Dotazovací jazyk SQL I

Historický vývoj I

- IBM - 70. léta - prototyp relačního DBMS - System R
- 80. léta - základ 2 komerčních DBMS**:** SQL/DS**,** DB2

SQL jako standard

- ***Standardizační instituce***
  - ANSI: American National Standards Institute
  - ISO: International Organization for Standardization
  - IEC: International Electrotechnical Commission

- ***SQL86 (někdy též SQL87) - SQL 1***
  - 1986 ANSI X3.135-1986 *Database language SQL*
  - 1987 ISO 9075-1987 *Database language SQL*

  Stále nebyla standardizována referenční integrita, ta přišla až
  - 1989 ANSI X3.135-1989 *Database Language SQL With Integrity Enhancement*
  - 1989 ISO 9075-1989 *Database language SQL*

Historický vývoj II

- ***Embedded SQL***
  - 1989 ANSI X3.168-1989 Database Language Embedded SQL
  - neexistuje ISO standard pro embedde SQL

  Embedded SQL umožňuje klást SQL dotazy z prostředí hostitelského programovacího jazyka, typicky jazyka C

  Umožňuje zapisovat dotazy přímo do zdrojového kódu v jazyce C.

  Speciální preprocesor pak tyto dotazy nahradí voláním příslušných knihovních procedur.

  Počátkem 90. let hojně využíváno při vývoji databázových aplikací v jazyce C.

Historický vývoj III

- ***SQL92 – známý jako SQL2:***
  - 1992 ANSI X3.135-1992 *Database language SQL*
  - 1992 ISO/IEC 9075-1992 *Database language SQL*

  - Dnes nejběžněji používaný standard jazyka SQL
  - Hlavně tomuto standardu se budeme na přednáškách věnovat

Další vývoj

- ***SQL99 – známý jako SQL3***
  - Regulární výrazy
  - Rekurzivní dotazy
  - Non-scalar datové typy
  - a další
- ***2003: SQL2003***
  - Podpora XML
  - Standardizované sekvence a automaticky generované hodnoty
  - *http://www.wiscorp.com/sql_2003_standard.zip*
- ***2006: SQL2006***
  - Rozšířená podpora XML (XQuery)
  - Převody XML $\leftrightarrow$ SQL
  - Export/import XML

SQL – datové typy I

Numerické datové typy ukládané přesně

INTEGER	Celá čísla	Rozsah implementačně závislý, obvykle -2147483648 až 214783647
SMALLINT	Celá čísla	Rozsah implementačně závislý, obvykle -32768 až 32767. Definice: rozsah není větší než u typu INTEGER.
NUMERIC **NUMERIC(p)** **NUMERIC(p,s)**	Číslo - může mít desetinnou část, ale uložené přesně. Dekadické číslo o *p* cifrách, z toho *s* za desetinnou čárkou. Např. DECIMAL(5,2) má 3 pozice před čárkou, 2 pozice za čárkou. Pokud *p* nebo *s* nevyjádřeno, vezme se implementačně závislý default. Číslo vždy uloženo v předepsané přesnosti (ledaže by se narazilo na implementačně dané maximum).	
DECIMAL **DECIMAL(p)** **DECIMAL(p,s)**	Obdoba jako NUMERIC, avšak číslo může být ukládáno přesněji než je požadavek (ledaže by se narazilo na implementačně dané maximum).	

Poznámka: NUMERIC zřejmě vždy ukládáno jako řetězec cifer - znaků, zatímco DECIMAL ve vnitřním formátu, pro repúrezentaci se použije tolik bytů, aby bylo dosaženo (alespoň) požadované přesnosti.

SQL – datové typy II

Numerické datové typy ukládané nepřesně

REAL	plovoucí řádová čárka jednoduchá přesnost	přesnost implementačně závislá, obvykle defaultní přesnost pro data s plovoucí řádovou čárku v jednoduché přesnosti pro danou hardwarovou platformu
DOUBLE PRECISON	plovoucí řádová čárka v dvojnás. přesnosti	přesnost implementačně závislá, obvykle defaultní přesnost pro data s plovoucí řádovou čárku v dvojnásobné přesnosti pro danou hardwarovou platformu. Definice: přesnost větší než u typu REAL.
FLOAT **FLOAT(p)**	Dovoluje definovat požadovanou přesnost. Přesnost uložení může být větší než požadovaná. Požadovaná přesnost *p* může být na jedné platformě realizována jednoduchou přesností, zatímco na jiné platformě dvojnásobnou přesností.	

