

Základy jazyka SQL pro IBM i

Vladimír Župka

Obsah

Obsah.....	2
Předmluva.....	5
Úvod	6
Základní pojmy – terminologie.....	6
Jazyk SQL	7
Relační databáze v systému IBM i	7
Systémový katalog.....	8
Tabulka, řádek, sloupec (table, row, column).....	8
Index.....	8
Pohled (view).....	8
Schema (collection)	9
Interakční SQL – průvodce obrazovkami	10
Zahájení.....	10
Vytvoření a zrušení kolekce.....	13
Vytvoření tabulky	14
Zápis dat do tabulky.....	16
Dotazy příkazem SELECT.....	18
Změna atributů relace.....	23
Statement processing	24
SELECT output.....	24
Collection list.....	25
List type	25
Vytvoření tabulek pro příklady.....	26
Tabulka CENY	26
Tabulka STAVY	26
Tabulka OBRATY.....	26
Schema tabulek a jejich vztahu	26
Data tabulek.....	27
Dotazy	28
Párové řádky ze dvou tabulek - WHERE.....	28
Výběr ze dvou a více tabulek - JOIN	29
JOIN.....	29
LEFT JOIN	30
RIGHT JOIN.....	30
FULL OUTER JOIN	30
EXCEPTION JOIN.....	31
CROSS JOIN	31
Přejmenované sloupce a pojmenovaný výraz v seznamu sloupců	31
Poddotaz (subquery)	32
Výběr jedné hodnoty (skalární subselect).....	32
Výběr množiny hodnot	33
Hierarchický dotaz	35
Příklad hierarchického dotazu	35
Operace s tabulkami (fullselect)	36
Sjednocení tabulek s duplicitami	37
Sjednocení tabulek bez duplicit	37
Průnik tabulek (bez duplicit).....	38
Řádky z první tabulky, které nejsou v druhé tabulce	38
Vnořený tabulkový výraz (nested table expression)	38
Společný tabulkový výraz (common table expression)	38
Rekurzivní hledání - kusovník.....	39
Rozpad kusovníku	39
Výskyt zvolené podsoučásti v ostatních součástech	40
Sumář podsoučástí pro výrobek	40

Kontrola zacyklení rekurze	41
<i>OLAP (On-Line Analytical Processing)</i>	42
Hodnocení (ranking)	42
Číslování řádků	43
GROUP BY	45
GROUP BY - jednoduché seskupování	45
GROUP BY GROUPING SETS - seznamy skupin	46
GROUP BY ROLLUP - skupinové úrovně	46
Hierarchické součty bez detailů	46
Skupinové součty s detaily	47
Skupinové součty bez detailů s vynescháním úrovně	48
GROUP BY CUBE - skupiny více rozměrů	48
HAVING	50
Skupinové součty s výběrem skupin	50
Zjišťování údajů o databázi	50
<i>Seznam schemat (knihoven) z databáze</i>	50
<i>Seznam tabulek (souborů) ve schematu (knihovně)</i>	51
<i>Seznam sloupců tabulky (polí souboru)</i>	52
Vytvoření tabulky příkazem CREATE TABLE	52
Změna tabulky příkazem ALTER TABLE	53
Úpravy tabulek příkazem UPDATE	54
Vkládání dat do tabulky	55
<i>Vložení jednoho řádku</i>	55
<i>Vložení více řádků</i>	56
Odstraňování dat z tabulky	56
Kopírování tabulek a dat	56
<i>Vytvoření nové tabulky podle jiné tabulky</i>	56
<i>Kopírování dat do nově vytvořené tabulky</i>	57
<i>Kopírování dat do existující tabulky</i>	57
Úprava tabulky podle druhé tabulky - příkaz MERGE	57
Pohled na tabulky (view)	57
Index	58
Integrita dat	58
<i>Primární klíč</i>	59
Tabulka CENY	59
Tabulka STAVY	59
<i>Referenční integrita</i>	60
Tabulka STAVY	60
Tabulka OBRATY	61
Schema referenčních omezení	62
Typy dat	63
<i>INTEGER nebo INT</i>	63
<i>SMALLINT</i>	63
<i>DECIMAL(n,d) nebo DEC(n,d)</i>	63
<i>DECIMAL(n) nebo DEC(n) – stejné jako DECIMAL(n,0)</i>	63
<i>DECIMAL nebo DEC – stejné jako DECIMAL(5,0)</i>	63
<i>NUMERIC(n,d)</i>	63
<i>NUMERIC(n) – stejné jako NUMERIC(n,0)</i>	63
<i>NUMERIC – stejné jako NUMERIC(5,0)</i>	63
<i>FLOAT</i>	63
<i>FLOAT(n)</i>	63

<i>REAL</i>	63
<i>DOUBLE PRECISION or DOUBLE</i>	63
<i>CHARACTER(n) nebo CHAR(n)</i>	63
<i>CHARACTER nebo CHAR – stejné jako CHARACTER(1)</i>	63
<i>VARCHAR(n)</i>	64
<i>CHARACTER VARYING (n) nebo CHAR VARYING (n)</i>	64
<i>DATE</i>	64
<i>TIME</i>	64
<i>TIMESTAMP</i>	65
<i>BLOB [(délka [K M G])]</i>	65
<i>CLOB [(délka [K M G])]</i>	65
<i>NCLOB a DBCLOB [(délka [K M G])]</i>	65
<i>DATALINK(délka) nebo DATALINK</i>	65
Speciální registry	66
Standardní funkce	66
Agregátní (sloupcové) funkce	66
Skalární funkce	67
Změny typu dat (cast scalar functions)	67
Datum a čas (datetime scalar functions)	67
Různé skalární funkce	68
Číselné skalární funkce	69
Znakové skalární funkce	70
Výraz (expression)	72
Case expression	73
Cast expression (přetypování)	74
Predikáty	75
Základní predikáty	75
Kvantifikované predikáty	75
Predikát BETWEEN	75
Predikát DISTINCT (odlišný)	75
Predikát EXISTS	76
Predikát IN	76
Predikát LIKE	76
Predikát NULL	77
Syntaktické definice příkazů	77
CREATE TABLE	77
SELECT	82
select-clause	82
from-clause	82
hierarchical-query-clause	83
fullselect	84
select-statement	84
common-table-expression	85
CREATE VIEW	86
CREATE INDEX	86
CREATE SEQUENCE	87
INSERT	88
UPDATE	88
ALTER TABLE	89
DROP	92

Předmluva

Tento kurz je určen zájemcům o jazyk SQL, kteří nejsou profesionální programátoři, ale kteří již mají zkušenosti s databázovým systémem DB2. Měli by znát zejména vytváření fyzických souborů příkazem CRTPF, logických souborů příkazem CRTLTF a dalších souvisejících příkazů. Dále je dobré znát některé služební programy systému IBM i, zejména dotazovací program Query, Data File Utility (DFU), příkaz Copy File (CPYF) apod., není to však podmínka.

Příkazy jazyka SQL se používají také v programovacích jazycích, což ovšem vyžaduje znalost programování v dotedném jazyku. To však není předmět tohoto kurzu.

Účelem kurzu je naučit účastníky nejdůležitější příkazy jazyka SQL a jejich vytváření i použití interakčním způsobem v příkazu STRSQL nebo dávkově v různých skriptech.

Přestože systém řízení databáze – DBMS (database management system) – je společný pro SQL a operační systém, existují rozdíly v terminologii i ve způsobech ovládání databáze. V kurzu jsou vztahy mezi pojmy užívanými v operačním systému a pojmy užívanými v SQL také probírány.

Úvod

Základní pojmy – terminologie

Databází se rozumí množina dat ovládaná pomocí řídicího systému báze dat. V době vzniku databázových systémů v 60. letech 20. století existovaly rozličné přístupy k organizaci dat, zejména tzv. síťová metoda (systémy IDMS, TOTAL) nebo hierarchická metoda (systémy IMS, DL/1), které jsou dnes téměř zapomenuty. Místo nich se ujal tzv. relační přístup a dotazovací jazyk SQL, který se používá dodnes. Později byly vyvinuty ještě další organizační metody, jako např. objektové databáze nebo XML (Extended Markup Language), které ovšem relační metodu nena nahradily.

Základem relačních databází je pojem *databázové tabulky*. Ta představuje množinu dat organizovanou do sloupců a řádků. V následující tabulce je znázorněna databázová tabulka o třech sloupcích a dvou řádcích, přičemž sloupce mají v nadpisech názvy.

Číslo zboží	Cena za jednotku	Název zboží
15425	17.50	Žitný chléb
85400	155000000.00	Lokomotiva elektrická

Domény jsou množiny všech možných hodnot určitého typu, např. množina všech možných názvů zboží (nejen těch, které jsou v konkrétní tabulce opravdu použity). *Sloupce* (columns) obsahují hodnoty z domén. *Řádky* (rows) pak představují *n-tice* konkrétních hodnot z *n* různých sloupců. Řádky se někdy nazývají *záznamy* (records) a sloupce se pak nazývají *pole* (fields).

V naší tabulce doménu pro první sloupec představuje množina čísel, kterou si zvolíme, např. množina všech možných čísel zboží 1 až 9999999. Jinými slovy, množina všech kladných celých čísel menších nebo rovných 9999999. Doména druhého sloupce je množina všech možných jednotkových cen. Tu už neumíme tak přesně určit jako první. Nevíme například předem, jakých maximálních hodnot může jednotková cena nabývat: stovek nebo miliard korun? Ještě horší je to s určením třetí domény jako množiny všech možných názvů zboží. Přes tyto potíže můžeme stanovit podmnožiny těchto těžko zachytitelných domén a v každém řádku zvolit konkrétní hodnoty, které do nich patří.

Slovo „relační“ se týká speciálního druhu organizace dat a je odvozeno od matematického pojmu *relace*. V matematické teorii množin se relací nazývá podmnožina kartézského součinu množin. Kupříkladu kartézský součin tří množin A_1, A_2, A_3 je množina všech uspořádaných trojic tvaru (a_1, a_2, a_3) , kde a_1 je prvek množiny A_1 , a_2 je prvek množiny A_2 , a_3 je prvek množiny A_3 .

Kartézský součin našich tří domén je nepředstavitelně obrovský počet trojic (tabulkových řádků). Ze zřejmých důvodů tak mluvíme o podmnožině kartézského součinu neboli podmnožině uspořádaných trojic hodnot, které jsou čerpány z jednotlivých obrovských domén. V našem příkladu jsou v tabulce (relaci) jen dva řádky (dvě uspořádané trojice). V praxi bývají řádků statisíce i více. Sloupců bývá zpravidla mnohem více než tři (běžně desítky). Protože podmnožiny kartézských součinů se v matematice nazývají relace, vznikl název *relační databáze*.

V průběhu doby se však výklad slova relace pozmenil tak, že označuje vztah mezi dvěma nebo více databázovými tabulkami.

V souvislosti s relační databází se slovem *databáze* v praxi označují nejméně tři různé věci. Databázemi se nazývají řídicí systémy báze dat – SŘBD (Data Base Management System – DBMS), implementované pod různými jmény a značkami na různých platformách (počítačích a operačních systémech). Databází se někdy rozumí celá množina dat ovládaná pomocí řídicího

systému báze dat. Konečně se databáze říká jednotlivé databázové *tabulce* či databázovému *souboru*.

U relačních databází se uplatňuje dvojí názvosloví: SQL a souborové. Z nich jen několik nejdůležitějších pojmu:

Názvosloví SQL	Názvosloví souborové	Význam názvu
schema	knihovna (library)	knihovna obsahující SQL objekty
tabulka (table)	fyzický soubor (physical file)	množina dat uspořádaná do řádků a sloupců (podmnožina kartézského součinu)
sloupec (column)	pole (field)	množina hodnot stejného typu
řádek (row)	záznam, věta (record)	n-tice hodnot z různých sloupců
primární klíč (primary key)	unikátní klíč (unique key)	sloupec (sloupce) určující jednoznačně řádek tabulky
index	klíč (key) fyzického nebo logického souboru	způsob seřazení řádků podle daného kritéria
pohled (view) výběr sloupců	logický soubor (logical file) výběr polí	podmnožina tabulky se sloupci vybranými z n-tice sloupců
výběr (selection) řádků	logický soubor (logical file) výběr záznamů	podmnožina tabulky s řádky vybranými podle daného kritéria
spojení (join) tabulek	logický soubor (logical file) spojení souborů	tabulka vzniklá spojením sloupců a řádků několika tabulek podle daného kritéria

Jazyk SQL

Jazyk SQL vznikl v laboratořích firmy IBM v 70. letech 20. století zprvu pod názvem SEQUEL (Structured English QUEry Language) a byl použit v souvislosti s prvním relačním databázovým systémem System R firmy IBM. Později byl kvůli konfliktu s dříve registrovaným jménem přejmenován na SQL (Structured Query Language - strukturovaný dotazovací jazyk) a byl několikrát standardizován v organizacích ANSI a ISO.

Přestože existují mezinárodní normy, nejsou realizace jazyka SQL na různých platformách jednotné. Každá sice dodržuje určitou část normy, ale nabízí další, vlastní rozšíření jazyka, což je způsobeno bojem o udržení dosavadních zákazníků. Zde budeme probírat verzi SQL pro systém IBM i (dříve AS/400 aj.) odpovídající normě ISO 2011 Core.

V jazyku SQL se tradičně rozlišují příkazy pro definici dat (DDL - Data Definition Language), příkazy pro manipulaci s daty (DML - Data Manipulation Language) a dynamické příkazy (Dynamic SQL Statements). Kromě nich existují další "různé" příkazy. V normě ISO / ANSI se však kategorizace příkazů liší. Příkazy SQL se nazývají v angličtině *SQL statements*.

Relační databáze v systému IBM i

Relační databáze jakožto řídicí systém i jako množina dat je v jednom (lokálním) systému IBM i *jen jedna*. Je to souhrn všech objektů sloužících k manipulaci s relačními daty: tabulky (fyzické soubory), pohledy (logické soubory bez klíče), indexy (logické soubory s klíčem) aj. Objekty se

v názvosloví SQL seskupují do tzv. *schemat* zvaných též *collections*, což jsou vlastně knihovny IBM i.

V systému IBM i se o zpracování relační databáze stará databázový manažer, což jsou systémové programy obsažené v operačním systému a ve vrstvě SLIC (System Licensed Internal Code). Relační databázi (RDB) v místním systému IBM i lze pojmenovat, a to příkazem ADDRDBDIRE (Add RDB Directory Entry). Jméno relační databáze může být použito při vzdáleném použití v síti.

Systémový katalog

Systémový katalog je soustava tabulek a pohledů poskytujících informace o všech databázových objektech systému IBM i. Je obsažen v knihovně QSYS a QSYS2.

Tabulka, řádek, sloupec (table, row, column)

Databázová *tabulka* (*table*) je objekt, který obsahuje uživatelská data uspořádaná do řádků a sloupců. Sloupce (*columns*) představují údaje stejného typu. Řádky (*rows*) se skládají ze sloupců a může jich být 0 až n, kde n může být malé, nebo i velmi veliké číslo.

Databázovou tabulku můžeme schematicky znázornit takto:

	1. sloupec	2. sloupec	3. sloupec	4. sloupec	...
1. řádek	hodnota	hodnota	
2. řádek		
3. řádek	...				
...					

Řádky nejsou v tabulce nijak uspořádány podle velikosti dat, tj. tabulka v sobě neobsahuje žádný index (na rozdíl od fyzického souboru, který může a nemusí obsahovat definici klíče). Tabulka se vytváří příkazem CREATE TABLE.

Index

Index je zvláštní objekt vystavěný na tabulce a má své jméno. Skládá se z položek obsahujících klíč a ukazatel. Klíč je hodnota z určitého sloupce (nebo několika sloupců) v řádku. Je-li klíč složen z několika hodnot, říkáme těmto hodnotám složky. Ukazatel je číslo odpovídajícího řádku. Položek indexu je tolik, kolik je řádků v tabulce, k níž se index vztahuje. Jednotlivé složky klíče mohou být v indexu uspořádány vzestupně nebo sestupně.

Index je v podstatě logický soubor s definicí klíče a má stejnou strukturu polí jako fyzický soubor, na němž je postaven. Index se vytváří příkazem CREATE INDEX.

Pohled (view)

Výběrem řádků a sloupců z tabulky vznikne objekt zvaný pohled (view). Pohled může být vystavěn na základní tabulce, ale i na jiném pohledu. Pohled sám *neobsahuje* uživatelská *data*, ta jsou obsažena jen v základní tabulce, na níž je pohled (přímo či nepřímo) postaven. Pohled se vytváří příkazem CREATE VIEW.

Pohled se podobá logickému souboru bez definice klíče, ale s výběrem datových polí a s výběrem záznamů. Rozdíl je v tom, že podmínka pro výběr řádků může být v pohledu složitější než podmínka pro výběr záznamů v logickém souboru.

Schema (collection)

Tabulky, pohledy, indexy se sdružují do schematu (schema) neboli kolekce (collection). Schemat můžeme v systému IBM i vytvořit více, podle potřeby. Schema může obsahovat tabulky (tables), pohledy (views), indexy, katalog (catalog), žurnál (journal), žurnálové přijímače (journal receivers) aj. Schema se vytváří příkazem CREATE SCHEMA nebo (starším) příkazem CREATE COLLECTION.

Katalog

V každém schematu je samostatný katalog poskytující informace o databázových objektech daného schematu (na rozdíl od systémového katalogu, který zahrnuje celý lokální systém IBM i).

Interakční SQL – průvodce obrazovkami

Účelem této části je ukázat práci s obrazovkami interakčního SQL, ne však probrat všechny možnosti. Podrobnosti příkazů se probírají v dalších kapitolách.

Zahájení

Práci s interakčním SQL začínáme CL příkazem STRSQL z příkazového řádku. Napíšeme STRSQL a stiskneme F4. Objeví se následující obrazovka.

```
Type choices, press Enter.

Commitment control . . . . . *NONE
Naming convention . . . . . *SYS
Statement processing . . . . . *RUN
Library option . . . . . *LIBL
List type . . . . . *ALL
Data refresh . . . . . *ALWAYS
Allow copy data . . . . . *YES
Date format . . . . . *JOB
Date separator character . . . . . *JOB
Time format . . . . . *HMS
Time separator character . . . . . *JOB
Decimal point . . . . . *JOB
Sort sequence . . . . . *JOB
    Library . . . . . *JOB
Language identifier . . . . . *JOB

*NONE, *CHG, *CS, *ALL...
*SYS, *SQL
*RUN, *VLD, *SYN
Name, *LIBL, *USRLIBL...
*ALL, *SQL
*ALWAYS, *FORWARD
*YES, *OPTIMIZE, *NO
*JOB, *USA, *ISO, *EUR...
*JOB, /, ., -, ', *BLANK
*HMS, *USA, *ISO, *EUR, *JIS
*JOB, :, ., , ', *BLANK
*JOB, *PERIOD, *COMMA...
Name, *HEX, *JOB, *JOBRUN...
Name, *LIBL, *CURLIB
*JOB, *JOBRUN...

Bottom

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys

M a           MW           05/03/7
Connected to remote server/host IAASSIST using port 23
```

Zde můžeme zadat parametry relace (session), které se vztahují ke všem následujícím příkazům.

