Tabulka PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

Vnořené dotazy, subquery I

```
SELECT PACKID, PACKNAME
  FROM PACKAGE
  WHERE PACKCOST >
                    ( SELECT AVG( PACKCOST )
                      FROM PACKAGE
                      WHERE PACKTYPE = 'Database' )
```

Komentář: Nejprve se vyhodnotí subquery, její výsledek se uloží do dočasné tabulky (v tomto případě 1 sloupec, 1 řádek), pak se vyhodnotí vnější query.

Výsledek vnořeného dotazu byl:

AVG1
405.09

Výsledek celého dotazu:

PACKID	PACKNAME
AC01	Boise Accounting
DB33	Manta

Vnořené dotazy, subquery II

Stejného výsledku, jako dává equijoin, lze dosáhnout i jinými prostředky – viz níže.
Měla by se však dávat přednost equijoinu před použitím zanořených dotazů.

```
SELECT PACKNAME
FROM PACKAGE
WHERE PACKID IN
   ( SELECT PACKID
     FROM SOFTWARE
     WHERE TAGNUM = '32808')
```

Výsledek:

PACKNAME
Boise Accounting
Manta

```
SELECT PACKNAME
   FROM SOFTWARE JOIN PACKAGE
   WHERE TAGNUM = '32808'
```

Výsledek:

PACKNAME
Boise Accounting
Manta

Vnořené dotazy, subquery III

IN versus EXISTS

```
SELECT TAGNUM, COMPID
FROM PC
WHERE EXISTS
 ( SELECT *
   FROM SOFTWARE
   WHERE PC.TAGNUM  = SOFTWARE.TAGNUM
         AND PACKID = 'WP08')
```

```
SELECT TAGNUM, COMPID
FROM PC
WHERE TAGNUM   IN
        ( SELECT TAGNUM
          FROM SOFTWARE
          WHERE PACKID = 'WP08')
```

Korelovaný poddotaz (correlated subquery):
Vnořený dotaz se vyhodnocuje (provádí) pro každou řádku vnějšího dotazu znovu, neboť hodnota atributu PC.TAGNUM pro momentálně vyhodnocovaný daný řádek vnějšího dotazu je vlastně parametrem dotazu vnořeného.
Použití korelovaných poddotazů bychom se měli vyhnout, neboť je mimořádně neefektivní.

Výsledek:

TAGNUM	COMPID
32808	M759
37691	B121
57772	C007

Výsledek:

TAGNUM	COMPID
32808	M759
37691	B121
57772	C007

Kvantifikátor ALL

Slovní formulace dotazu:
Najdi instalaci software, jejíž pořizovací cena byla větší než současná katalogová cena **libovolného** produktu.

SOFTWARE

PACKID	TAGNUM	INSTDATE	SOFTCOST
AC01	32808	09/13/95	754.95
DB32	32808	12/03/95	380.00
DB32	37691	06/15/95	380.00
DB33	57772	05/27/95	412.77
WP08	32808	01/12/96	185.00
WP08	37691	06/15/95	227.50
WP08	57772	05/27/95	170.24
WP09	59836	10/30/95	35.00
WP09	77740	05/27/95	35.00

SELECT *PACKID, TAGNUM, INSTDATE, SOFTCOST*
FROM *SOFTWARE*
WHERE *SOFTCOST* **> ALL**
(SELECT *PACKCOST*
FROM *PACKAGE* **)**

Výsledek:

PACKID	*TAGNUM*	*INSTDATE*	*SOFTCOST*
AC01	32808	09/13/95	754.95

Kvantifikátor ANY

Slovní formulace dotazu:
Najdi instalaci software, jejíž pořizovací cena byla větší než současná katalogová cena některého produktu.

SOFTWARE

PACKID	TAGNUM	INST DATE	SOFT COST
AC01	32808	09/13/95	754.95
DB32	32808	12/03/95	380.00
DB32	37691	06/15/95	380.00
DB33	57772	05/27/95	412.77
WP08	32808	01/12/96	185.00
WP08	37691	06/15/95	227.50
WP08	57772	05/27/95	170.24
WP09	59836	10/30/95	35.00
WP09	77740	05/27/95	35.00

SELECT *PACKID, TAGNUM, INSTDATE, SOFTCOST*
FROM *SOFTWARE*
WHERE *SOFTCOST* > **ANY**
(**SELECT** *PACKCOST*
FROM *PACKAGE*)

Výsledek:

PACKID	TAGNUM	INSTDATE	SOFTCOST
AC01	32808	09/13/95	754.95
DB32	32808	12/03/95	380.00
DB32	37691	06/15/95	380.00
DB33	57772	05/27/95	412.77
WP08	32808	01/12/96	185.00
WP08	37691	06/15/95	227.50
WP08	57772	05/27/95	170.24
WP09	59836	10/30/95	35.00
WP09	77740	05/27/95	35.00

Význam použití ALIASu

Slovní formulace dotazu:
Najdi všechny dvojice produktů mající tentýž název.

PACKAGE

PACKID	PACKNAME	PACKVER	PACKTYPE	PACKCOST
AC01	Boise Accounting	3.00	Accounting	725.83
DB32	Manta	1.50	Database	380.00
DB33	Manta	2.10	Database	430.18
SS11	Limitless View	5.30	Spreadsheet	217.95
WP08	Words & More	2.00	Word Processing	185.00
WP09	Freeware Processing	4.27	Word Processing	30.00

Tabulku *PACKAGE* otevírám 2x – jednou k ní budu přistupovat pod jménem *FIRST*, podruhé pod jménem *SECOND*. Zajímají mne tedy všechny kombinace vět z tabulek *FIRST* a *SECOND*, které se shodují v hodnotě atributu *PACKNAME (klíčové slovo AS můžeme vynechat: FROM PACKAGE FIRST)*.

```
SELECT FIRST.PACKID, FIRST.PACKNAME, SECOND.PACKID, SECOND.PACKNAME
  FROM PACKAGE AS FIRST, PACKAGE AS SECOND
  WHERE FIRST.PACKNAME = SECOND.PACKNAME AND FIRST.PACKID < SECOND.PACKID
```

Výsledek:

PACKID	PACKNAME	PACKID	PACKANME
DB32	Manta	DB33	Manta