Poznámka: Nepřesnost uložení je dána tím, že se jedná o plovoucí řádovou čárku. Číslo je vyjádřeno dvojicí *<mantisa, exponent>*

SQL – datové typy III

Znakové řetězce

CHARACTER **CHAR**ACTER**(x)**	Znakový řetězec o definované délce. Není-li x vyjádřeno, jedná se o CHAR(1).
CHARACTER **VAR**YING **VARCHAR** **CHAR**ACTER **VAR**YING**(x)** **VARCHAR(x)**	Znakové řetězce s proměnnou délkou (alokováno tolik byte, kolik je potřeba) – maximální délka stringu (počet znaků v řetězci) omezena číslem *x*. Maximální hodnota *x* stringu implementačně závislá.
NATIONAL **CHAR**ACTER **NCHAR** **NVARCHAR**	Pro potřeby národních abeced. UNICODE

SQL – datové typy IV

Datum a čas

DATE	Datum: rok 4 cifry, měsíc 2 cifry, den 2 cifry Délka definována na 10 znaků včetně oddělovačů **YYYY-MM-DD**
TIME **TIME(p)**	Čas: hodiny(2 cifry), minuty(2 cifry), vteřiny (2 cifry, navíc možno p desetinných míst) Délka 8 cifer pokud p=0, jinak 9+p Default 0 desetinných míst **HH:MM:SS**
TIMESTAMP **TIMESTAMP(p)**	Datum + čas. Délka 19 cifer pokud p=0, jinak 20+p Default 6 desetinných míst **YYYY-MM-DD HH:MM:SS**

CREATE TABLE I

CREATE TABLE *PACKAGE*
(*PACKID* **CHAR(***4***),**
PACKNAME **CHAR(***20***),**
PACKVER **DECIMAL(***3,2***),**
PACKTYPE **CHAR(***15***),**
PACKCOST **DECIMAL(***5,2***))**

Jméno tabulky, která má být vytvořena

Jmeno atributu

Typ atributu

Vytvoří prázdnou tabulku s 5 definovanými sloupci příslušných typů.

DROP TABLE *Computer*

Jméno tabulky, která má být zrušena

Zruší existující tabulku daného jména.

CREATE TABLE II (integritní omezení atributu)

```
CREATE TABLE COMPUTER
    (   COMPID      DECIMAL(2)  NOT NULL,
        MFGNAME     CHAR(15)    NOT NULL,
        MFGMODEL    CHAR(25),
        PROCTYPE    DECIMAL(7,2)    )
```

Integritní omezení (atributu), specifikující, že daný atribut musí mít povinně vyplněnou hodnotu

Vkládáme-li do tabulky nový řádek, nemusíme v obecném případě specifikovat hodnoty všech atributů (sloupců). Takový řádek pak bude mít ve sloupcích, pro něž jsme nezadali hodnotu, hodnotu uvedenu NULL.

Pokud ovšem při vkládání řádku do tabulky neuvedeme hodnotu takového atributu, který má specifikováno integritní omezení NOT NULL, databázový engine odmítne takový řádek do tabulky vložit (chybová hláška nebo výjimka), protože by došlo k porušení příslušného integritního omezení.

CREATE TABLE III (integritní omezení atributu)

```
CREATE TABLE Films
    (   CODE CHAR(5)   CONSTRAINT Firstkey  PRIMARY KEY
        TITLE          VARCHAR(40)  NOT NULL,
        DateProd       DATE,
        KIND           VARCHAR(10),
        LEN            INTERVAL HOUR TO MINUTE
```

Integritní omezení může být (ale nemusí a obvykle nebývá) pojmenováno. Šedivý text tedy může být vynechán.

Toto integritní omezení říká, že atribut CODE je primárním klíčem. Musí mít tudíž povinně zadanou hodnotu a tato hodnota musí být unikátní přes všechny řádky dané tabulky.