Zde si všimneme jen jednoho.

Naming convention:

*SYS – bude se používat pojmu a značení podle operačního systému,

*SQL – bude se používat pojmu a značení podle SQL.

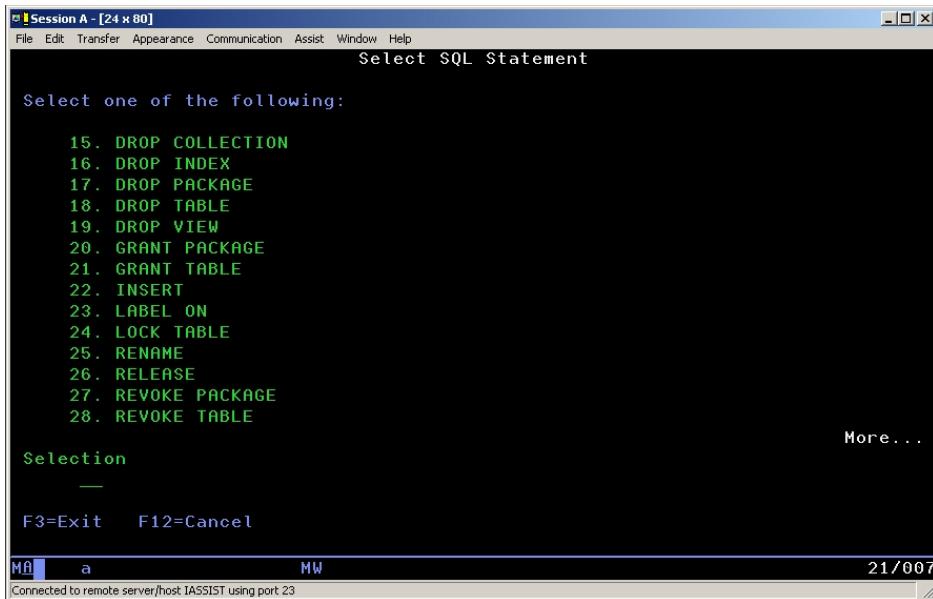
Pro další ukázku si zvolíme hodnotu *SQL a zapíšeme ji do parametru. Po stisku Enter se objeví následující obrazovka.

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Enter SQL Statements
Type SQL statement, press Enter.
Current connection is to relational database S6567B0D.
==>
Bottom
F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys
MA a MW 05/007
Connected to remote server/host IASSIST using port 23

Zde můžeme bud' napsat celý příkaz nebo několik příkazů, jestliže to umíme nazepamět'. Jestliže ne, můžeme stisknout klávesu F4 a získat přehled příkazů.

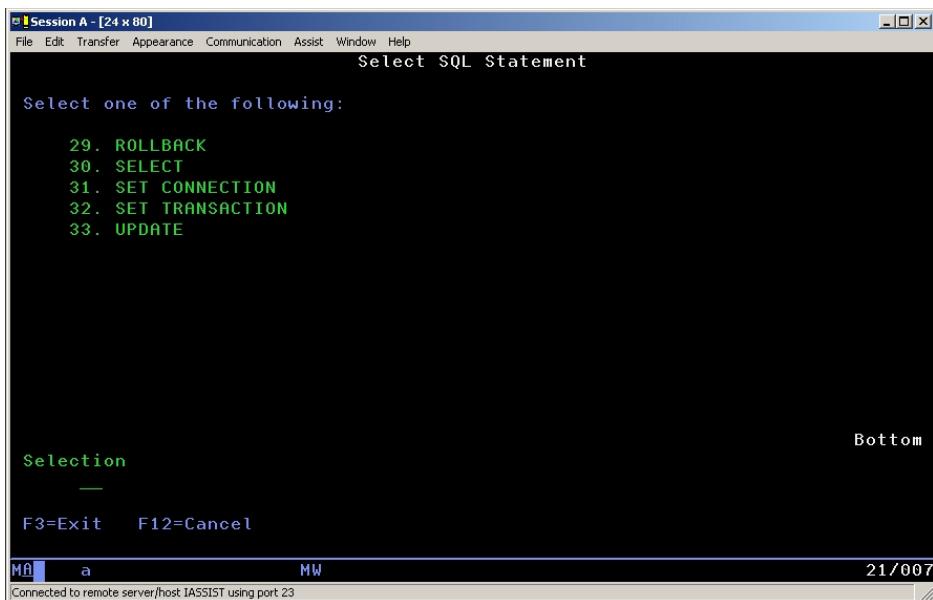
Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Select SQL Statement
Select one of the following:
1. ALTER TABLE
2. CALL
3. COMMENT ON
4. COMMIT
5. CONNECT
6. CREATE ALIAS
7. CREATE COLLECTION
8. CREATE INDEX
9. CREATE PROCEDURE
10. CREATE TABLE
11. CREATE VIEW
12. DELETE
13. DISCONNECT
14. DROP ALIAS
More...
Selection
—
F3=Exit F12=Cancel
MA a MW 21/007
Connected to remote server/host IASSIST using port 23

Na další stránce jsou další příkazy.



Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Select SQL Statement
Select one of the following:
15. DROP COLLECTION
16. DROP INDEX
17. DROP PACKAGE
18. DROP TABLE
19. DROP VIEW
20. GRANT PACKAGE
21. GRANT TABLE
22. INSERT
23. LABEL ON
24. LOCK TABLE
25. RENAME
26. RELEASE
27. REVOKE PACKAGE
28. REVOKE TABLE
More...
Selection
—
F3=Exit F12=Cancel
Ma a MW 21/007
Connected to remote server/host IASSIST using port 23

A ještě na další stránce.



Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Select SQL Statement
Select one of the following:
29. ROLLBACK
30. SELECT
31. SET CONNECTION
32. SET TRANSACTION
33. UPDATE
Bottom
Selection
—
F3=Exit F12=Cancel
Ma a MW 21/007
Connected to remote server/host IASSIST using port 23

Vytvoření a zrušení kolekce

Zadáme volbu 7 – CREATE COLLECTION a dostaneme následující obrazovku.

Napíšeme jméno kolekce (knihovny) a stiskneme Enter. Vytvoří se kolekce KOLEKCE.

Session A - [24 x 80]

File Edit Transfer Appearance Communication Assist Window Help

Enter SQL Statements

Type SQL statement, press Enter.

> CREATE COLLECTION KOLEKCE
Schema KOLEKCE created.

====> _____

Bottom

F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys

MB a MW 06/007

Connected to remote server/host TASSIST using port 23

Kdybychom chtěli kolekci zrušit, můžeme to učinit buď volbou 15 nebo zápisem příkazu DROP COLLECTION KOLEKCE na příkazový řádek.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Enter SQL Statements

Type SQL statement, press Enter.
> CREATE COLLECTION KOLEKCE
Schema KOLEKCE created.
==> DROP COLLECTION KOLEKCE
Bottom

F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys

Ma a MW 06/030
Connected to remote server/host IASSIST using port 23

```

Kolekce se zruší, ale mezitím musíme odpovědět volbou I (Ignore) na zprávu žádající rozhodnutí, zda opravdu zrušit kolekci (když nebyl uložen žurnálový přijímač).

Výsledek rušení se pak objeví na obrazovce.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Enter SQL Statements

Type SQL statement, press Enter.
> CREATE COLLECTION KOLEKCE
Schema KOLEKCE created.
> DROP COLLECTION KOLEKCE
Drop of KOLEKCE in QSYS complete.
==>
Bottom

F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys

Ma a MW 08/007
Connected to remote server/host IASSIST using port 23

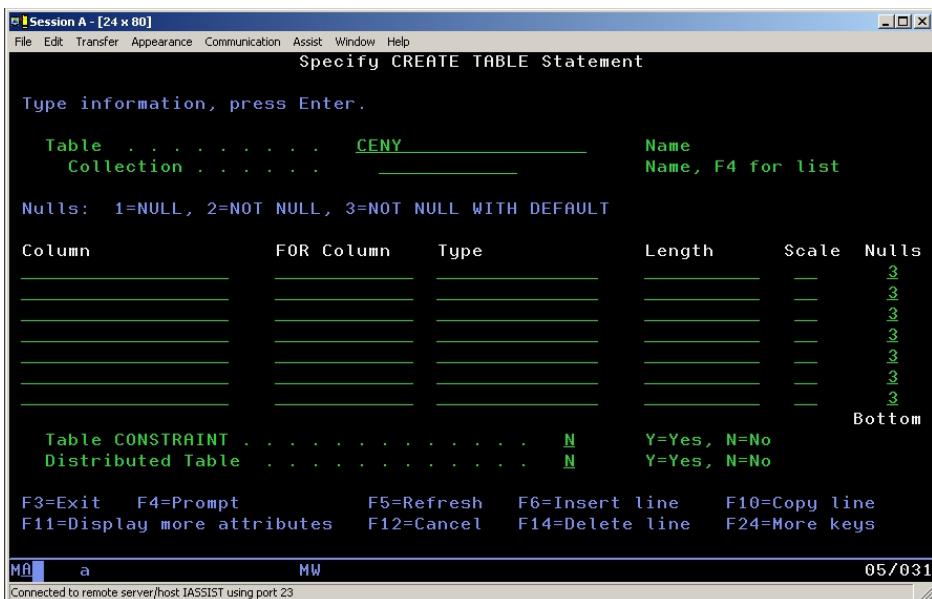
```

Vytvoření tabulky

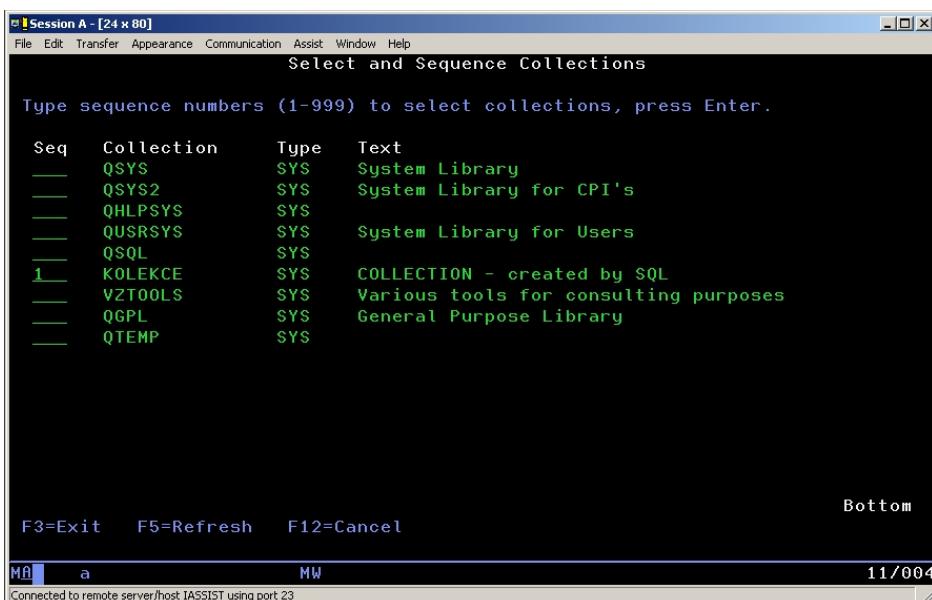
Nyní předpokládejme, že máme vytvořenou kolekci. V ní vytvoříme databázovou tabulku CENY se sloupci CZBOZI (číslo zboží), CENAJ (cena za jednotku) a NAZZBO (název zboží).

CZBOZI	CENAJ	NAZZBO
--------	-------	--------

Zadáme příkaz CREATE TABLE CENY a stiskneme klávesu F4.



Když budeme chtít znát jméno kolekce (nebo, což je totéž, knihovny), můžeme nastavit kurzor na řádek Collection a stisknout F4. Dostaneme seznam knihoven právě platný v naší úloze za předpokladu, že jsme do něj předem zapsali knihovnu KOLEKCE, např. příkazem ADDLIBLE.



Z tohoto seznamu si můžeme vybrat knihovnu tak, že k ní napíšeme do sloupce Seq pořadové číslo 1. (Další pořadová čísla bychom použili u jiného příkazu, např. SELECT.) Pak stiskneme Enter a jméno kolekce se automaticky doplní do předchozí obrazovky.

Na následující obrazovce zadáme zbývající údaje pro definici tabulky CENY, tedy definice sloupců CZBOZI, CENAJ a NAZZBO zároveň s jejich delšími SQL názvy.

The screenshot shows the 'Specify CREATE TABLE Statement' window. It defines a table named 'CENY' under collection 'KOLEKCE'. The table has three columns: 'CISLO_ZBOZI' (CHARACTER, length 5, NOT NULL), 'CENA_ZA_JEDNOTU' (NUMERIC, length 9 scale 2, NOT NULL), and 'NAZEV_ZBOZI' (CHAR, length 30, NOT NULL). The window also shows table constraints and distribution information. F11 is set to 'Display more attributes'.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify CREATE TABLE Statement
Type information, press Enter.

Table . . . . . CENY
Collection . . . . . KOLEKCE Name
Name, F4 for list

Nulls: 1=NULL, 2=NOT NULL, 3=NOT NULL WITH DEFAULT

Column FOR Column Type Length Scale Nulls
CISLO_ZBOZI CZBOZI CHARACTER 5
CENA_ZA_JEDNOTU CENAJ NUMERIC 9,2
NAZEV_ZBOZI NAZZBO CHAR 30
Bottom

Table CONSTRAINT . . . . . . . . . N Y=Yes, N=No
Distributed Table . . . . . . . . . N Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F10=Copy line
F11=Display more attributes F12=Cancel F14=Delete line F24=More keys

MA a MW 13/058
Connected to remote server/host IASSIST using port 23

```

První jméno je jméno sloupce pro SQL, druhé je systémové jméno. Obě jména by měla být jedinečná. Systémové jméno nemusíme zadat, vyhovuje-li první jméno pravidlům tvorby jmen v operačním systému. Jinak bude systémové jméno vygenerováno systémem. Po vyplnění všech údajů pro definici tabulky CENY a stisku Enter dostaneme tuto obrazovku.

The screenshot shows the 'Enter SQL Statements' window. It displays the SQL command used to create the table CENY in the schema KOLEKCE. The table was successfully created, as indicated by the message at the end of the command.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Enter SQL Statements
Type SQL statement, press Enter.
> CREATE TABLE KOLEKCE.CENY (CISLO_ZBOZI FOR COLUMN CZBOZI CHARACTER
(5 ) NOT NULL WITH DEFAULT, CENA_ZA_JEDNOTU FOR COLUMN CENAJ
NUMERIC (9 , 2) NOT NULL WITH DEFAULT, NAZEV_ZBOZI FOR COLUMN
NAZZBO CHAR (30 ) NOT NULL WITH DEFAULT)
Table CENY created in KOLEKCE.
==>
Bottom

F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys

MA a MW 09/007
Connected to remote server/host IASSIST using port 23

```

Tabulka CENY v kolekci (schematu, knihovně) KOLEKCE má nyní definovanou strukturu řádků, ale je prázdná. Abychom ji naplnili, můžeme použít různé prostředky, např. DFU (Data File Utility). My ale použijeme prostředek SQL, tedy příkaz INSERT, abychom si jej ilustrovali.

Zápis dat do tabulky

Na příkazový řádek napíšeme INSERT a stiskneme F4. Dostaneme následující obrazovku.

Session A - [24 x 80]

File Edit Transfer Appearance Communication Assist Window Help

Specify INSERT Statement

Type choices, press Enter.

```
INTO table . . . . . ceny _____ Name, F4 for list
Collection . . . . . kolekce _____ Name, F4 for list
```

Select columns to insert

```
INTO . . . . . N Y=Yes, N=No
```

Insertion method . . . 1 1=Input VALUES
2=Subselect

Type choices, press Enter.

```
WITH isolation level . . 1 1=Current level, 2=NC (NONE)
3=UR (CHG), 4=CS, 5=RS (ALL)
6=RR
```

F3=Exit F4=Prompt F5=Refresh F12=Cancel F20=Display entire name
F21=Display statement

Ma a MW 06/041

Connected to remote server/host IASSIST using port 23

Vyplníme jméno tabulky a kolekce a stiskneme Enter. Dostaneme následující obrazovku.

Session A - [24 x 80]

File Edit Transfer Appearance Communication Assist Window Help

Specify INSERT Statement

Type values to insert, press Enter.

Column	Value
CISLO_ZBOZI	
CENA_ZA_JEDNOTKU	
NAZEV_ZBOZI	

Bottom

F3=Exit F5=Refresh F6=Insert line F10=Copy line F11=Display type
F12=Cancel F14=Delete line F15=Split line F24=More keys

Ma a MW 06/024

Connected to remote server/host IASSIST using port 23

Zde můžeme použít klávesu F11 a zobrazí se typy sloupců.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify INSERT Statement
Type values to insert, press Enter.

Column          Type      Length  Scale  Value
CISLO_ZBOZI    CHARACTER 5        0      '00002'
CENA_ZA_JEDNOTKU  NUMERIC 9      2      5.20
NAZEV_ZBOZI    CHARACTER 30       0      'Zubní pasta Kalodont'

Bottom
F3=Exit   F5=Refresh   F6=Insert line   F10=Copy line   F11=Display nulls
F12=Cancel   F14=Delete line   F15=Split line   F24=More keys
MA a           MW           12/042
Connected to remote server/host IASSIST using port 23

```

Zde již máme bližší informace o vlastnostech sloupců a můžeme vyplnit hodnoty. Všimněme si, že v jednotkové ceně zapisujeme desetinnou tečku, protože přebíráme parametr DECPNT příkazu STRSQL (kde je uvedeno *JOB), tedy z úlohy, kde je standardně zapsána hodnota . (tečka) v parametru DECFMT (což zjistíme příkazem DSPJOB).

Po stisku Enter dostaneme následující obrazovku.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Enter SQL Statements
Type SQL statement, press Enter.
> INSERT INTO KOLEKCE.CENY VALUES('00002', 5.20,
  'Zubní pasta Kalodont')
1 rows inserted in CENY in KOLEKCE.
==>
Bottom
F3=Exit   F4=Prompt   F6=Insert line   F9=Retrieve   F10=Copy line
F12=Cancel   F13=Services   F24=More keys
MA a           MW           07/007
Connected to remote server/host IASSIST using port 23

```

Podařilo se nám vložit do tabulky jeden řádek. Tímto způsobem můžeme pokračovat dále, až pořídíme potřebný počet řádků tabulky.

Poznámka: Praktičtější způsob hromadného pořizování dat je použít program Data File Utility (DFU).

Dotazy příkazem SELECT

Nyní předpokládáme, že máme tabulku CENY naplněnou řádky s čísly, názvy a jednotkovými cenami zboží. Ukážeme si, jak se používá interaktivní příkaz SELECT k jednoduchým dotazům.

Tabulka CENY obsahuje následující řádky.

CISLO_ZBOZI	CENA_ZA_JEDNOTKU	NAZEV_ZBOZI
CZBOZI	CENAJ	NAZZBO
00002	5.20	Zubní pasta Kalodont
00001	8.99	PIŠKOTY OPAVIA
00003	1.25	Prádelní šňůra
00004	10.50	Ponožky pánské tmavé
00005	120.00	Tričko bílé
00006	10.50	Ponožky pánské bílé
00007	1,510.00	Kalhoty manšestrové
00008	1,700.00	Kalhoty džínové
00009	250.00	Whisky Balantine
00010	6,500.00	Koňak Gruzínský
00011	159.00	Taška sportovní
00012	45.00	Pálka pingpongová
00013	360.00	Míč kožený, na odbíjenou
00014	3,500.00	Sako tvídové, nadměr. velikost
00015	12,500.00	Motorové koště
00016	380.00	Slepičí peří na polštáře
00017	130.00	Hrábě dřevěné, kolíky plastové
00018	56.00	Husí sádlo v konzervě
00019	8.50	Rádio malé s kazetákom

Na příkazový řádek napíšeme SELECT a stiskneme F4. Dostaneme tuto obrazovku.

```
Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify SELECT Statement
Type SELECT statement information. Press F4 for a list.

FROM tables . . . . .
SELECT columns . . . . .
WHERE conditions . . . . .
GROUP BY columns . . . . .
HAVING conditions . . . . .
ORDER BY columns . . . . .
FOR UPDATE OF columns . . .

Bottom

Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys
Ma a MW 05/035
Connected to remote server/host IASSIST using port 23
```

Kurzor nastavíme na řádek s FROM tables a stiskneme opět F4. V zobrazeném seznamu knihoven volbou 1 opět zvolíme KOLEKCE. Získáme tak seznam tabulek, z nichž si vybereme CENY. (Do toho řádku ovšem můžeme rovnou zapsat text KOLEKCE.CENY.)

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Select and Sequence Tables

```
Type sequence numbers (1-999) to select tables, press Enter.
```

Seq	Table	Collection	Type	Text
—	CENY	KOLEKCE	TAB	
—	SYSCHKCST	KOLEKCE	VW	SQL catalog view
—	SYSCOLUMNS	KOLEKCE	VW	SQL catalog view
—	SYSCST	KOLEKCE	VW	SQL catalog view
—	SYSCSTCOL	KOLEKCE	VW	SQL catalog view
—	SYSCSTDEP	KOLEKCE	VW	SQL catalog view
—	SYSINDEXES	KOLEKCE	VW	SQL catalog view
—	SYSKEYCST	KOLEKCE	VW	SQL catalog view
—	SYSKEYS	KOLEKCE	VW	SQL catalog view
—	SYSPACKAGE	KOLEKCE	VW	SQL catalog view
—	SYSREFCST	KOLEKCE	VW	SQL catalog view
—	SYSTABDDEP	KOLEKCE	VW	SQL catalog view
—	SYSTABLEDEP	KOLEKCE	VW	SQL catalog view
—	SYSTABLES	KOLEKCE	VW	SQL catalog view
—	SYSTRIGCOL	KOLEKCE	VW	SQL catalog view

More...

F3=Exit F5=Refresh F12=Cancel F16>Select collections
F20=Display entire name

MA a MW 06/002
Connected to remote server/host IASSIST using port 23

K řádku CENY zapíšeme 1 a stiskneme Enter. Do předchozí obrazovky se doplní text KOLEKCE.CENY do řádku FROM tables.

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify SELECT Statement

```
Type SELECT statement information. Press F4 for a list.
```

FROM tables	KOLEKCE.CENY
SELECT columns	—
WHERE conditions	—
GROUP BY columns	—
HAVING conditions	—
ORDER BY columns	—
FOR UPDATE OF columns	—

Bottom

Type choices, press Enter.

DISTINCT rows in result table	N Y=Yes, N=No
UNION with another SELECT	N Y=Yes, N=No
Specify additional options	N Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys

MA a MW 05/048
Connected to remote server/host IASSIST using port 23

Ted' máme určenou tabulku, z níž budeme získávat informace dotazem.

Nastavíme kurzor na řádek SELECT columns a stiskneme F4. Dostaneme náznak všech sloupců, z nichž si očíslováním vybereme dva a určíme jejich pořadí ve výsledné tabulce.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Select and Sequence Columns
Type sequence numbers (1-999) to select columns, press Enter.

Seq Column Table Type Length Scale
1 CISLO_ZBOZI CENY CHARACTER 5
2 CENA_ZA_JEDNOTK > CENY NUMERIC 9 2
3 NAZEV_ZBOZI CENY CHARACTER 30

Bottom
F3=Exit F5=Refresh F11=Display text F12=Cancel F17=Select tables
F19=Display system column names F20=Display entire name

```

Ma a MW 09/003
Connected to remote server/host IASSIST using port 23

Po stisku Enter se do první obrazovky doplní obě jména sloupců.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify SELECT Statement
Type SELECT statement information. Press F4 for a list.

FROM tables . . . . . KOLEKCE.CENY
SELECT columns . . . . . NAZEV_ZBOZI, CENA_ZA_JEDNOTKU
WHERE conditions . . . . .
GROUP BY columns . . . .
HAVING conditions . . . .
ORDER BY columns . . . .
FOR UPDATE OF columns . . .

Bottom
Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

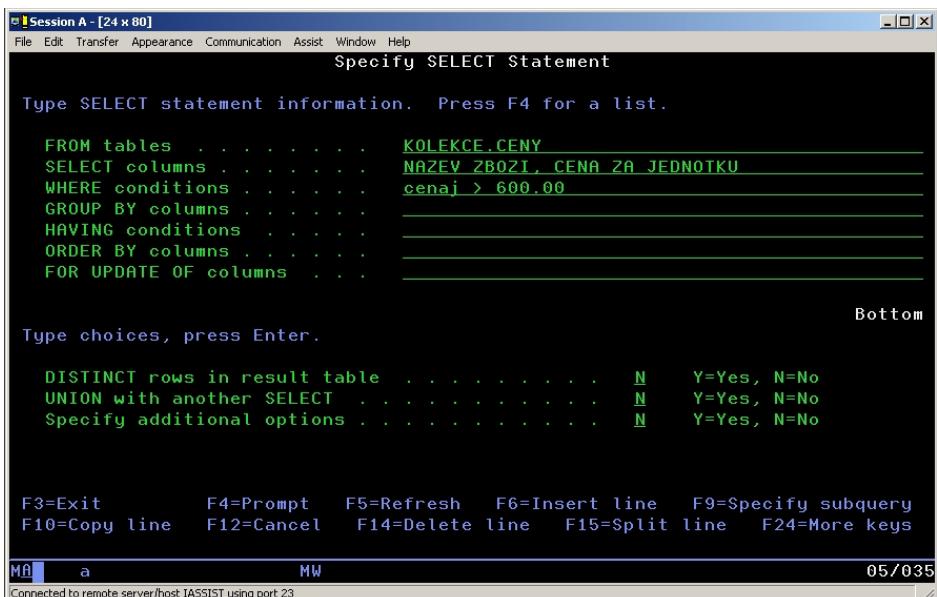
F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys

```

Ma a MW 06/065
Connected to remote server/host IASSIST using port 23

Stejným způsobem pokračujeme dále, dokud uznáme za vhodné.

Údaje můžeme pochopitelně zapisovat přímo do řádků i bez pomoci klávesy F4, známe-li je zpaměti. Například do řádku WHERE conditions napíšeme omezující podmínu, kterou se do výsledné tabulky zařadí jen některé řádky tabulky CENY. Zapíšeme třeba podmínu, že jednotková cena musí být větší než 600.00.



```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify SELECT Statement
Type SELECT statement information. Press F4 for a list.

FROM tables . . . . . KOLEKCE.CENY
SELECT columns . . . . . NAZEV_ZBOZI, CENA_ZA_JEDNOTKU
WHERE conditions . . . . . cenaj > 600.00
GROUP BY columns . . . . .
HAVING conditions . . . . .
ORDER BY columns . . . . .
FOR UPDATE OF columns . . . .

Bottom

Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

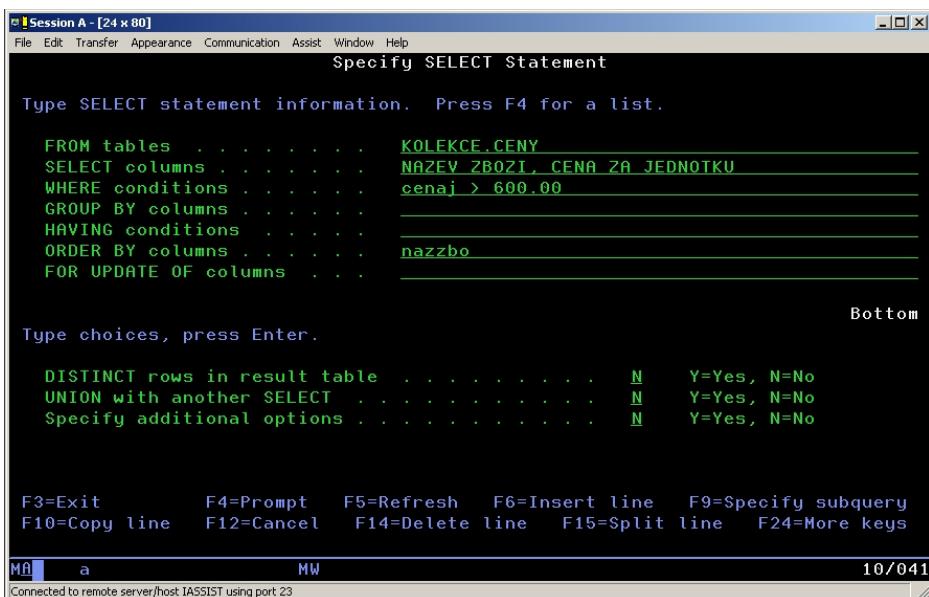
F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys
MW 05/035
Connected to remote server/host IASSIST using port 23

```

Po stisku Enter dostaneme na obrazovce tuto výslednou tabulkou.

NAZEV_ZBOZI	CENA_ZA_JEDNOTKU
Kalhoty manšestrové	1,510.00
Kalhoty džínové	1,700.00
Koňak Gruzínský	6,500.00
Sako tvídové, nadmér. velikost	3,500.00
Motorové koště	12,500.00

Výsledná tabulka je neuspořádaná, protože jsme žádné uspořádání nepředepsali. To bychom mohli udělat vyplněním řádku s ORDER BY columns.



```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Specify SELECT Statement
Type SELECT statement information. Press F4 for a list.

FROM tables . . . . . KOLEKCE.CENY
SELECT columns . . . . . NAZEV_ZBOZI_CENA_ZA_JEDNOTKU
WHERE conditions . . . . . cenaj > 600.00
GROUP BY columns . . . . .
HAVING conditions . . . . .
ORDER BY columns . . . . . nazzbo
FOR UPDATE OF columns . . . .

Bottom

Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys
MW 10/041
Connected to remote server/host IASSIST using port 23

```

Jestliže do řádku ORDER BY columns zapíšeme NAZZBO a stiskneme Enter, dostaneme na obrazovce výslednou tabulkou seřazenou podle názvu zboží.

NAZEV_ZBOZI	CENA_ZA_JEDNOTKU
Kalhoty džínové	1,700.00
Kalhoty manšestrové	1,510.00
Koňak Gruzínský	6,500.00
Motorové koště	12,500.00
Sako tvídové, nadmér. velikost	3,500.00

Když pak stiskneme Enter, zobrazí se příkazový řádek se zprávou a plným tvarem příkazu SELECT.

The screenshot shows a terminal window titled "Session A - [24 x 80]". The title bar includes "File Edit Transfer Appearance Communication Assist Window Help". The main area is labeled "Enter SQL Statements" and contains the following text:

```
Type SQL statement, press Enter.
> SELECT NAZEV_ZBOZI, CENA_ZA_JEDNOTKU FROM KOLEKCE.CENY WHERE cenaj
> < 600 ORDER BY nazboz
SELECT statement run complete.
```

Below the text area, there are several blank lines for input. At the bottom of the window, status information is displayed:

F3=Exit F4=Prompt F6=Insert line F9=Retrieve F10=Copy line
F12=Cancel F13=Services F24=More keys

Bottom

MA a MW 07/007

Connected to remote server/host IASSIST using port 23

Všimněme si, že SQL jména a systémová jména sloupců můžeme libovolně zaměňovat.

Změna atributů relace

Kdykoliv je zobrazena obrazovka Enter SQL Statements, můžeme změnit atributy relace. Stiskem klávesy F13 získáme následující nabídku.

The screenshot shows a terminal window titled "Session A - [24 x 80]". The title bar includes "File Edit Transfer Appearance Communication Assist Window Help". The main area is labeled "Interactive SQL Session Services" and contains the following text:

```
Select one of the following:
```

1. Change session attributes
2. Print current session
3. Remove all entries from current session
4. Save session in source file

Below the menu, there is a section labeled "Selection" followed by a dash. At the bottom of the window, status information is displayed:

F3=Exit F12=Cancel

Bottom

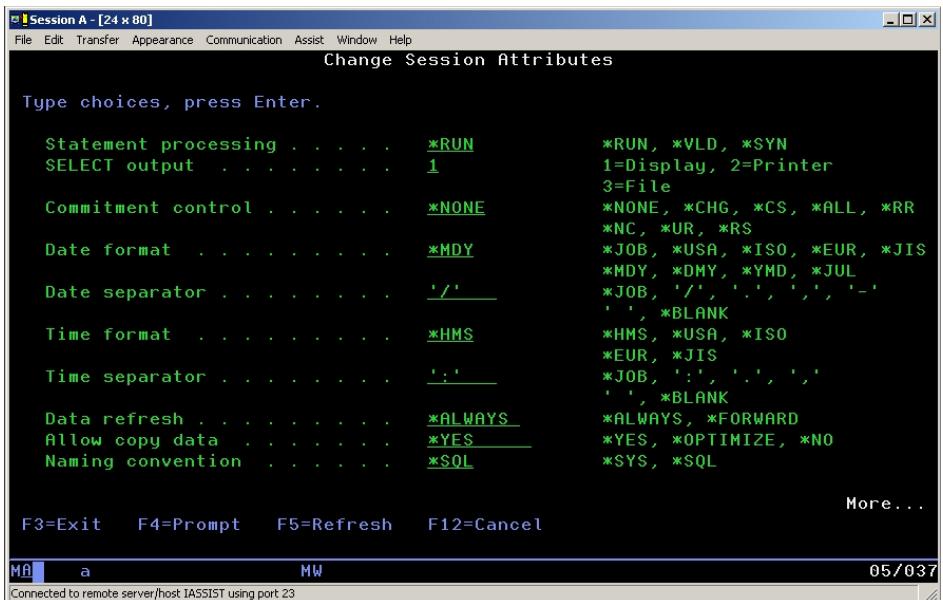
MA a MW 21/007

Connected to remote server/host IASSIST using port 23

Volby 2 a 4 jsou pomocné, registrační. Dovolují zaregistrovat události, které se odehrály při práci v současné relaci (session) v potřebné formě – do tiskového nebo zdrojového souboru. Ve zdrojovém souboru můžeme záznamy editovat, např. vymazat systémové zprávy a ponechat jen příkazy SQL pro pozdější použití, třeba v aplikačním programu.

Volba 3 dovoluje smazat všechny záznamy ze současné relace.

Volbu 1 probereme trochu podrobněji. Vyplníme-li volbu 1, dostaneme nabídku ke změně atributů relace



Podrobněji rozvedeme jen některé z parametrů. Ostatní jsou buď jasné nebo je lze vyhledat v dokumentaci.

Statement processing

*RUN – u příkazu se zkонтroluje syntaxe a věcná správnost a příkaz se provede, je-li správný.

*VLD – u příkazu se zkontaoluje syntaxe a věcná správnost a oznamí se výsledek, např:

```
> CREATE VIEW KOLEKCE.CENY1 AS SELECT * FROM kolekce.ceny WHERE cenaj
> 0
The syntax and validity checks found no errors.
===>
```

*SYN – u příkazu se zkontaoluje jen syntaxe podle pravidel operačního systému; zde však správnost není zárukou, že příkaz bude správně proveden, např:

```
> INSERT INTO *OLEKCE.CENY VALUES('00002', 5,20,
  'Zubní pasta Kalodont')
Token * was not valid. Valid tokens: <IDENTIFIER>.
===> INSERT INTO *OLEKCE.CENY VALUES('00002', 5,20,
  'Zubní pasta Kalodont')
```

SELECT output

1 - výsledek příkazu SELECT se zobrazí na obrazovce,

2 - výsledek příkazu SELECT se vytiskne na tiskárně,

3 - výsledek příkazu SELECT se zapíše do tabulky (souboru), jehož zadání bude vyžádáno.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
Change Session Attributes

Type choices, press Enter.