CREATE TABLE IV (integritní omezení tabulky)

```
CREATE TABLE Films
    (   TITLE          VARCHAR(40)  NOT NULL,
        DateProd       DATE,
        KIND           VARCHAR(10),
        LEN            INTERVAL HOUR TO MINUTE  ),
        CONSTRAINT pk_const PRIMARY KEY (TITLE, DateProd)
    )
```

Integritní omezení tabulky

Nepovinné jméno integritního omezení tabulky.
Integritní omezení je vhodné pojmenovávat, abychom je mohli popřípadě odtsranit, pokud nevyhovují:

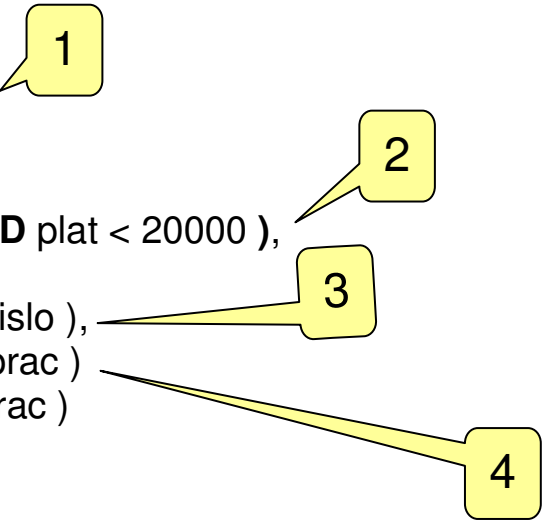
ALTER TABLE *Films* **DROP CONSTRAINT** pk_const;

V případě, že je prim
skutečnost vyjádřit in
na primárním klíči totiž
primárním klíčem dvojice atributů (*TITLE*, *DateProd*), což vyjádříme uvedeným integritním omezením tabulky. Unikátní přes všechny řádky nemá být hodnota každého z atributů *TITLE, DateProd*, ale jejich kombinace.

PRIMARY KEY je jedním z možných integritních omezení tabulky.

CREATE TABLE V (integritní omezení)

```
CREATE TABLE osoby (
  os_cislo   NUMBER(5)    NOT NULL,
  rod_cis    VARCHAR2(30) NOT NULL UNIQUE,
  jmeno      VARCHAR2(30) NOT NULL,
  prijmeni   VARCHAR2(30) NOT NULL,
  plat       NUMBER(5)    CHECK ( plat > 5000 AND plat < 20000 ),
  cislo_prac NUMBER(5)    NOT NULL,
  CONSTRAINT pk_osoby PRIMARY KEY ( os_cislo ),
  CONSTRAINT fk_prac  FOREIGN KEY ( cislo_prac )
                 REFERENCES pracoviste ( cislo_prac )
                 ON UPDATE CASCADE
                 ON DELETE CASCADE
)
```



1. Atribut může mít zadáno více integritních omezení současně – v tomto případě je hodnota atributu povinná (NOT NULL) a unikátní (UNIQUE) přes všechny řádky.
2. Integritní omezení může být zadáno i obecnou podmínkou, která musí být pro vkládaný řádek TRUE, jinak chyba.
3. Libovolné integritní omezení atributu může být rovněž vyjádřeno jako integritní omezení tabulky. V tomto případě jsme mohli skutečnost, že os_cislo je primárním klíčem, rovnocenně vyjádřit integritním omezením atributu rod_cis.
4. Toto je tzv. referenční integrita – bude probrána na samostatném slajdu.

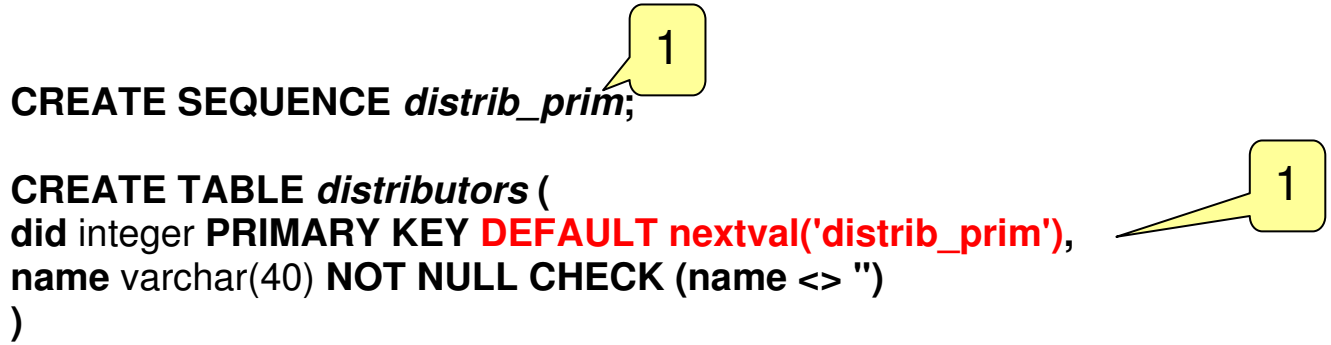
CREATE TABLE VI (integritní omezení)

```
CREATE TABLE PREDMETY (
  zkratka       VARCHAR2(10) PRIMARY KEY,
  nazev         VARCHAR2(30) NOT NULL,
  kredity       NUMBER(2) DEFAULT 2,
);
```