Collection list . . . . . *LIBL
List type . . . . . *ALL
Decimal point . . . . . *PERIOD
Sort sequence . . . . . *HEX
Language identifier . . . . ENU
SQL rules . . . . . *DB2
Display password . . . . *NO

Name, *LIBL, *USRLIBL
*ALLUSR, *ALL, *CURLIB
*ALL, *SQL
*PERIOD, *COMMA, *JOB,
*SYSVAL
Name, *JOB, *JOBRUN
*LANGIDUNQ, *LANGIDSHR
*HEX
Name, *JOB, *JOBRUN
F4 for Prompt
*DB2, *STD
*NO, *YES

Bottom

F3=Exit F4=Prompt F5=Refresh F12=Cancel
M8 a MW 05/037
Connected to remote server/host IASSIST using port 23

```

Collection list

Name – jméno konkrétní kolekce (knihovny); v náznacích (F4) se budou nabízet objekty z této kolekce,

*LIBL – seznam knihoven úlohy; v náznacích (F4) se bude nabízet nejprve seznam, z něhož lze vybrat jednu nebo více knihoven; z nich se vytvoří seznam objektů, z nichž lze vybrat potřebné.
atd. – seznamy knihoven podle voleb v operačním systému.

List type

*ALL – v náznacích (F4) se zobrazí systémové i SQL objekty,

*SQL – v náznacích (F4) se zobrazí jen SQL objekty.

Vytvoření tabulek pro příklady

Abychom mohli ilustrovat příkazy SQL, upravíme tabulkou CENY a vytvoříme ještě dvě další tabulky, STAVY a OBRATY.

Poznámka: Sloupce mohou mít kromě konkrétních typů typ NULL, tj. žádnou (prázdnou) hodnotu. Fráze WITH DEFAULT určuje předvolenou hodnotu sloupce pro daný typ. Slovo WITH je nepovinné. Fráze NOT NULL znamená, že ve stejné definici sloupce nesmí být zapsána fráze WITH DEFAULT NULL.

Tabulka CENY

CZBOZI	CENAJ	NAZZBO
--------	-------	--------

```
CREATE TABLE KOLEKCE.CENY
( CZBOZI CHAR ( 5 ) NOT NULL WITH DEFAULT,
  CENAJ NUMERIC ( 9, 2 ) NOT NULL WITH DEFAULT,
  NAZZBO CHAR ( 30 ) NOT NULL WITH DEFAULT )
```

Do tabulky vložíme data bud' SQL příkazem INSERT nebo programem DFU (STRDFU).

Tabulka STAVY

ZAVOD	SKLAD	CZBOZI	MNOZSTVI
-------	-------	--------	----------

```
CREATE TABLE KOLEKCE.STAVY
( ZAVOD CHAR ( 2 ) NOT NULL WITH DEFAULT,
  SKLAD CHAR ( 2 ) NOT NULL WITH DEFAULT,
  CZBOZI CHAR ( 5 ) NOT NULL WITH DEFAULT,
  MNOZSTVI NUMERIC ( 9 , 3 ) NOT NULL WITH DEFAULT )
```

Do tabulky vložíme data bud' SQL příkazem INSERT nebo programem DFU (STRDFU).

Tabulka OBRATY

ZAVOD	SKLAD	CZBOZI	MNOBRATU
-------	-------	--------	----------

```
CREATE TABLE KOLEKCE.OBRATY
( ZAVOD CHAR ( 2 ) NOT NULL WITH DEFAULT,
  SKLAD CHAR ( 2 ) NOT NULL WITH DEFAULT,
  CZBOZI CHAR ( 5 ) NOT NULL WITH DEFAULT,
  MNOBRATU NUMERIC ( 9 , 3 ) NOT NULL WITH DEFAULT )
```

Do tabulky vložíme data bud' SQL příkazem INSERT nebo programem DFU (STRDFU).

Schema tabulek a jejich vztahu

Složky primárních klíčů jsou vyznačeny tučným písmem.



Data tabulek

Tabulka CENY

CZBOZI	CENAJ	NAZZBO
00002	459.00	Zubní pasta Kalodont
00001	8.99	PIŠKOTY OPAVIA
00003	1.25	Prádelní šňůra
00004	10.50	Ponožky pánské tmavé
00005	120.00	Tričko bílé
00006	10.55	Ponožky pánské bílé, nové
00007	1510.00	Kalhoty manšestrové
00008	1700.00	Kalhoty džínové
00009	250.00	Whisky Balantine
00010	6500.00	Koňak Gruzínský
00011	159.00	Taška sportovní
00012	45.00	Taška sportovní
00013	360.00	Míč
00014	3500.00	Sako tvídové, nadměr
00015	12500.00	Motorové koště
00016	380.00	Slepičí peří na polštáře
00017	130.00	Hrábě dřevěné, kolíky plastové
00018	56.00	Husí sádlo v konzervě

Tabulka STAVY

ZAVOD	SKLAD	CZBOZI	MNOZSTVI
01	01	00001	1579.444
01	01	00002	1.000
01	01	00010	7.000
01	02	00003	19.000
01	02	00009	1.080
01	02	00010	3.000
01	03	00003	1.000
02	01	00005	2.000
02	01	00006	2.000
02	01	00008	2.000
02	01	00019	2.000
02	02	00009	2.000
02	02	00011	2.000
02	02	00014	2.000
02	02	00018	2.000
02	02	00019	2.000

Tabulka OBRATY

ZAVOD	SKLAD	CZBOZI	MNOBRATU
01	01	00001	789.222
01	02	00003	9.000
01	01	00010	4.000-
01	02	00010	1.000
01	02	00009	.040

Dotazy

K dotazování slouží příkaz SELECT (výběr), který lze použít samostatně (např. v CL příkazu STRSQL) nebo jako součást jiného SQL příkazu. V druhém případě se mu říká *subselect* (*podvýběr*). Příkaz zahrnující dva nebo více podvýběrů spojených frázemi UNION, EXCEPT nebo INTERSECT se nazývá *fullselect* (*plný výběr*). Podvýběry i plné výběry lze vnořovat.

Subselect nebo fullselect uzavřený v závorkách poskytující jediný (nebo žádný) výsledný řádek s jediným sloupcem, se nazývá *scalar subselect* nebo *scalar fullselect*.

Je-li subselect použit v *podmínce* hledání (ve frázích WHERE, JOIN nebo HAVING), nazývá se *subquery* (poddotaz).

Zjednodušený schematický tvar příkazu SELECT (resp. subselect):

```
SELECT sloupce
    FROM odkazy na tabulky
    WHERE podmínka
    CONNECT BY hierarchický dotaz
    GROUP BY sloupce
    HAVING podmínka
    ORDER BY sloupce
    FETCH FIRST počet
```

Jednotlivé složky příkazu se nazývají *clause* (vedlejší věta). Zde je nazýváme *fráze*. Povinné jsou fráze SELECT a FROM, ostatní jsou nepovinné.

Další podrobnosti je třeba hledat v dokumentaci IBM.

V příkladech předpokládáme, že byl uplatněn příkaz

```
SET SCHEMA KOLEKCE
```

který dovoluje vynechat kvalifikaci jménem schematu KOLEKCE.

Příklad nejjednoduššího dotazu je příkaz

```
SELECT * FROM CENY
```

Ten vybere všechny sloupce i řádky z tabulky CENY. Hvězdička znamená seznam všech sloupců v pořadí, jak byly zapsány při vytvoření tabulky.

Párové řádky ze dvou tabulek - WHERE

Dotaz vybere z tabulek STAVY a CENY *všechny* sloupce z tabulky STAVY (hvězdička za tečkou) a sloupec CENAJ z tabulky CENY, přičemž vybírá jen řádky se stejným číslem zboží. Výslednou tabulkou setřídí podle čísla závodu, čísla skladu a čísla zboží.

```
SELECT STAVY.* , CENAJ
FROM STAVY, CENY
    WHERE STAVY.CZBOZI = CENY.CZBOZI
    ORDER BY STAVY.ZAVOD, STAVY.SKLAD, STAVY.CZBOZI
```

ZAVOD	SKLAD	CZBOZI	MNOZSTVI	CENAJ
01	01	00001	1,579.444	8.99
01	01	00002	1.000	459.00
01	01	00010	7.000-	6,500.00
01	02	00003	19.000	1.25
01	02	00009	1.080	250.00
01	02	00010	3.000	6,500.00
01	03	00003	1.000	1.25
02	01	00005	2.000	120.00
02	01	00006	2.000	10.55
02	01	00008	2.000	1,700.00

02	02	00009	2.000	250.00
02	02	00011	2.000	159.00
02	02	00014	2.000	3,500.00
02	02	00018	2.000	56.00

Výběr ze dvou a více tabulek - JOIN

Fráze JOIN je součást fráze FROM.

JOIN

Také INNER JOIN. Vybírá ze dvou tabulek řádky, které splňují párovací kritéria (párují) a spojuje je do řádků výsledné tabulky. Kritérium může mít tvar

ON spojovací-výraz

nebo

USING (sloupec1, sloupec2, ...)

kde sloupce se jmenují stejně v obou tabulkách. Např. příkaz

```
SELECT TAB1.*, TAB2.*
FROM TAB1 JOIN TAB2
    USING (SL1, SL2, SL3)
    dodá totéž jako příkaz
SELECT TAB1.*, TAB2.*
FROM TAB1 JOIN TAB2
    ON TAB1.SL1 = TAB2.SL1 AND
    TAB1.SL2 = TAB2.SL2 AND
    TAB1.SL3 = TAB2.SL3
```

Následující příkaz dává stejný výsledek jako předchozí příklad s frází WHERE.

```
SELECT S.* , C.CENAJ
FROM STAVY S
    JOIN CENY C ON S.CZBOZI = C.CZBOZI
    ORDER BY S.ZAVOD, S.SKLAB, S.CZBOZI
```

Písmena S a C jsou tzv. *korelační jména* tabulek STAVY a CENY. Slouží k rozlišení při označování stejnojmenných sloupců v obou tabulkách, ale také aby bylo zřetelné, ve které z tabulek jsou sloupce definovány. Proto se korelační jména volí krátká, ale přitom co nejurčitější.

Poznámka 1: Místo rovnítka ve frázi ON lze použít i jiný relační operátor, ale to většinou nedává smysl.

Poznámka 2: Stejný dotaz může být zapsán s použitím fráze USING bez korelace sloupců takto:

```
SELECT S.* , C.CENAJ
FROM STAVY S
    JOIN CENY C USING (CZBOZI)
    ORDER BY S.ZAVOD, S.SKLAB, S.CZBOZI
```

Následující příkaz vybere zboží ve stavech, které mělo nějaký obrat.

```
SELECT DISTINCT C.CZBOZI, C.NAZZBO, S.CZBOZI, O.CZBOZI Z_OBRATU
FROM CENY C
    INNER JOIN STAVY S ON C.CZBOZI = S.CZBOZI
    INNER JOIN OBRATY O ON S.CZBOZI = O.CZBOZI
```

CZBOZI	NAZZBO	CZBOZI	Z_OBRATU
00003	Prádelní šňůra	00003	00003
00001	PIŠKOTY OPAVIA	00001	00001
00009	Whisky Balantine	00009	00009
00010	Koňak Gruzínský	00010	00010

Slovo DISTINCT (odlišný) způsobí, že z každé skupiny stejných řádků vybere jen jeden, aby všechny řádky byly odlišné.

LEFT JOIN

Také LEFT OUTER JOIN. Vybírá řádky jako INNER JOIN a přidává řádky z levé tabulky, které nemají v pravé tabulce párový řádek. Sloupce z pravé tabulky mají hodnotu *NULL* označenou pomlčkou.

```
SELECT DISTINCT C.CZBOZI, C.NAZZBO, S.CZBOZI, O.CZBOZI Z_OBRATU
FROM CENY C
    INNER JOIN STAVY S ON C. CZBOZI = S. CZBOZI
    LEFT JOIN OBRATY O ON S. CZBOZI = O. CZBOZI
```

CZBOZI	NAZZBO	CZBOZI	Z_OBRATU
00014	Sako tvídové, nadměr	00014	-
00008	Kalhoty džínové	00008	-
00010	Koňak Gruzínský	00010	00010
00011	Taška sportovní	00011	-
00003	Prádelní šňůra	00003	00003
00006	Ponožky pánské bílé, nové	00006	-
00001	PIŠKOTY OPAVIA	00001	00001
00005	Tričko bílé	00005	-
00009	Whisky Balantine	00009	00009
00018	Husí sádlo v konzervě	00018	-
00002	Zubní pasta Kalodont	00002	-

RIGHT JOIN

Také RIGHT OUTER JOIN. Je jako LEFT (OUTER) JOIN, ale s opačným pořadím tabulek.

```
SELECT DISTINCT C.CZBOZI, C.NAZZBO, S.CZBOZI, O.CZBOZI Z_OBRATU
FROM CENY C
    INNER JOIN STAVY S ON C. CZBOZI = S. CZBOZI
    RIGHT JOIN OBRATY O ON S. CZBOZI = O. CZBOZI
```

CZBOZI	NAZZBO	CZBOZI	Z_OBRATU
00010	Koňak Gruzínský	00010	00010
00003	Prádelní šňůra	00003	00003
00001	PIŠKOTY OPAVIA	00001	00001
00009	Whisky Balantine	00009	00009

FULL OUTER JOIN

Kombinuje LEFT a RIGHT (OUTER) JOIN a přidává nepárové řádky z obou tabulek.

```
SELECT DISTINCT C.CZBOZI, C.NAZZBO, S.CZBOZI, O.CZBOZI Z_OBRATU
FROM CENY C
    INNER JOIN STAVY S ON C. CZBOZI = S. CZBOZI
    FULL JOIN OBRATY O ON S. CZBOZI = O. CZBOZI
```

CZBOZI	NAZZBO	CZBOZI	Z_OBRATU
00006	Ponožky pánské bílé, nové	00006	-
00003	Prádelní šňůra	00003	00003
00008	Kalhoty džínové	00008	-
00001	PIŠKOTY OPAVIA	00001	00001
00011	Taška sportovní	00011	-
00018	Husí sádlo v konzervě	00018	-
00009	Whisky Balantine	00009	00009
00002	Zubní pasta Kalodont	00002	-
00014	Sako tvídové, nadměr	00014	-
00005	Tričko bílé	00005	-
00010	Koňak Gruzínský	00010	00010

EXCEPTION JOIN

Také LEFT EXCEPTION JOIN. Vrací řádky první tabulky, které nepárují s druhou tabulkou.
RIGHT EXCEPTION JOIN vrací řádky druhé tabulky, které nepárují s první tabulkou.

```
SELECT DISTINCT S.CZBOZI, O.CZBOZI  
FROM STAVY S  
    EXCEPTION JOIN OBRATY O  
        ON S.CZBOZI = O.CZBOZI
```

CZBOZI	CZBOZI
00018	-
00005	-
00002	-
00008	-
00011	-
00006	-
00019	-
00014	-

```
SELECT DISTINCT S.CZBOZI, O.CZBOZI  
FROM STAVY S  
    RIGHT EXCEPTION JOIN OBRATY O  
        ON S.CZBOZI = O.CZBOZI
```

CZBOZI	CZBOZI
*****	End of data *****

CROSS JOIN

Kombinuje všechny řádky první tabulky se všemi řádky druhé tabulky (nemá žádné párovací kritérium).

Přejmenované sloupce a pojmenovaný výraz v seznamu sloupců

```
SELECT S.ZAVOD ZAV, S.SKLADEK SKL, S.CZBOZI ZBO,  
    (C.CENAJ * S.MNOZSTVI) AS CELKEM  
FROM STAVY S  
    JOIN CENY C ON S.CZBOZI = C.CZBOZI  
    ORDER BY S.ZAVOD, S.SKLADEK, S.CZBOZI
```

ZAV	SKL	ZBO	CELKEM
01	01	00001	14,199.20156
01	01	00002	459.00000
01	01	00010	45,500.00000-
01	02	00003	23.75000
01	02	00009	270.00000
01	02	00010	19,500.00000
01	03	00003	1.25000
02	01	00005	240.00000
02	01	00006	21.10000
02	01	00008	3,400.00000
02	02	00009	500.00000
02	02	00011	318.00000
02	02	00014	7,000.00000
02	02	00018	112.00000

V seznamu výsledných sloupců je výraz násobení jednotkové ceny zboží množstvím ve skladu, který je pojmenován frází AS jako CELKEM. Závorky i slovo AS lze vynechat. Bez pojmenování výrazu by ve výpisu hlavička obsahovala nadpis Numeric Expression.

V seznamu fráze SELECT lze používat i jiné výrazy, např. také *skalární fullselect* (viz dále). Skalární znamená, že výsledkem výrazu (zde podvýběru) musí být jediná hodnota.

```
SELECT DISTINCT
```

```

( SELECT SUM(S.MNOZSTVI*C.CENAJ)
  FROM STAVY S
  JOIN CENY C USING(CZBOZI)
 ) CENA_CELKEM
FROM STAVY

```

```

CENA_CELKEM
544.30156

```

Poddotaz (subquery)

Poddotaz je příkaz SELECT uzavřený v závorkách a použitý v podmínce hledání jiného příkazu (SELECT, UPDATE, DELETE). Poddotazy mohou být vnořené. Používají se ve frázi WHERE, JOIN nebo HAVING. Poddotaz může být v podmínkovém výrazu operandem, např.

```
WHERE ( X <= (poddotaz1) OR (poddotaz2) = Y ) AND Z NOT IN (poddotaz3)
```

Výběr jedné hodnoty (skalární subselect)

Chceme zjistit, které položky zboží ve skladech mají větší hodnotu než je průměrná. Poddotaz spočítá průměrnou cenu zboží ze všech skladů a vnější dotaz vybere čísla zboží, u nichž je cena vyšší. Podmínce vyjádřené relačním operátorem (např. $>$) se říká ve vztahu k poddotazu *základní predikát (base predicate)*.

```

SELECT S.CZBOZI, (CENAJ*S.MNOZSTVI) CELKEM
FROM CENY, STAVY S
WHERE CENY.CZBOZI = S.CZBOZI
AND CENAJ*S.MNOZSTVI >
( SELECT AVG(CENAJ*MNOZSTVI)
  FROM CENY, STAVY
  WHERE CENY.CZBOZI = STAVY.CZBOZI
)

```

CZBOZI	CELKEM
00002	459.00000
00001	14,199.20156
00005	240.00000
00008	3,400.00000
00009	270.00000
00009	500.00000
00010	19,500.00000
00011	318.00000
00014	7,000.00000
00018	112.00000

Chceme-li zjistit, které zboží v rámci skladu má větší hodnotu než je průměrná cena ve skladu, použijeme korelaci. V poddotazu se porovnávají čísla závodu a skladu mezi tabulkami definovanými v poddotazu a ve vnějším dotazu pomocí *korelace S*:

```

... STAVY.ZAVOD = S.ZAVOD
... STAVY.SKLAD = S.SKLAD

```

Poddotaz se provádí pro každý zkoumaný řádek vnějšího dotazu zvlášť.