Pomocí svého druhu integritního omezení můžeme definovat i defaultní hodnotu atributu.

Budeme-li vkládat řádek do tabulky vytvořené výše uvedeným příkazem a neuvedeme-li přitom hodnotu sloupce kredity, nezústane tento sloupec nevyplněn (NULL), ale bude mít hodnotu 2.

CREATE TABLE VII (generování hodnot)



1. Nejprve definujeme tzv. sekvenci. V daném případě jsme ji pojmenovali *distrib_prim*.
2. Při vkládání nového řádku bude chtít integritní omezení DEFAULT přiřadit sloupci *did* vkládaného řádku hodnotu. Tuto hodnotu zjistí vyhodnocením funkce nextval(), jež ovšem vygeneruje nový (ještě neexistující) prvek sekvence *distrib_prim*. Jako výsledek bude mít každý řádek vygenerovanou unikátní hodnotu sloupce *did*.

Toto není SQL standard, ale syntax DB systému ProgreSQL. Generování hodnot bylo standardizováno až v SQL2006.

REFERENČNÍ INTEGRITA I

```
CREATE TABLE osoby (
  os_cislo   NUMBER(5)    PRIMARY KEY,
  rod_cis    VARCHAR2(30) NOT NULL UNIQUE,
  cislo_prac NUMBER(5)    NOT NULL,
  CONSTRAINT fk_prac  FOREIGN KEY ( cislo_prac )
                 REFERENCES pracoviste ( cislo_prac )
                 ON UPDATE CASCADE
                 ON DELETE CASCADE
)
```

1. Integritní omezení *fk_prac* říká, že atribut *cislo_prac* je cizím klíčem, jehož hodnota odkazuje na takový řádek tabulky *pracoviste,* jehož hodnota primárního klíče se shoduje s hodnotou atributu *cislo_prac.*
2. Znamená to tedy, že řádky reprezentující všechny osoby z daného pracoviště se prostřednictvím atributu *cislo_prac* odkazují na stejnou řádku tabulky *pracoviste*, jež odpovídá jejich společnému pracovišti.
3. Co se stane, když někdo změní hodnotu primárního klíče jejich společného pracoviště? To řeší sekce *ON UPDATE ...* V žádném případě by nemělo dojít k tomu, že nějaký řádek tabulky *osoby* bude odkazovat na neexistující řádek tabulky *pracoviste*. V daném případě je specifikováno *ON UPDATE* ***CASCADE*****.** To znamená, že příkaz modifikující hodnotu primárního klíče bude proveden, ale současně budou změněny příslušným způsobem hodnoty cizího klíče (atributu *cislo_prac*) u všech řádků tabulky *osoby*, jež reprezentují osoby zařazené na pracoviště, jehož primární klíč jsme změnili.

REFERENČNÍ INTEGRITA II

```
CREATE TABLE osoby (
  os_cislo   NUMBER(5)    PRIMARY KEY,
  rod_cis    VARCHAR2(30) NOT NULL UNIQUE,
  cislo_prac NUMBER(5)    NOT NULL,
  CONSTRAINT fk_prac  FOREIGN KEY ( cislo_prac )
                REFERENCES pracoviste ( cislo_prac )
                ON UPDATE CASCADE
                ON DELETE CASCADE
)
```

4. Co se stane, když někdo zruší řádek tabulky pracoviště, který reprezentuje jejich společné pracoviště? To řeší sekce *ON DELETE ...* Opět by v žádném případě nemělo dojít k tomu, že nějaký řádek tabulky *osoby* bude odkazovat na neexistující řádek tabulky *pracoviste*.
V daném případě je specifikováno *ON DELETE* ***CASCADE*****.** To znamená, že příkaz rušící jejich společné pracoviště bude proveden, ale současně budou smazány i všechny řádky tabulky *osoby*, jež odpovídaly odsobám pracujícím na zrušeném pracovišti.