```

SELECT S.ZAVOD, S.SKLAD, C.CZBOZI,
       (CENAJ * S.MNOZSTVI) CELKEM
FROM CENY C, STAVY S
WHERE C.CZBOZI = S.CZBOZI
AND C.CENAJ * S.MNOZSTVI >
( SELECT AVG(CENY.CENAJ * STAVY.MNOZSTVI)
  FROM CENY, STAVY
  WHERE STAVY.ZAVOD = S.ZAVOD
    AND STAVY.SKLAD = S.SKLAD
)

```

```

    AND STAVY.CZBOZI = CENY.CZBOZI
)
ORDER BY S.ZAVOD, S.SKLAD, S.CZBOZI

```

ZAVOD	SKLAD	CZBOZI	CELKEM
01	01	00001	14,199.20156
01	01	00002	459.00000
01	02	00010	19,500.00000
02	01	00008	3,400.00000
02	02	00014	7,000.00000

Pro porovnání výsledků je zde seznam skladů s cenami zboží.

ZAVOD	SKLAD	CZBOZI	CELKEM	
01	01	00001	14,199.20156	sklad 01 01
01	01	00002	459.00000	
01	01	00010	45,500.00000-	
<hr/>				
01	02	00003	23.75000	sklad 01 02
01	02	00009	270.00000	
01	02	00010	19,500.00000	
<hr/>				
01	03	00003	1.25000	sklad 01 03 byl vynechán
<hr/>				
02	01	00005	240.00000	sklad 02 01
02	01	00006	21.10000	
02	01	00008	3,400.00000	
<hr/>				
02	02	00009	500.00000	sklad 02 02
02	02	00011	318.00000	
02	02	00014	7,000.00000	
02	02	00018	112.00000	

Sklad 01 03 byl z výsledků vynechán, protože obsahuje jen jeden řádek, a ten nemůže být větší než je sám. Je sám sobě průměrem. Bylo by asi lepší volit relaci “větší nebo rovno”. Pak by byl výsledek

ZAVOD	SKLAD	CZBOZI	CELKEM
01	01	00001	14,199.20156
01	01	00002	459.00000
01	02	00010	19,500.00000
01	03	00003	1.25000
02	01	00008	3,400.00000
02	02	00014	7,000.00000

Výběr množiny hodnot

Predikát **EXISTS** zjišťuje, zda poddotaz vybere *jeden nebo více řádků*.

Čísla zboží obsažená v tabulce OBRATY získáme příkazem

```

SELECT C.CZBOZI
FROM CENY C, OBRATY O
WHERE EXISTS
  ( SELECT O.CZBOZI
    FROM OBRATY
    WHERE C.CZBOZI = O.CZBOZI )

```

Totéž ale získáme i jednodušším příkazem

```

SELECT C.CZBOZI
FROM CENY C, OBRATY O
WHERE C.CZBOZI = O.CZBOZI

```

Nyní chceme zjistit čísla a názvy zboží, u nichž *nebyl* žádný obrat. Pro kontrolu nejprve provedeme obrácenou (kladnou) úlohu: seznam zboží, u něhož *byl* obrat.

```
SELECT DISTINCT C.CZBOZI, C.NAZZBO
FROM CENY C, OBRATY O
WHERE C.CZBOZI = O.CZBOZI
```

Dostaneme výsledek

CZBOZI	NAZZBO
00003	Prádelní šňůra
00010	Koňak Gruzínský
00001	PIŠKOTY OPAVIA
00009	Whisky Balantine

Slovo DISTINCT (odlišný) způsobí, že z každé skupiny stejných řádků vybere jen jeden, aby všechny řádky byly odlišné.

Původní (zápornou) úlohu zkusíme vyřešit záměnou rovnítka za nerovnítko ($<>$) ve frázi WHERE. To však nevede k cíli, dostaneme totiž seznam *všeho* zboží z tabulky CENY. Systém SQL porovnává postupně každý řádek první tabulky se všemi řádky druhé tabulky. Pokaždé, když najde nerovnost, je výběrová podmínka splněna a zboží z tabulky CENY se vybere. Jestliže najde rovnost, podmínka není splněna a zboží se nevybere. Problém je v tom, že těch případů nerovnosti je příliš mnoho. Aby ve výsledné tabulce chybělo jen jediné číslo zboží ze všech čísel z tabulky CENY, musely by všechny řádky tabulky OBRATY obsahovat toto číslo.

Řešení spočívá v použití *poddotazu* (příkaz SELECT v závorkách) a *predikátu NOT IN*. Predikát IN zjišťuje, zda operand je *obsažen* v množině hodnot.

```
WHERE CZBOZI IN (SELECT ...)
WHERE CZBOZI NOT IN (SELECT ...)
```

```
SELECT DISTINCT CZBOZI, NAZZBO
FROM CENY
WHERE CZBOZI NOT IN
(SELECT DISTINCT CZBOZI
FROM OBRATY)
ORDER BY CZBOZI
```

Dostaneme výsledek, v němž chybí zboží z kontrolního příkazu, což jsme potřebovali.

CZBOZI	NAZZBO
00002	Zubní pasta Kalodont
00004	Ponožky pánské tmavé
00005	Tričko bílé
00006	Ponožky pánské bílé, nové
00007	Kalhoty manšestrové
00008	Kalhoty džínové
00011	Taška sportovní
00012	Taška sportovní
00013	Míč
00014	Sako tvídové, nadměr
00015	Motorové koště
00016	Slepičí peří na polštáře
00017	Hrábě dřevěné, kolíky plastové
00018	Husí sádlo v konzervě

Poddotaz se provádí nejdřív, jeho výsledkem je seznam všech (odlišných) čísel zboží z tabulky OBRATY:

```
00003
00010
00001
00009
```

Potom se povede vnější příkaz SELECT kde výběrová podmínka zkoumá, zda čísla zboží v řádcích tabulky CENY *nejsou obsažena* (not in) v tomto seznamu.

Predikát IN zkoumá, zda operand na levé straně je obsažen v množině hodnot na pravé straně. Jeho zápor NOT IN naopak zkoumá, zda operand není obsažen v té množině.

Poznámka: Stejný výsledek dostaneme provedením příkazu s frází EXCEPTION JOIN. V této úloze záleží jen na vkusu, který příkaz použijeme.

```
SELECT DISTINCT C.CZBOZI, C.NAZZBO
FROM CENY C
    EXCEPTION JOIN OBRATY O
        ON C.CZBOZI = O.CZBOZI
ORDER BY CZBOZI
```

Hierarchický dotaz

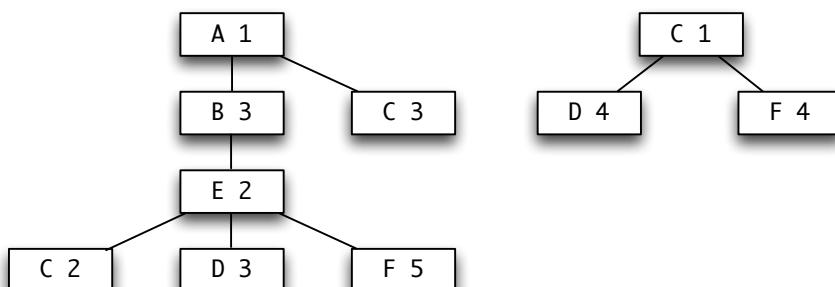
Hierarchický dotaz umožňuje získat informace o hierarchických strukturách jako kusovníky, organizační struktury apod. pomocí fráze CONNECT BY

[START WITH *podmínka*] **CONNECT BY** [NOCYCLE] *podmínka*

Tato tato část poddotazu se zapisuje za podmínkovou frází WHERE, ale vyhodnocuje se ještě před ní. Zpracování probíhá rekurzivně. Předpokládá se, že řádky vstupní tabulky obsahují odkaz (jméno sloupce) na řádek předka (kromě kořene) a řádek následníka (kromě listu) hierarchické struktury. K rozlišení předka a následníka v rekurzivním kroku slouží operátor PRIOR. Slovo NOCYCLE zastaví rekurzi, jestliže je ve struktuře zabudován cyklus (odkazy vedou k řádku již jednou zpracovanému). Podrobnosti je třeba hledat v dokumentaci.

Příklad hierarchického dotazu

Kusovník výrobku A se skládá ze součástí podle schematu:



Výrobek A považujeme za součást sama sebe (kořen). Součásti B, C, E jsou podsoučásti výrobku a součásti D, F jsou koncové díly (listy). Všimněme si, že součást C je ve výrobku A obsažena na dvou místech. Sama se skládá jen ze dvou koncových dílů.

Pro kusovník vytvoříme tabulku SOUCASTI:

```
CREATE TABLE SOUCASTI
( SOUCAST CHAR(2), PODSOUCAST CHAR(2), POSET SMALLINT )
```

Řádky tabulky mají tyto sloupce a hodnoty:

Součást	Podsoučást	Počet
A	B	3
A	C	3
B	E	2
E	C	2
E	D	3
E	F	5
C	D	4
C	F	4

Následujícím příkazem vypíšeme hierarchii kusovníku (rozpad) s uvedením počtu podsoučástí u jednotlivých součástí.

```

SELECT
    SMALLINT(LEVEL) ,
    SOUCAST ,
    PODSOUCAST ,
    SMALLINT(POCET) ,
    CONNECT_BY_ROOT SOUCAST AS KOREN,
    CAST(SYS_CONNECT_BY_PATH(SOUCAST, '-' ) AS VARCHAR(50)) AS VETEV
FROM SOUCASTI
START WITH SOUCAST = 'A'
CONNECT BY SOUCAST = PRIOR PODSOUCAST
ORDER SIBLINGS BY PODSOUCAST

```

Vysvětlivky:

- LEVEL je speciální pseudosloupec vyjadřující úroveň rekurze.
- CONNECT BY ROOT dosadí hodnotu sloupce SOUCAST (kořene) v dané úrovni rekurze určenou ve frázi START WITH.
- SYS_CONNECT_BY_PATH řadí do jednoho sloupce celou cestu (větev) hierarchické struktury od kořenového uzlu. V závorece je jméno výstupního sloupce a oddělovač jeho hodnot ('- ') ve věti – cestě (path). Sloupec je interně definován jako CLOB (1M); v příkazu je zkrácen funkcí CAST na maximálních 50 znaků.
- START WITH určuje řádek se součástí A jako výchozí řádek zpracování (kořen dotazu).
- CONNECT BY určuje následníka podle předchůdce. Hodnota sloupce PODSOUCAST v následujícím (n + 1) kroku rekurze má být rovna hodnotě sloupce SOUCAST předchozího (n) kroku. Je-li takových následníků více, nazývají se sourozenci (siblings).
- ORDER SIBLINGS uspořádá výsledek uvnitř skupiny sourozenců.

Vypíše se :

LEVEL	SOUCAST	PODSOUCAST	POCET	KOREN	VETEV
1	A	B	3	A	- A
2	B	E	2	A	- A - B
3	E	C	2	A	- A - B - E
4	C	D	4	A	- A - B - E - C
4	C	F	4	A	- A - B - E - C
3	E	D	3	A	- A - B - E
3	E	F	5	A	- A - B - E
1	A	C	3	A	- A
2	C	D	4	A	- A - C
2	C	F	4	A	- A - C

Upozornění: V hierarchickém dotazu nelze použít slova GROUP BY, HAVING, ani DISTINCT, protože by se tím narušilo spojování následníků s předky.

Operace s tabulkami (fullselect)

Tabulky lze chápat jako množiny řádků a lze s nimi tedy provádět množinové operace sjednocení (UNION), průnik (INTERSECT) a vyjmutí (EXCEPT). Příkaz SELECT, který obsahuje takové operace, se nazývá **fullselect** (plný výběr). Fullselect, který poskytuje jedinou hodnotu, se nazývá **skalární**.

Operace	Výsledek
TAB1 UNION TAB2	Všechny řádky obou tabulek bez duplicit

Operace	Výsledek
TAB1 UNION ALL TAB2	Všechny řádky obou tabulek s duplicitami
TAB1 INTERSECT TAB2	Řádky společné oběma tabulkám - bez duplicit
TAB1 EXCEPT TAB2	Řádky, které jsou pouze v TAB1 - bez duplicit

Podmínkou je, aby obě tabulky měly stejný počet sloupců.

Sjednocení tabulek s duplicitami

```
SELECT ZAVOD, SKLAD, MNOZSTVI, '1 - STAVY' ID
FROM STAVY
UNION ALL
    SELECT ZAVOD, SKLAD, MNOBRATU, '2 - OBRATY' ID
    FROM OBRATY
ORDER BY ZAVOD, SKLAD, ID
```

ZAVOD	SKLAD	MNOZSTVI	ID
01	01	1,579.444	1 - STAVY
01	01	1.000	1 - STAVY
01	01	7.000-	1 - STAVY
01	01	789.222	2 - OBRATY
01	01	4.000-	2 - OBRATY
01	02	19.000	1 - STAVY
01	02	1.080	1 - STAVY
01	02	3.000	1 - STAVY
01	02	9.000	2 - OBRATY
01	02	1.000	2 - OBRATY
01	02	.040	2 - OBRATY
01	03	1.000	1 - STAVY
02	01	2.000	1 - STAVY
02	01	2.000	1 - STAVY
02	01	2.000	1 - STAVY
02	01	2.000	1 - STAVY
02	02	2.000	1 - STAVY
02	02	2.000	1 - STAVY
02	02	2.000	1 - STAVY

Sjednocení tabulek bez duplicit

```
SELECT ZAVOD, SKLAD, MNOZSTVI, '1 - STAVY' ID
FROM STAVY
UNION
    SELECT ZAVOD, SKLAD, MNOBRATU, '2 - OBRATY' ID
    FROM OBRATY
ORDER BY ZAVOD, SKLAD, ID
```

ZAVOD	SKLAD	MNOZSTVI	ID
01	01	7.000-	1 - STAVY
01	01	1.000	1 - STAVY
01	01	1,579.444	1 - STAVY
01	01	4.000-	2 - OBRATY
01	01	789.222	2 - OBRATY
01	02	1.080	1 - STAVY
01	02	3.000	1 - STAVY
01	02	19.000	1 - STAVY
01	02	.040	2 - OBRATY
01	02	1.000	2 - OBRATY
01	02	9.000	2 - OBRATY
01	03	1.000	1 - STAVY
02	01	2.000	1 - STAVY

Průnik tabulek (bez duplicit)

```
SELECT ZAVOD, SKLAD
FROM STAVY
    INTERSECT
SELECT ZAVOD, SKLAD
FROM OBRATY
ORDER BY ZAVOD, SKLAD

ZAVOD  SKLAD
01      01
01      02
```

Řádky z první tabulky, které nejsou v druhé tabulce

```
SELECT ZAVOD, SKLAD
FROM STAVY
    EXCEPT
SELECT ZAVOD, SKLAD
FROM OBRATY
ORDER BY ZAVOD, SKLAD

ZAVOD  SKLAD
01      03
02      01
02      02
```

Vnořený tabulkový výraz (*nested table expression*)

Ve frázi FROM lze použít podvýběr (subselect) nebo i plný výběr (fullselect) v roli odkazu na tabulku. Takové tabulce se říká *nested table expression* nebo také *derived table*. Vnořený tabulkový výraz může nahradit pohled na tabulku (view).

```
SELECT DISTINCT S.ZAVOD, OBR.POSET, OBR.SOUSET
FROM STAVY S,
     TABLE (SELECT S.ZAVOD, COUNT(*) POSET, SUM(O.MNOBRATU) SOUSET
              FROM OBRATY O
              WHERE S.ZAVOD = O.ZAVOD
        ) AS OBR
WHERE S.ZAVOD = OBR.ZAVOD
```

Vypíše se:

ZAVOD	POSET	SOUSET
01	5	795.262
02	0	-

Příkaz zjišťuje pro každý závod z tabulky STAVY, jaký má počet obratů a v jakém množství (z tabulky OBRATY). Poddotaz vrací tabulku se strukturou sloupců (závod, počet, součet). Součet nemá žádnou hodnotu, resp. má hodnotu *NULL*, protože počet obratů v závodu 02 je nulový a nemá se tedy co sčítat. Slovo TABLE (nebo také LATERAL – postranní) umožňuje odkazovat v podvýběru na tabulku STAVY podle korelátoru S, který je definován na vyšší úrovni. Slovo AS určuje jméno OBR pro tabulku vzniklou podvýběrem.

Společný tabulkový výraz (*common table expression*)

Používá se v příkazu začínajícím slovem WITH, zejména pro rekurzivní zpracování. Tvar příkazu (zjednodušeně):

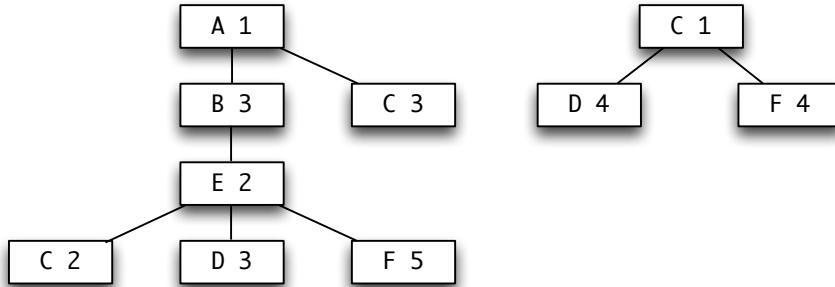
WITH [RECURSIVE] *common-table-expression*, ... *fullselect [order-by] [fetch-first]*

kde *common-table-expression* je:

jméno-tabulky [(sloupec, sloupec, ...)] AS (fullselect [order-by] [fetch-first]) [search] [cycle]

Rekurzivní hledání - kusovník

Kusovník výrobku A se skládá ze součástí podle schematu:



Kusovník je určen tabulkou SOUCASTI, stejnou jako v odstavci o hierarchickém dotazu.

Rozpad kusovníku

Chceme zjistit, ze kterých *pod součástí* (a dílů) se skládá výrobek A (kořenová součást kusovníku). K dotazu použijeme společnou tabulku RS (rekurzivní, společná). Je označena slovem RECURSIVE (nepovinným). Tvoří ji fullselect - dva podvýběry spojené operací UNION ALL. Slovo ALL je povinné. Nedílnou součástí je závěrečný SELECT, který vypisuje výsledky z tabulky RS.

```

WITH RECURSIVE RS (SOUCAST, PODSOUCAST, POCET) AS
( SELECT ROOT.SOUCAST, ROOT.PODSOUCAST, ROOT.POCET
  FROM SOUCASTI ROOT
  WHERE ROOT.SOUCAST = 'A'
UNION ALL
  SELECT CHILD.SOUCAST, CHILD.PODSOUCAST, CHILD.POCET
    FROM RS PARENT, SOUCASTI CHILD
    WHERE PARENT.PODSOUCAST = CHILD.SOUCAST
)
SELECT SOUCAST, PODSOUCAST, POCET
FROM RS
ORDER BY SOUCAST, PODSOUCAST
  
```

SOUCAST	PODSOUCAST	POCET
A	B	3
A	C	3
B	E	2
C	D	4
C	D	4
C	F	4
C	F	4
E	C	2
E	D	3
E	F	5

První podvýběr SELECT ROOT.SOUCAST, ... se provede jednou a vybere do tabulky RS dva řádky pro součást A: AB3, AC3.

Druhý podvýběr SELECT CHILD.SOUCAST, ... (za UNION ALL) už je rekurzivní. Vybírá řádky pomocí dvou tabulek označených **PARENT** (rodič) a **CHILD** (dítě). PARENT čerpá ze společné tabulky RS, zatímco CHILD čerpá z původní tabulky SOUCASTI.

Nejprve se zkoumá první vybraný řádek AB3 z tabulky RS. K podsoučásti B existuje součást E, vybere se tedy řádek BE2. Pak se zkoumá druhý vybraný řádek AC3 z tabulky RS. Podsoučást C obsahuje díly D a F, takže se vyberou řádky CD4 a CF4 a tato větev končí. Testování přechází na již vybraný řádek BE2. K němu patří tři řádky EC2, ED3, EF5, které se vyberou. Testování pokračuje už jen řádkem EC2 (protože ED3 a EF5 jsou koncové). K němu se vyberou dva řádky CD4, CF4, a protože podsoučásti (díly) D a F nemají žádné podřízené součásti, výběr končí.

K sestavení výsledné tabulky RS se připojí řádky z druhého podvýběru operací UNION ALL k řádkům z prvního podvýběru.

Nakonec, příkazem SELECT, se vyberou všechny řádky z tabulky RS a seřadí se podle podsoučásti.

Vynecháme-li z tohoto dotazu sloupec SOUCAST, dostaneme seznam *podsoučástí* výrobku A s jejich počty a s opakováním.

PODSOUCAST	POČET
B	3
C	3
C	2
D	4
D	3
D	4
E	2
F	4
F	5
F	4

Výskyt zvolené podsoučásti v ostatních součástech

Zjistíme, ve kterých součástech se nachází podsoučást (díl) F. Obrátíme role rodiče a dítěte a také pořadí součásti a podsoučásti v dotazu.

```
WITH RECURSIVE RS (PODSOUCAST, SOUCAST) AS
( SELECT ROOT.PODSOUCAST, ROOT.SOUCAST
  FROM SOUCASTI ROOT
 WHERE ROOT.PODSOUCAST = 'F'
UNION ALL
  SELECT PARENT.PODSOUCAST, PARENT.SOUCAST
    FROM RS CHILD, SOUCASTI PARENT
   WHERE PARENT.PODSOUCAST = CHILD.SOUCAST
)
SELECT DISTINCT SOUCAST AS SOUCASTI_S_F
FROM RS
ORDER BY SOUCAST
```

SOUCASTI_S_F
A
B
C
E

Sumář podsoučástí pro výrobek

Na rozdíl od hierarchického dotazu můžeme použít seskupování a sčítání sloupců. Chceme zjistit, kolik kterých *podsoučástí* je třeba pro výrobek A.

```
WITH RS (SOUCAST, PODSOUCAST, POČET) AS
( SELECT ROOT.SOUCAST, ROOT.PODSOUCAST, ROOT.POČET
  FROM SOUCASTI ROOT
 WHERE ROOT.SOUCAST = 'A'
UNION ALL
  SELECT CHILD.SOUCAST, CHILD.PODSOUCAST, PARENT.POČET * CHILD.POČET
    FROM RS PARENT, SOUCASTI CHILD
```

```

        WHERE PARENT.PODSOUCAST = CHILD.SOUCAST
)
SELECT PODSOUCAST, SUM(POCET) CELKEM
FROM RS
GROUP BY PODSOUCAST
ORDER BY PODSOUCAST

```

PODSOUCAST	CELKEM
B	3
C	15
D	78
E	6
F	90

Předchozí příkaz je doplněn o násobení počtu nadřazené součásti počtem podsoučásti a součtem těchto hodnot ve výsledném rádku. Může se zdát, že podsoučásti D a F jsou započítány příliš mnohokrát. Podsoučást C, která je obsahuje ve 4 výskytech, je zařazena na dvou místech. Přímo v A a ještě v B. V A je započítána 3krát. V B (přes E) je započítána 12krát ($2 \times 2 \times 3$). Výskytů podsoučásti C je tedy dohromady 15. Podsoučást D je v C obsažena 4krát, rovněž tak i podsoučást F. U každé z nich je to $15 \times 4 = 60$ výskytů. Když u D připočteme $3 \times 2 \times 3 = 18$ výskytů z E a B, dostaneme 78 výskytů pro D. U F přičteme k 60 výskytům z C ještě $5 \times 2 \times 3 = 30$ výskytů z E a B, dostaneme 90 výskytů pro F.

Průběh výpočtu můžeme pozorovat, když z dotazu odstraníme seskupení a seřazení.

```

WITH RS (SOUCAST, PODSOUCAST, POCET) AS
( SELECT ROOT.SOUCAST, ROOT.PODSOUCAST, ROOT.POSET
    FROM SOUCASTI ROOT
    WHERE ROOT.SOUCAST = 'A'
UNION ALL
    SELECT CHILD.SOUCAST, CHILD.PODSOUCAST, PARENT.POSET * CHILD.POSET
        FROM RS PARENT, SOUCASTI CHILD
        WHERE PARENT.PODSOUCAST = CHILD.SOUCAST
)
SELECT PODSOUCAST, POCET CELKEM
FROM RS

```

PODSOUCAST	CELKEM
B	3
C	3
E	6
D	12
F	12
C	12
D	18
F	30
D	48
F	48

Kontrola zacyklení rekurze

U rekurzivního hledání je nutné zajistit, aby nedošlo k nekonečnému cyklu.

```

WITH RECURSIVE RS (SOUCAST, PODSOUCAST, POCET) AS
( SELECT ROOT.SOUCAST, ROOT.PODSOUCAST, ROOT.POSET
    FROM SOUCASTI ROOT
    WHERE ROOT.SOUCAST = 'B'
UNION ALL
    SELECT CHILD.SOUCAST, CHILD.PODSOUCAST, CHILD.POSET
        FROM RS PARENT, SOUCASTI CHILD
        WHERE PARENT.PODSOUCAST = CHILD.SOUCAST
) CYCLE SOUCAST, PODSOUCAST

```

```

SET CYKLUS TO 'Y' DEFAULT 'N'
SELECT DISTINCT SOUCAST, PODSOUCAST, POCET, CYKLUS
FROM RS
ORDER BY SOUCAST, PODSOUCAST, POCET

```

SOUCAST	PODSOUCAST	POCET	CYKLUS
B	E	2	N
C	D	4	N
C	F	4	N
E	C	2	N
E	D	3	N
E	F	5	N

Přidáme do kusovníku položku, která způsobí zacyklení:

```

Součást Podsoučást Počet
C           B           1

```

Pak bude výsledek

SOUCAST	PODSOUCAST	POCET	CYKLUS
B	E	2	N
B	E	2	Y
C	B	1	N
C	D	4	N
C	F	4	N
E	C	2	N
E	D	3	N
E	F	5	N

Vidíme, že v tabulce je cyklus E - C, C - B, B - E. Kdybychom nekontrolovali zacyklení, hledání by trvalo nekonečně dlouho (dokud bychom neukončili výpočet násilně).

OLAP (On-Line Analytical Processing)

Specifikace OLAP může vystupovat v roli sloupce v seznamu *SELECT* nebo ve frázi *ORDER BY* příkazu *SELECT* a její hodnotou je celé kladné číslo od 1 výše. Specifikace OLAP se někdy nazývá *okenní funkce* (*window function*). Používá se k hodnocení nebo číslování řádků výsledné tabulky.

Hodnocení (ranking)

Číselné hodnocení, které výsledný řádek je přednější, je určeno specifikací **RANK** nebo **DENSE_RANK**.

RANK | DENSE_RANK () OVER ([window-partition] window-order)

DENSERANK

Údaj *window-partition* má tvar

PARTITION BY (sloupec, ...)

kde *sloupec* je výraz, který jednoznačně odkazuje na některý sloupec výsledné tabulky z téhož podvýběru *SELECT*, který obsahuje specifikaci OLAP (tj. **RANK** nebo **DENSE_RANK**).

Údaj *window-order* definuje pořadí řádků použité k výpočtu hodnoty specifikace OLAP. Nedefinuje uspořádání výsledné tabulky. Má tvar

ORDER BY (třídící-klíč ASC | DESC NULL FIRST | NULL LAST , ...)

| (ORDER OF *odkaz-na-tabulku* , ...)

kde

třídící-klíč jednoznačně odkazuje na některý sloupec výsledné tabulky poddotazu se specifikací OLAP.

odkaz-na-tabulkou odkazuje na tabulku z fráze FROM, která sama je vyjádřena jako *nested table expression* nebo *common table expression*. Podvýběr (nebo plný výběr) odpovídající této tabulce musí obsahovat frázi ORDER BY závislou na datech. Uspořádání je pak takové jako kdyby sloupce z vnořeného podvýběru (nebo plného výběru) byly zahrnuty ve vnějším podvýběru (nebo plném výběru) a tyto sloupce by byly zadány místo fráze ORDER OF.

Vybereme zboží z organizačních útvarů a ohodnotíme je (udělíme mu hodnost - rank) podle výše obratu. Hodnost 1 má nejvyšší obrat zboží, protože jsme zadali ASC. Stejně číslo hodnosti má zboží se stejným obratem v různých útvarech. Některé hodnosti jsou přeskočeny, např. trojka: 1, 2, 2, 4, Hodnocení RANK je nesouvislé (řídké, s mezerami), zvané také 1224 podle prvních čtyř hodností. Sloupec RANK obsahuje čísla typu BIGINT (19 dekadických číslic), proto je tak široký.

```
SELECT DISTINCT S.ZAVOD, S.SKLAD, C.CZBOZI, C.CENAJ, O.MNOBRATU,
               O.MNOBRATU * C.CENAJ AS CENA,
               RANK() OVER(ORDER BY O.MNOBRATU * C.CENAJ ASC)
               AS RANK
FROM CENY C
JOIN STAVY S ON C.CZBOZI = S.CZBOZI
JOIN OBRATY O ON S.CZBOZI = O.CZBOZI
ORDER BY RANK
```

ZAVOD	SKLAD	CZBOZI	CENAJ	MNOBRATU	CENA	RANK
01	02	00010	6,500.00	4.000-	26,000.00000-	1
01	01	00010	6,500.00	4.000-	26,000.00000-	1
01	02	00009	250.00	.040	10.00000	3
02	02	00009	250.00	.040	10.00000	3
01	03	00003	1.25	9.000	11.25000	5
01	02	00003	1.25	9.000	11.25000	5
01	02	00010	6,500.00	1.000	6,500.00000	7
01	01	00010	6,500.00	1.000	6,500.00000	7
01	01	00001	8.99	789.222	7,095.10578	9

Další dotaz hodnotí zboží podle hodnocení DENSE_RANK (souvislé, husté). Nazývá se také 1234 podle prvních čtyř hodností.

```
SELECT DISTINCT S.ZAVOD, S.SKLAD, C.CZBOZI, C.CENAJ, O.MNOBRATU,
               O.MNOBRATU * C.CENAJ AS CENA,
               DENSE_RANK() OVER(ORDER BY MNOBRATU DESC) AS RANK
FROM CENY C
JOIN STAVY S ON C.CZBOZI = S.CZBOZI
JOIN OBRATY O ON S.CZBOZI = O.CZBOZI
ORDER BY RANK
```

Hodnost 1 má nejmenší obrat zboží, protože jsme zadali DESC. Stejně číslo hodnosti má zboží se stejným obratem v různých útvarech. Zde je již řada hodnotících čísel souvislá.

ZAVOD	SKLAD	CZBOZI	CENAJ	MNOBRATU	CENA	RANK
01	01	00001	8.99	789.222	7,095.10578	1
01	03	00003	1.25	9.000	11.25000	2
01	02	00003	1.25	9.000	11.25000	2
01	01	00010	6,500.00	1.000	6,500.00000	3
01	02	00010	6,500.00	1.000	6,500.00000	3
01	02	00009	250.00	.040	10.00000	4
02	02	00009	250.00	.040	10.00000	4
01	02	00010	6,500.00	4.000-	26,000.00000-	5
01	01	00010	6,500.00	4.000-	26,000.00000-	5

Číslování řádků

Číslování řádků se zadává OLAP specifikací ROW_NUMBER. Pořadové číslo řádku se vypočítává pro řádek v okně (partition) definovaném v části OVER (PARTITION BY ... ORDER BY ...), počínaje číslem 1 pro první řádek. Není-li v okně zadána (nepovinná) fráze ORDER BY, čísla se

řádkům přiřazují v libovolném pořadí, tak jak jsou vracena z podvýběru (ne podle fráze ORDER BY v příkazu SELECT).

Výsledné číslo je typu BIGINT (19 dekadických číslic).

ROW_NUMBER () OVER ([window-partition] [window-order])

ROWNUMBER

Vybereme číslo závodu, číslo skladu, množství na skladě a k tomu pořadové číslo v rámci závodu (bez ohledu na sklad), protože oknem (partition by zavod) je skupina řádků se stejnou hodnotou sloupce ZAVOD.

```
SELECT ZAVOD, SKLAD, MNOZSTVI,  
       ROW_NUMBER() OVER(PARTITION BY ZAVOD ORDER BY ZAVOD, SKLAD)  
              AS P  
FROM STAVY  
   JOIN CENY USING (CZBOZI)  
   ORDER BY ZAVOD, SKLAD
```

Řádky jsou očíslovány pořadově v rámci každého závodu. Čísla jsou 19místná, proto je sloupec P tak široký.

ZAVOD	SKLAD	MNOZSTVI	P
01	01	1,579.444	1
01	01	1.000	2
01	01	7.000-	3
01	02	19.000	4
01	02	1.080	5
01	02	3.000	6
01	03	1.000	7
02	01	2.000	1
02	01	2.000	2
02	01	2.000	3
02	02	2.000	4
02	02	2.000	5
02	02	2.000	6
02	02	2.000	7

Následující dotaz vybere číslo závodu, číslo skladu, množství a pořadové číslo pro všechny řádky bez ohledu na závod a sklad, protože oknem jsou nyní všechny řádky (prázdný údaj v OVER).

```
SELECT ZAVOD, SKLAD, MNOZSTVI,  
       SMALLINT(ROW_NUMBER() OVER()) AS P  
FROM STAVY  
   JOIN CENY USING (CZBOZI)  
   ORDER BY ZAVOD, SKLAD
```

Konverze čísla BIGINT do SMALLINT (5 dekadických číslic) způsobila zmenšení šířky sloupce P

ZAVOD	SKLAD	MNOZSTVI	P
01	01	1,579.444	1
01	01	1.000	2
01	01	7.000-	3
01	02	19.000	4
01	02	1.080	5
01	02	3.000	6
01	03	1.000	7
02	01	2.000	8
02	01	2.000	9
02	01	2.000	10
02	02	2.000	11
02	02	2.000	12
02	02	2.000	13
02	02	2.000	14

GROUP BY

Fráze GROUP BY dovoluje seskupovat řádky podle hodnot sloupců do skupin a z těchto skupin získat funkce jiných, i odvozených sloupců. Jde o tzv. *agregátní* funkce SUM (součet), AVG (průměr), MAX (maximum), MIN (minimum) aj. Obecný tvar fráze GROUP BY je

GROUP BY *sloupce*

grouping-sets

super-groups

Označení *grouping-sets* (seznamy skupin) odpovídá definici

GROUP BY GROUPING SETS (*sloupce* | *super-groups* | (*sloupce* | *super-groups*), ...)

Označení *super-groups* (nadskupiny) odpovídá definicím

ROLLUP (*sloupce* | (*sloupce*), ...)

CUBE (*sloupce* | (*sloupce*), ...)

()

kde prázdné závorky () znamenají celek (grand total).

GROUP BY - jednoduché seskupování

Následující příkaz seskupí řádky se stejným číslem závodu a čísla zboží (bez ohledu na číslo skladu). Každá skupina představuje jeden záznam ve výsledné tabulce. Z každé skupiny je vypočtena cena zboží (CELKEM).

```
SELECT S.ZAVOD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
FROM STAVY AS S
      INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
      GROUP BY S.ZAVOD, S.CZBOZI
      ORDER BY S.ZAVOD, S.CZBOZI
```

V seznamu výsledných sloupců je aggregátní funkce SUM, která sečte součiny ceny a množství v každé skupině. Výsledek by měl však příliš velký počet číslic spočítaný interně. Proto je funkce SUM ještě transformována *skalární* funkcí DECIMAL s určením 9 číslic, z toho 2 desetinných míst.

ZAVOD	CZBOZI	CELKEM
01	00001	14,199.20
01	00002	459.00
01	00003	25.00
01	00009	270.00
01	00010	26,000.00-
02	00005	240.00
02	00006	21.10
02	00008	3,400.00
02	00009	500.00
02	00011	318.00
02	00014	7,000.00
02	00018	112.00

Bez funkce DECIMAL je řádek širší:

ZAVOD	CZBOZI	CELKEM
01	00001	14,199.20156
01	00002	459.00000
01	00003	25.00000
01	00009	270.00000
01	00010	26,000.00000-
...		

GROUP BY GROUPING SETS - seznamy skupin

Seznamy skupin (závorky v závorkách) slouží ke zjednodušení úloh, které by bylo možné řešit spojením (union) dvou nebo více seskupených řádků do jedné výsledné tabulky:

```
SELECT ...
GROUP BY ...
    UNION
SELECT ...
GROUP BY ...
    UNION
...
...
```

Následující dotaz sestaví a setřídí součty za závody, za sklady, za zboží a celkem jako samostatné skupiny. Pomlčkou je označena prázdná hodnota NULL.

```
SELECT S.ZAVOD, S.SKLAD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
FROM STAVY AS S
    INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
    GROUP BY GROUPING SETS( S.ZAVOD, S.SKLAD, S.CZBOZI, () )
    ORDER BY S.ZAVOD, S.SKLAD, S.CZBOZI
```

ZAVOD	SKLAD	CZBOZI	CELKEM
01	-	-	11,046.79-
02	-	-	11,591.10
-	01	-	27,180.69-
-	02	-	27,723.75
-	03	-	1.25
-	-	00001	14,199.20
-	-	00002	459.00
-	-	00003	25.00
-	-	00005	240.00
-	-	00006	21.10
-	-	00008	3,400.00
-	-	00009	770.00
-	-	00010	26,000.00-
-	-	00011	318.00
-	-	00014	7,000.00
-	-	00018	112.00
-	-	-	544.30

GROUP BY ROLLUP - skupinové úrovně

Následující příklad ilustruje výpočet součtů a podsoučtů pomocí slova ROLLUP. Fráze

GROUP BY ROLLUP(S.ZAVOD, S.SKLAD)

určuje hierarchii součtových skupin. Hodnoty se nemusí jen sčítat, mohou se pořizovat maxima, minima, průměry apod.

Hierarchické součty bez detailů

Příkaz SELECT seskupí řádky se stejným číslem skladu do skupin první úrovně a tyto skupiny seskupí do skupin druhé úrovně – se stejným číslem závodu. Z každé skupiny pořídí součet vypočítané ceny. Skupiny setřídí podle závodu a skladu.