REFERENČNÍ INTEGRITA III

```
CREATE TABLE osoby (
  os_cislo   NUMBER(5)    PRIMARY KEY,
  rod_cis    VARCHAR2(30) NOT NULL UNIQUE,
  cislo_prac NUMBER(5)    NOT NULL,
  CONSTRAINT fk_prac  FOREIGN KEY ( cislo_prac )
                REFERENCES pracoviste ( cislo_prac )
                ON UPDATE CASCADE
                ON DELETE CASCADE
)
```

Jaké jsou jiné možnosti než ***CASCADE***?

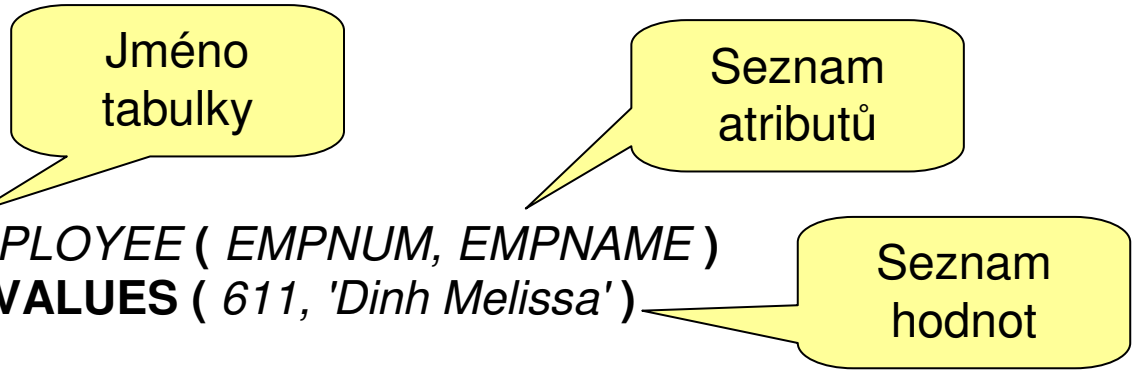
1. RESTRICT – změna, která by porušila referenční integritu se neprovede. DB systém odmítne změnu primárního klíče (ON UPDATE) nebo rušení řádku (ON DELETE) provést – chybová hláška, výjimka.
2. SET NULL – změna se provede, ale řádky tabulky *osoby* odkazující svým cizím klíčem na modifikovaný (změna primárního klíče nebo zrušení řádku) řádek tabulky *pracoviste* dostanou ve sloupci *cislo_prac* hodnotu NULL (Nebudou tedy odkazovat na neexistující řádek).
3. SET DEFAULT – obdoba jako SET NULL určená pro případ, že cizí klíč má definovanou defaultní hodnotu.

Modifikátory *CASCADE, RSTRICT, SET NULL, SET DEFAULT* se v sekcích *ON UPDATE* a *ON DELETE* nastavují nezávisle.

INSERT INTO

INSERT INTO *EMPLOYEE* **VALUES (** *611, 'Dinh Melissa', 2963* **)**

Seznam atributů neuveden – hodnoty budou atributům přiřazeny v pořadí, jak byly atributy definovány v příkazu CREATE TABLE.



INSERT INTO *EMPLOYEE* **(** *EMPNUM, EMPNAME* **)**
VALUES (*611, 'Dinh Melissa'* **)**

V pořadí, v jekém jsou zde vyjmenovány atributy, bude těmto atributům přiřazena hodnota ze seznamu hodnot v sekci VALUES.

Atributy neuvedené v tomto seznamu neuvedené dostanou hodnotu NULL (t.j. hodnota nedefinována).

V tomto případě EMPPHONE dostane hodnotu NULL.