```
SELECT S.ZAVOD, S.SKLAD,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
FROM STAVY AS S
    INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
    GROUP BY ROLLUP( S.ZAVOD, S.SKLAD )
    ORDER BY S.ZAVOD, S.SKLAD
```

Výsledkem je tato tabulka:

ZAVOD	SKLAD	CELKEM
01	01	30,841.79-
01	02	19,793.75
01	03	1.25
01	-	11,046.79-
02	01	3,661.10
02	02	7,930.00
02	-	11,591.10
-	-	544.30

V jednotlivých řádcích jsou součty cen zboží za sklad. V řádku, kde je u čísla závodu 01 místo čísla skladu zapsáno znaménko -, je součet součtů za předchozí sklady neboli součet za závod 01.

Podobně u závodu 02. Poslední řádek je celkový součet (grand total) za všechny závody. Znaménko - (pomlčka) nahrazuje prázdnou (NULL) hodnotu sloupce.

Skupinové součty s detaily

Příkaz SELECT zpracovává tři úrovně součtů – za závod, sklad a číslo zboží s celkovým součtem. Jde vlastně o detailní sestavu se všemi druhy zboží a součty za sklad a závod. Součty za zboží nejsou v pravém slova smyslu součty, poněvadž se skládají z jediného sčítance.

```
SELECT S.ZAVOD, S.SKLAD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
  FROM STAVY AS S
    JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
 GROUP BY ROLLUP( S.ZAVOD, S.SKLAD, S.CZBOZI )
 ORDER BY S.ZAVOD, S.SKLAD, S.CZBOZI
```

ZAVOD	SKLAD	CZBOZI	CELKEM
01	01	00001	14,199.20
01	01	00002	459.00
01	01	00010	45,500.00-
01	01	-	30,841.79-
01	02	00003	23.75
01	02	00009	270.00
01	02	00010	19,500.00
01	02	-	19,793.75
01	03	00003	1.25
01	03	-	1.25
01	-	-	11,046.79-
02	01	00005	240.00
02	01	00006	21.10
02	01	00008	3,400.00
02	01	-	3,661.10
02	02	00009	500.00
02	02	00011	318.00
02	02	00014	7,000.00
02	02	00018	112.00
02	02	-	7,930.00
02	-	-	11,591.10
-	-	-	544.30

Stejný výsledek dostaneme dotazem se seznamem skupin.

```
SELECT S.ZAVOD, S.SKLAD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
  FROM STAVY AS S
    JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
 GROUP BY GROUPING SETS( ( S.ZAVOD, S.SKLAD, S.CZBOZI ),
                         ( S.ZAVOD, S.SKLAD ),
                         ( S.ZAVOD ),
                         ( ) )
 ORDER BY S.ZAVOD, S.SKLAD, S.CZBOZI
```

Skupinové součty bez detailů s vynecháním úrovně

Součtové skupiny ve frázi GROUP BY ROLLUP lze seskupovat závorkami. Například seskupíme sklad a zboží do závorek a dostaneme tak jen dvě skupiny. První skupina bude sčítána za závod a druhá skupina bude sčítána jako celek. Výsledkem bude, že součty za sklad budou vynechány.

```
SELECT S.ZAVOD, S.SKLAD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
  FROM STAVY AS S
  JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
 GROUP BY ROLLUP( S.ZAVOD, ( S.SKLAD, S.CZBOZI ) )
 ORDER BY S.ZAVOD, S.SKLAD, S.CZBOZI
```

ZAVOD	SKLAD	CZBOZI	CELKEM
01	01	00001	14,199.20
01	01	00002	459.00
01	01	00010	45,500.00-
01	02	00003	23.75
01	02	00009	270.00
01	02	00010	19,500.00
01	03	00003	1.25
01	-	-	11,046.79-
02	01	00005	240.00
02	01	00006	21.10
02	01	00008	3,400.00
02	02	00009	500.00
02	02	00011	318.00
02	02	00014	7,000.00
02	02	00018	112.00
02	-	-	11,591.10
-	-	-	544.30

GROUP BY CUBE - skupiny více rozměrů

Slovo cube (krychle) naznačuje seskupování řádků v několika rozměrech. Tři rozměry značí krychli, více rozměrů značí zobecněnou krychli. Napíšeme-li CUBE(a, b), znamená to vytvoření čtyř skupin (a, b), (a), (b) a celkem (). CUBE(a, b, c) vytvoří 8 skupin: (a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), (). Obecně, n rozměrů vytvoří 2^n skupin.

```
SELECT S.ZAVOD, S.SKLAD,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
  FROM STAVY AS S
  INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
 GROUP BY CUBE( S.ZAVOD, S.SKLAD )
 ORDER BY S.ZAVOD, S.SKLAD
```

ZAVOD	SKLAD	CELKEM	
01	01	30,841.79-	závod, sklad
01	02	19,793.75	
01	03	1.25	
01	-	11,046.79-	závod
02	01	3,661.10	
02	02	7,930.00	
02	-	11,591.10	
-	01	27,180.69-	sklad
-	02	27,723.75	
-	03	1.25	
-	-	544.30	grand total

Proti stejnemu dotazu s ROLLUP přibyly zvýrazněné řádky, které tvoří součty za sklad bez ohledu na závod. Seskupení jsou (závod, sklad), (závod), (sklad) a grand total () .

Přidáme-li do skupin ještě číslo zboží, dostaneme 8 seskupení: (závod, sklad, zboží), (závod, sklad), (závod, zboží), (sklad, zboží), (závod), (sklad), (zboží) a grand total () .

```

SELECT S.ZAVOD, S.SKLAD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
FROM STAVY AS S
    INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
GROUP BY CUBE( S.ZAVOD, S.SKLAD, S.CZBOZI )
ORDER BY S.ZAVOD, S.SKLAD, S.CZBOZI

```

ZAVOD	SKLAD	CZBOZI	CELKEM	
01	01	00001	14,199.20	závod, sklad, zboží
01	01	00002	459.00	
01	01	00010	45,500.00-	
01	01	-	30,841.79-	závod, sklad
01	02	00003	23.75	
01	02	00009	270.00	
01	02	00010	19,500.00	
01	02	-	19,793.75	
01	03	00003	1.25	
01	03	-	1.25	
01	-	00001	14,199.20	závod, zboží
01	-	00002	459.00	
01	-	00003	25.00	
01	-	00009	270.00	
01	-	00010	26,000.00-	
01	-	-	11,046.79-	závod
02	01	00005	240.00	
02	01	00006	21.10	
02	01	00008	3,400.00	
02	01	-	3,661.10	
02	02	00009	500.00	
02	02	00011	318.00	
02	02	00014	7,000.00	
02	02	00018	112.00	
02	02	-	7,930.00	
02	-	00005	240.00	
02	-	00006	21.10	
02	-	00008	3,400.00	
02	-	00009	500.00	
02	-	00011	318.00	
02	-	00014	7,000.00	
02	-	00018	112.00	
02	-	-	11,591.10	
-	01	00001	14,199.20	sklad, zboží
-	01	00002	459.00	
-	01	00005	240.00	
-	01	00006	21.10	
-	01	00008	3,400.00	
-	01	00010	45,500.00-	
-	01	-	27,180.69-	sklad
-	02	00003	23.75	
-	02	00009	770.00	
-	02	00010	19,500.00	
-	02	00011	318.00	
-	02	00014	7,000.00	
-	02	00018	112.00	
-	02	-	27,723.75	
-	03	00003	1.25	
-	03	-	1.25	
-	-	00001	14,199.20	zboží
-	-	00002	459.00	
-	-	00003	25.00	
-	-	00005	240.00	
-	-	00006	21.10	
-	-	00008	3,400.00	
-	-	00009	770.00	
-	-	00010	26,000.00-	
-	-	00011	318.00	
-	-	00014	7,000.00	
-	-	00018	112.00	
-	-	-	544.30	grand total

HAVING

Fráze HAVING se používá ve spojitosti s frází GROUP BY pro stanovení dodatečných podmínek pro tvoření skupin řádků.

Skupinové součty s výběrem skupin

Následující dotaz je stejný jako předchozí, ale je v něm doplněna fráze HAVING omezující skupiny tak, aby se tvořily jen z řádků obsahujících určité zboží (čísla mezi 00010 a 00020).

```
SELECT S.ZAVOD, S.CZBOZI,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
  FROM STAVY AS S
    INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
   GROUP BY S.ZAVOD, S.CZBOZI
  HAVING S.CZBOZI BETWEEN '00010' AND '00020'
 ORDER BY S.ZAVOD, S.CZBOZI
```

ZAVOD	CZBOZI	CELKEM
01	00010	26,000.00-
02	00011	318.00
02	00014	7,000.00
02	00018	112.00

Následující dotaz používá frázi GROUP BY ROLLUP. Sčítá a třídí skupiny skladů a závodů. Vybírá však jen skupiny, které předepíšeme frází HAVING, zde jen čísla skladů *různá od 01* a skladы s číslem *NULL*.

```
SELECT S.ZAVOD, S.SKLAD,
       DECIMAL( SUM(C.CENAJ*S.MNOZSTVI), 9, 2 ) AS CELKEM
  FROM STAVY AS S
    INNER JOIN CENY AS C ON S.CZBOZI = C.CZBOZI
   GROUP BY ROLLUP( S.ZAVOD, S.SKLAD )
  HAVING S.SKLAD <> '01' OR S.SKLAD IS NULL
 ORDER BY S.ZAVOD, S.SKLAD
```

ZAVOD	SKLAD	CELKEM
01	02	19,793.75
01	03	1.25
01	-	11,046.79-
02	02	7,930.00
02	-	11,591.10
-	-	544.30

Ve výsledné tabulce chybí řádky s číslem skladu 01 a jsou zachovány řádky s číslem skladu NULL (součet za závod bez skladu 01), tedy i celkový součet.

Kdybychom z podmínky vynechali pozitivní test na NULL, součtové řádky by zmizely a zbyly by jen tři:

ZAVOD	SKLAD	CELKEM
01	02	19,793.75
01	03	1.25
02	02	7,930.00

Zjištování údajů o databázi

Seznam schemat (knihoven) z databáze

```
SELECT varchar (SCHEMA_NAME, 10) schema_name,
       varchar (SCHEMA_OWNER, 10) owner,
```

```

        varchar (SCHEMA_CREATOR, 10) creator,
        CREATION_TIMESTAMP,
        SCHEMA_SIZE size,
        varchar (SCHEMA_TEXT, 20) schema_text,
        varchar (SYSTEM_SCHEMA_NAME, 10) sys_name,
        IASP_NUMBER isp
FROM QSYS2.SYSSCHEMAS
    WHERE substring(SCHEMA_NAME, 1, 1) <> 'Q'
    AND   substring(SCHEMA_NAME, 1, 3) <> 'SYS'

```

SCHEMA_NAME	OWNER	CREATOR	Creation Timestamp	SIZE	SCHEMA_TEXT	SYS_NAME	ISP
#CGULIB	QSYS	QLPINSTALL	2016-07-20-09.22.31.000000	65,536	-	#CGULIB	0
#COBLIB	QSYS	QLPINSTALL	2016-07-18-09.50.07.000000	114,688	-	#COBLIB	0
#DFULIB	QSYS	QLPINSTALL	2016-07-20-09.22.31.000000	65,536	-	#DFULIB	0
#DSULIB	QSYS	QLPINSTALL	2016-07-20-09.22.31.000000	65,536	-	#DSULIB	0
#LIBRARY	QSYS	QLPINSTALL	2016-07-18-09.50.37.000000	73,728	-	#LIBRARY	0
#RPGLIB	QSYS	QLPINSTALL	2016-07-18-09.49.51.000000	98,304	-	#RPGLIB	0
...							

Seznam tabulek (souborů) ve schématu (knihovně)

```

SELECT varchar(TABLE_SCHEMA, 10) schema,
       varchar(TABLE_NAME, 10) table,

```

```

/* Table type: */
CASE TABLE_TYPE
    WHEN 'T' THEN
        CONCAT (TABLE_TYPE, ' = Table')
    WHEN 'P' THEN
        CONCAT (TABLE_TYPE, ' = Physical file')
    WHEN 'L' THEN
        CONCAT (TABLE_TYPE, ' = Logical file')
    WHEN 'V' THEN
        CONCAT (TABLE_TYPE, ' = View')
    WHEN 'A' THEN
        CONCAT (TABLE_TYPE, ' = Alias')
    WHEN 'M' THEN
        CONCAT (TABLE_TYPE, ' = Materialized query table')
    END table_type,
/* varchar(TABLE_TEXT, 20) text, */
/* File type: */
CASE WHEN FILE_TYPE = 'D'
    THEN
        CONCAT (FILE_TYPE, ' = DATA')
    ELSE
        CONCAT (FILE_TYPE, ' = SOURCE')
    END file_type,
    int (COLUMN_COUNT) col_cnt,
    int (ROW_LENGTH) row_len

```

```

FROM QSYS2.SYSTABLES
WHERE TABLE_SCHEMA = 'KOLEKCE'
    AND SUBSTRING(TABLE_NAME, 1, 1) <> 'Q'
    AND SUBSTRING(TABLE_NAME, 1, 3) <> 'SYS'

```

SCHEMA	TABLE	TABLE_TYPE	FILE_TYPE	COL_CNT	ROW_LEN
KOLEKCE	STAVY	P = Physical file	D = DATA	4	18
KOLEKCE	CENY	T = Table	D = DATA	3	44
KOLEKCE	SOUCASTI	T = Table	D = DATA	3	6
KOLEKCE	OBRATY	T = Table	D = DATA	4	18
KOLEKCE	STAVY2	T = Table	D = DATA	4	18

Seznam sloupců tabulky (polí souboru)

```
SELECT
    varchar(TABLE_SCHEMA, 10) table_schema,
    varchar(TABLE_NAME, 10) table_name,
    varchar (DATA_TYPE) data_type,
    varchar (CCSID) ccsid,
    varchar (COLUMN_NAME, 10) column_name,
    int(LENGTH) length,
    int(NUMERIC_SCALE) numeric_scale
FROM QSYS2.SYSCOLUMNS2
WHERE TABLE_NAME = 'STAVY' AND
    TABLE_SCHEMA = 'KOLEKCE'
```

TABLE_SCHEMA	TABLE_NAME	DATA_TYPE	CCSID	COLUMN_NAME	LENGTH	NUMERIC_SCALE
KOLEKCE	STAVY	CHAR	870	ZAVOD	2	-
KOLEKCE	STAVY	CHAR	870	SKLAD	2	-
KOLEKCE	STAVY	CHAR	870	CZBOZI	5	-
KOLEKCE	STAVY	NUMERIC	-	MNOZSTVI	9	3

Vytvoření tabulky příkazem CREATE TABLE

Příkaz CREATE TABLE je definován takto (zjednodušeně):

```
CREATE TABLE jméno-tabulky [ FOR SYSTEM NAME system-identifier ]
    ( definice-sloupce, ... )
        LIKE table|view [ copy-options ]
        unique-constraint
        referential-constraint
        check-constraint

        LIKE table|view [ copy-options ]
        as-result-table [ copy-options ]
    ...
definice sloupce:
jméno-sloupce [ FOR [COLUMN] systémové-jméno ] typ-dat
[  

    [WITH] DEFAULT hodnota
    [GENERATED ALWAYS|BY DEFAULT]
        AS IDENTITY ( START WITH číslo
                        INCREMENT BY číslo
                        NO MINVALUE|NO MAXVALUE|MINVALUE číslo|MAXVALUE číslo
                        NO CYCLE|CYCLE
                        NO CACHE|CACHE 20|CACHE číslo
                        NO ORDER|ORDER
                    )
    [GENERATED ALWAYS|BY DEFAULT]
        FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
    NOT NULL
    [ CONSTRAINT constraint-name ] PRIMARY KEY
        UNIQUE
        CHECK (výraz-hledání)
        REFERENCES tabulka [(sloupec, ...)]
]
unique-constraint:
[ CONSTRAINT jméno ] PRIMARY KEY (sloupec, ...)
    UNIQUE
```

referential-constraint:

```
[ CONSTRAINT constraint-name ] FOREIGN KEY (sloupec, ...)  
    REFERENCES jméno-tabulky (sloupec, ...)  
[ ON DELETE [ NO ACTION ]|RESTRICT|CASCADE|SET NULL|SET DEFAULT ]  
[ ON UPDATE [ NO ACTION ]|RESTRICT ]
```

check-constraint:

```
[ CONSTRAINT constraint-name ] CHECK (výraz-hledání)
```

as-result-table:

```
[ ( seznam sloupců ) ]  
AS ( příkaz-select ) WITH NO DATA | WITH DATA
```

Poznámka: Seznam sloupců musí být zadán, jestliže z *příkazu-select* vyjdou duplicitní jména sloupců nebo nepojmenované sloupce a musí jich být stejný počet jako ve výsledné tabulce.

Změna tabulky příkazem ALTER TABLE

Příkaz ALTER TABLE má několik podob podle druhu změny. Definice příkazu (neúplná):

```
ALTER TABLE druh-změny, druh-změny, ...
```

druh-změny:

```
ALTER [ COLUMN ] SET [DATA TYPE typ-dat] ... (jako u CREATE TABLE)  
ALTER [ DROP DEFAULT|NOT NULL|IDENTITY|ROW CHANGE TIMESTAMP|FIELDPROC ]  
ALTER [ identity-alteration ]  
  
ADD [ COLUMN ] definice-sloupce [ BEFORE jméno-sloupce ]  
ADD unique-constraint  
ADD referential-constraint  
ADD check-constraint  
DROP [ COLUMN ] jméno-sloupce  
DROP PRIMARY KEY  
DROP UNIQUE constraint-name  
DROP FOREIGN KEY constraint-name  
DROP CHECK constraint-name  
DROP CONSTRAINT constraint-name
```

identity-alteration:

```
SET INCREMENT BY číslo  
NO MINVALUE | MINVALUE číslo  
NO MAXVALUE | MAXVALUE číslo  
NO CYCLE | CYCLE  
NO CACHE | CACHE číslo  
NO ORDER | ORDER  
RESTART [ WITH číslo ]
```

Tabulkou STAVY můžeme doplnit o časové razítko dvěma příkazy:

```
ALTER TABLE KOLEKCE.STAVY ADD COLUMN RAZITKO TIMESTAMP  
ALTER completed for table STAVY in KOLEKCE.
```

```
UPDATE STAVY SET RAZITKO = CURRENT_TIMESTAMP  
You are about to alter (DELETE or UPDATE) all of the records in your file(s).
```

```
Press Enter to confirm your statement to alter the entire file.  
Press F12=Cancel to return and cancel your statement.  
16 rows updated in STAVY in KOLEKCE.
```

Po doplnění časového razítka vypíšeme obsah tabulky STAVY:

```
SELECT ZAVOD, SKLAD, CZBOZI, MNOZSTVI, RAZITKO  
FROM STAVY  
ORDER BY ZAVOD, SKLAD, CZBOZI
```

ZAVOD	SKLAD	CZBOZI	MNOZSTVI	RAZITKO
01	01	00001	1,579.444	2022-03-26-13.04.01.567251
01	01	00002	1.000	2022-03-26-13.04.01.567251
01	01	00010	7.000-	2022-03-26-13.04.01.567251
01	02	00003	19.000	2022-03-26-13.04.01.567251
...				

Alternativně můžeme razítko doplnit tak, aby se sloupec RAZITKO změnil na současný čas po každé změně řádku.