SELECT I

SELECT *<seznam sloupců na výstupu nebo *>*
FROM *<definice relace, v níž se vyhledává>*
WHERE *<selekční podmínka>*
GROUP BY *<seznam atributů>*
HAVING *<podmínka filtrace skupin>*
ORDER BY *<seznam atributů> <vzestupně nebo sestupně>*

Logické operátory pro konstrukci logických podmínek:

= rovná se
<= menší nebo rovno
< menší než
>= větší nebo rovno
> větší než
<> nerovná se
!= nerovná se

SELECT II

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 PACKID, PACKNAME, PACKCOST
    FROM PACKAGE
    WHERE   PACKCOST > 200 AND
            PACKCOST < 400
```

```
SELECT PACKID, PACKNAME, PACKCOST
    FROM PACKAGE
    WHERE PACKCOST
        BETWEEN 200 AND 400
```

Výsledek:

PACKID	PACKNAME	PACKCOST
DB32	Manta	380.00
SS11	Limitless View	217.95

SELECT III

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

boolean predikát LIKE
znak % plní roli wildcard, t.j. shoda s libovolným znakovým řetězcem

SELECT *PACKID, PACKNAME*
FROM *PACKAGE*
WHERE *PACKNAME* ***LIKE*** *'%&%'*

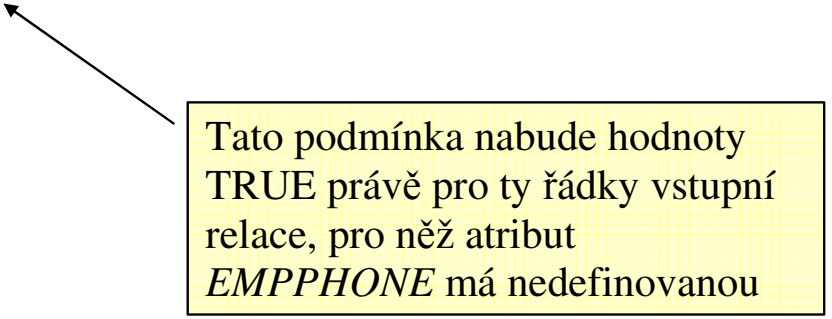
Výsledek:

PACKID	PACKNAME
WP08	Words & More

SELECT IV

Predikát **"IS NULL"** nabývá hodnoty **„true“** právě tehdy když příslušný atribut nemá přiřazenu hodnotu

SELECT *EMPNUM, EMPNAME*
FROM *EMPLOYEE*
WHERE *EMPPHONE* **IS NULL**



SELECT V (Aritmetické operátory)

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 PACKID, PACKNAME**, (.90 *** PACKCOST **)**
FROM **PACKAGE**

DBMS sám vymyslel jméno tohoto sloupce, protože nebylo možné převzít jméno příslušného atributu. EXP znamená *expression* = „*výraz*“.

Výsledek:

t.j. 0.9 * 725.83
0.9 * 380.00
0.9 * 430.18

PACKID	PACKNAME	EXP1
AC01	Boise Accounting	635.25
DB32	Manta	342.00
DB33	Manta	387.16
SS11	Limitless View	196.16
WP08	Words & More	166.50
WP09	Freeware Processing	27.00

SELECT VI

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 PACKID, PACKNAME, PACKTYPE
FROM PACKAGE
WHERE PACKTYPE IN ('Database',
                   'Spreadsheet',
                   'Word Processing')
```

```
SELECT PACKID, PACKNAME,
PACKTYPE
FROM PACKAGE
WHERE PACKTYPE = 'Database' OR
      PACKTYPE = Spreadsheet' OR
      PACKTYPE = 'Word Processing'
```

Výsledek:

PACKID	PACKNAME	PACKTYPE
DB32	Manta	Database
DB33	Manta	Database
SS11	Limitless View	Spreadsheet
WP08	Words & More	Word Processing
WP09	Freeware Processing	Word Processing

SELECT VII (třídění)

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

Pořadí vět ve výsledku dotazu není určeno, pokud nepředepíšeme, jakým způsobem mají být setříděny.

SELECT *PACKID, PACKNAME, PACKTYPE, PACKCOST*
FORM *PACKAGE*
ORDER BY *PACKTYPE, PACKCOST* **DESC**

Věty budou setříděny podle atributů *PACKTYPE* a *PACKCOST*. Budou tedy setříděny podle atributu *PACKTYPE,* věty se stejnou hodnotou atributu *PACKTYPE* pak budou setříděny podle atributu *PACKCOST*.
DESC znamená descending, t.j. sestupné setřídění. Vzestupné setřídění se předepíše klíč. slovem ASC, což je default, takže je není třeba uvádět.

Výsledek:

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