```
ALTER TABLE KOLEKCE.STAVY ADD COLUMN RAZITKO TIMESTAMP  
FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP NOT NULL
```

Úpravy tabulek příkazem UPDATE

Příkaz UPDATE dovoluje měnit data v (jediné) tabulce a lze jej zjednodušeně znázornit takto:

```
UPDATE tabulka-nebo-pohled  
SET sloupec = výraz, ...  
WHERE podmínka
```

Změnu dat pomocí UPDATE musíme provádět opatrně, protože je to *hromadná operace*. Jestliže vybereme konkrétní identifikaci jednoho řádku, můžeme použít i podvýběr. Podvýběr musí vyprodukrovat jediný (nebo žádný) řádek.

Chceme opravit množství ve stavech součtem množství odpovídajících obratů. Musíme však volit *konkrétní hodnoty jediného řádku* (konstanty).

Nejprve ověříme výběrové kritérium v poddotazu:

```
SELECT ST.ZAVOD, ST.SKLAD, ST.CZBOZI,  
DEC(SUM(MNOZSTVI), 9,2) PUVODNI_MNOZ,  
DEC(SUM(MNOBRATU), 9,2) OBRAT,  
DEC((AVG(MNOZSTVI) + SUM(MNOBRATU)), 9, 2) NOVE_MNOZ  
FROM STAVY ST, OBRATY O  
WHERE ST.ZAVOD = O.ZAVOD  
AND ST.SKLAD = O.SKLAD  
AND ST.CZBOZI = O.CZBOZI  
AND ST.ZAVOD = '01'  
AND ST.SKLAD = '02'  
AND ST.CZBOZI = '00003'  
GROUP BY ST.ZAVOD, ST.SKLAD, ST.CZBOZI
```

ZAVOD	SKLAD	CZBOZI	PUVODNI_MNOZ	OBRAT	NOVE_MNOZ
01	02	00003	19.00	9.00	28.00

Potom můžeme začlenit tento výběr jako poddotaz s uvedením jediného součtu do příkazu UPDATE:

```
UPDATE STAVY
  SET MNOZSTVI = MNOZSTVI +
  (SELECT SUM(O.MNOBRATU)
   FROM STAVY ST, OBRATY O
   WHERE ST.ZAVOD = O.ZAVOD
   AND ST.SKLAD = O.SKLAD
   AND ST.CZBOZI = O.CZBOZI
   AND ST.ZAVOD = '01'
   AND ST.SKLAD = '02'
   AND ST.CZBOZI = '00003'
   GROUP BY ST.ZAVOD, ST.SKLAD, ST.CZBOZI
  )
WHERE ZAVOD = '01'
  AND SKLAD = '02'
  AND CZBOZI = '00003'
1 rows updated in STAVY in KOLEKCE.
```

Výsledek zkontrolujeme:

```
SELECT * FROM STAVY
WHERE ZAVOD = '01'
  AND SKLAD = '02'
  AND CZBOZI = '00003'
```

ZAVOD	SKLAD	CZBOZI	MNOZSTVI	RAZITKO
01	02	00003	28.000	2022-03-26-13.04.01.567251

Do všech záznamů tabulky můžeme dosadit předvolené hodnoty (v tomto případě nuly) příkazem

UPDATE STAVY SET MNOZSTVI = DEFAULT

nebo dosadíme číslo 0 příkazem

UPDATE STAVY SET MNOZSTVI = 0

Vkládání dat do tabulky

Příkaz INSERT vkládá nové řádky do tabulky.

Vložení jednoho řádku

```
INSERT INTO tabulka-nebo-pohled
  ( sloupec, ... )
VALUES ( hodnota, ... )
```

Vložení několika řádků převzatých z jiné (nebo stejně) tabulky:

```
INSERT INTO tabulka-nebo-pohled
  fullselect
```

Pořadí sloupců může být libovolné:

```
INSERT INTO CENY (CENAJ, NAZZBO, CZBOZI)
VALUES (874.00, 'Tiskárna barevná laserová', '00452')
```

Pořadí hodnot musí odpovídat pořadí sloupců. Seznam sloupců není povinný, ale pak je nutné dodržet pořadí hodnot předepsané při vytvoření tabulky:

```
INSERT INTO CENY
    VALUES ('00453', 874.00, 'Tiskárna barevná laserová')
```

Slovo DEFAULT na místě hodnoty dosadí předvolenou hodnotu předepsanou při vytvoření tabulky, v tomto případě nulu.

```
INSERT INTO CENY
    VALUES ('00453', DEFAULT, 'Tiskárna barevná laserová')
```

Vložení více řádků

Vložíme všechny řádky tabulky OBRATY znovu do stejné tabulky:

```
INSERT INTO OBRATY
    SELECT ZAVOD, SKLAD, CZBOZI, MNOBRATU
    FROM OBRATY
```

Tímto příkazem se řádky v tabulce OBRATY *zdvojí*.

Odstraňování dat z tabulky

Příkaz DELETE dovoluje odstranit data v (jediné) tabulce a lze jej zjednodušeně znázornit takto:

```
DELETE FROM tabulka-nebo-pohled
    WHERE podmínka
```

Změnu dat pomocí DELETE musíme provádět opatrně, s použitím přesné podmínky pro výběr rušených řádků, protože je to *hromadná operace*.

Následující příkaz vymaže všechny řádky tabulky OBRATY.

```
DELETE OBRATY
5 rows deleted from OBRATY in KOLEKCE.
```

Následující příkaz vymaže řádek s číslem zboží 00019.

```
DELETE FROM CENY
    WHERE CZBOZI = '00019'
1 rows deleted from CENY in KOLEKCE.
```

Následující příkaz vymaže všechny řádky souboru STAVY, které obsahují číslo závodu 02.

```
DELETE FROM STAVY
    WHERE ZAVOD = '02'
9 rows deleted from STAVY in KOLEKCE.
```

Kopírování tabulek a dat

Ke kopírování dat lze použít příkaz CREATE TABLE, jestliže potřebujeme vytvořit novou tabulku, nebo příkaz INSERT, jestliže cílová tabulka už existuje.

Vytvoření nové tabulky podle jiné tabulky

Následující příkaz vytvoří novou (prázdnou) tabulku STAVY2 se stejnými sloupci jako má tabulka STAVY bez případných referenčních omezení nebo spouštěčů (triggerů).

```
CREATE TABLE STAVY2 LIKE STAVY
```

Následující příkaz dá stejný výsledek jako předchozí.

```
CREATE TABLE STAVY2
```

```
AS (SELECT * FROM STAVY ) WITH NO DATA
```

Kopírování dat do nově vytvořené tabulky

Následující příkaz vytvoří novou tabulku OBRATY2 podle tabulky OBRATY a zkopíruje do ní data. Stejně jako CL příkaz CPYF s parametrem CRTFILE(*YES) a MBROPT(*REPLACE nebo *ADD).

```
CREATE TABLE OBRATY2  
AS (SELECT * FROM OBRATY ) WITH DATA
```

Kopírování dat do existující tabulky

Následující příkaz zkopíruje data z tabulky OBRATY2 do tabulky OBRATY.

```
INSERT INTO OBRATY  
    SELECT * FROM OBRATY2  
5 rows inserted in OBRATY in KOLEKCE.
```

Obsahuje-li cílová tabulka jiná data, *přidají se data* zdrojové tabulky k nim. Příkaz INSERT *nemůže* existující data *přepsat*. K tomu bychom mohli použít CL příkaz CPYF s parametrem CRTFILE(*NO) a MBROPT(*REPLACE).

Dojde-li při vkládání dat k *porušení integrity dat*, např. jedinečnost klíče, ohlásí se chyba a příkaz se neprovede.

Úprava tabulky podle druhé tabulky - příkaz MERGE

```
MERGE INTO tabulka-nebo-pohled [korelace]  
    USING odkaz-na-tabulku ON podmínka-výběru  
    WHEN [NOT] MATCHED [ AND podmínka ] THEN operace [ ELSE IGNORE ]
```

kde *operace* je UPDATE, DELETE, INSERT.

Následující příkaz aktualizuje množství zboží ve skladu součtem množství obratů.

```
MERGE INTO STAVY S  
    USING (SELECT O.ZAVOD, O.SKLAB, O.CZBOZI,  
              SUM(O.MNOBRATU) SUMA_OBRATU  
        FROM OBRATY O  
        GROUP BY O.ZAVOD, O.SKLAB, O.CZBOZI) AS OBR  
    ON ( S.ZAVOD = OBR.ZAVOD  
      AND S.SKLAB = OBR.SKLAB  
      AND S.CZBOZI = OBR.CZBOZI )  
WHEN MATCHED THEN  
    UPDATE SET S.MNOZSTVI = S.MNOZSTVI + OBR.SUMA_OBRATU  
WHEN NOT MATCHED THEN  
    INSERT (ZAVOD, SKLAB, CZBOZI, MNOZSTVI)  
          VALUES(OBR.ZAVOD, OBR.SKLAB, OBR.CZBOZI, OBR.SUMA_OBRATU)
```

Pohled na tabulky (view)

Pohled (view) je výběr sloupců a řádků z jedné nebo několika tabulek (nebo i pohledů) uložený jako SQL objekt.

Vytváří se příkazem CREATE VIEW:

```
CREATE [ OR REPLACE ] [ RECURSIVE ] VIEW jméno-pohledu  
    FOR SYSTEM NAME jméno ( sloupec [FOR jméno], sloupec [FOR jméno], ... )  
    AS [ WITH common-table-expression, ... ] fullselect
```

Ruší se příkazem

```
DROP VIEW jméno-pohledu
```

Následující příkaz vytvoří pohled na spojené tabulky STAVY a OBRATY sčítané za zboží.

```
CREATE VIEW STAOPR
```

```
( ZAVOD, SKLAD, CZBOZI, MNOZSTVI, OBRAT )
AS SELECT ZAVOD, SKLAD, CZBOZI,
          DEC(SUM(MNOZSTVI),9,2), DEC(SUM(O.MNOBRATU),9,2)
FROM STAVY
JOIN OBRATY O USING( ZAVOD, SKLAD, CZBOZI )
GROUP BY ZAVOD, SKLAD, CZBOZI
```

```
View STAOPR created in KOLEKCE.
```

```
SELECT * FROM STAOPR
```

AVOD	SKLAD	CZBOZI	MNOZSTVI	OBRAT
01	02	00003	28.00	9.00
01	02	00009	1.08	.04
01	01	00010	7.00-	4.00-
01	02	00010	3.00	1.00
01	01	00001	1,579.44	789.22

Následující příkaz vytvoří pohled na spojené tabulky STAVY a OBRATY sčítané za zboží a sklad.

```
CREATE VIEW STAOPR2
```

```
( ZAVOD, SKLAD, MNOZSTVI, OBRAT )
AS SELECT ZAVOD, SKLAD,
          DEC(SUM(MNOZSTVI),9,2), DEC(SUM(O.MNOBRATU),9,2)
FROM STAVY
JOIN OBRATY O USING( ZAVOD, SKLAD )
GROUP BY ZAVOD, SKLAD
```

```
View STAOPR2 created in KOLEKCE.
```

```
SELECT * FROM STAOPR2
```

ZAVOD	SKLAD	MNOZSTVI	OBRAT
01	02	96.24	30.12
01	01	3,146.88	2,355.66

Index

Index je SQL objekt, který tvoří přístupovou cestu k tabulce. Vytváří se příkazem

```
CREATE INDEX jméno-indexu ON tabulka ( výraz-pro-klíč [ ASC | DESC ], ... )
[ WHERE výraz-hledání ]
```

Ruší se příkazem

```
DROP INDEX jméno-indexu
```

Fráze WHERE umožňuje vybrat určité řádky a index tak zkrátit. Výraz hledání nesmí obsahovat poddotaz (subquery) ani položky z výrazů pro klíč.

Příklad:

```
CREATE INDEX OBRATY_IX ON OBRATY ( ZAVOD, SKLAD, CZBOZI )
```

Integrita dat

V tabulkách si můžeme všimnout, že všechny tři mají společný údaj čísla zboží, sloupec CZBOZI a dvě mají společné označení závodu a skladu. Tabulka CENY je prvotní a obsahuje již data. Mlčky jsme předpokládali, že řádky neobsahují stejná čísla zboží.

Když budeme pořizovat data tabulky STAVY, budeme požadovat, aby v každém novém řádku bylo číslo zboží obsaženo také v tabulce CENY a aby v tabulce STAVY nemohly být duplicitní řádky (tj. řádky se stejným závodem, skladem a zbožím).

Když pak konečně budeme pořizovat data tabulky OBRATY (příjmy a výdeje ve skladech), budeme požadovat, aby všechny řádky měly odpovídající řádek v tabulce STAVY, tj. stejný závod, sklad a zboží.

Je patrné, že tabulka OBRATY závisí na tabulce STAVY a ty závisí na tabulce CENY, která je prvotní. Na naše tabulky tedy klademe určité omezující podmínky.

Primární klíč

Tabulka CENY

U tabulky CENY stanovíme podmínu, že sloupec CZBOZI musí jednoznačně určovat řádek tabulky, čili že je *primárním klíčem*. Primární klíč musí jednoznačně definovat řádek, nesmí obsahovat prázdné (null) hodnoty a každý řádek musí mít primární klíč s hodnotou.

```
ALTER TABLE KOLEKCE.CENY ADD PRIMARY KEY (CZBOZI)
```

Když naplníme tabulku ještě před určením primárního klíče, může tento příkaz zhavarovat, protože zjistí *duplicítní klíče*. Duplicítní záznamy (čísla zboží) můžeme spolu s jejich počty vypsat příkazem

```
SELECT CZBOZI, COUNT(*) FROM KOLEKCE.CENY
  GROUP BY KOLEKCE.CENY.CZBOZI
    HAVING COUNT(*) > 1
```

Kvalifikaci jménem kolekce (schematu) můžeme odstranit tak, že nejprve provedeme příkaz

```
SET SCHEMA KOLEKCE
```

a pak můžeme napsat

```
SELECT CZBOZI, COUNT(*) FROM CENY
  GROUP BY CENY.CZBOZI
    HAVING COUNT(*) > 1
```

Duplicítní záznamy odstraníme např. programem DFU nebo SQL příkazem

```
DELETE FROM CENY C WHERE RRN(C) <
(
  SELECT MAX(RRN(C2)) FROM CENY C2
  WHERE C.CZBOZI = C2.CZBOZI
)
```

Tady se používá relativní číslo řádku RRN (relative row number) a odkaz tabulky CENY na sebe samu pomocí tzv. korelačních jmen (*correlation name*), např. C a C2. Důležité je, aby obě tabulky měly různá korelační jména. Ponecháme-li např. první jméno tabulky bez korelace C, vymažou se všechny řádky až na jeden s nejvyšším relativním číslem.

Poznámka: Primární klíč můžeme zrušit příkazem

```
ALTER TABLE KOLEKCE.CENY DROP PRIMARY KEY
```

Tabulka STAVY

U tabulky STAVY stanovíme podmínu, že sloupce ZAVOD, SKLAD, CZBOZI musí jednoznačně určovat řádek tabulky, čili že tvoří *primární klíč*.

```
ALTER TABLE KOLEKCE.STAVY ADD PRIMARY KEY (ZAVOD, SKLAD, CZBOZI)
```

Referenční integrita

Tabulka STAVY

Druhá podmínka u tabulky STAVY je, že sloupec CZBOZI tabulky STAVY je tzv. *cizí klíč (foreign key)*. Každá jeho hodnota musí být přítomna ve sloupci CZBOZI právě jednoho řádku tabulky CENY. Tento sloupce se nazývá *rodičovský klíč (parent key)*. Tabulka CENY se v této souvislosti nazývá *rodičovská tabulka (parent table)*, tabulka STAVY se nazývá *závislá tabulka (dependent table)*.

```
ALTER TABLE KOLEKCE.STAVY ADD FOREIGN KEY (CZBOZI)
    REFERENCES KOLEKCE.CENY (CZBOZI)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
```

Fráze **DELETE RESTRICT** stanoví, že řádek rodičovské tabulky CENY se nesmí vymazat, existuje-li v závislé tabulce STAVY alespoň jeden řádek se stejným číslem zboží (CZBOZI).

U **DELETE** lze místo **RESTRICT** napsat **NO ACTION** (stejně jako **RESTRICT** nebo **nic**), **CASCADE** (zruší všechny závislé řádky v závislé tabulce), **SET NULL** (dosadí prázdné hodnoty - **NULL** do cizích klíčů závislých řádků, jestliže to dotyčný sloupec dovoluje).

Fráze **UPDATE RESTRICT** stanoví, že hodnoty rodičovského klíče CZBOZI v rodičovské tabulce CENY se nesmí měnit (aby v závislé tabulce STAVY nezbyly osiřelé řádky).

Tyto referenční podmínky se uplatňují i při vkládání nových řádků do tabulky příkazem **INSERT**. Např. příkaz

```
INSERT INTO KOLEKCE.STAVY VALUES ('01', '01', '99999', 50000)
```

který vkládá řádek s neexistujícím číslem zboží, způsobí chybové hlášení

```
Operation not allowed by referential constraint
Q_KOLEKCE_STAVY_CZBOZI_00001 in KOLEKCE.
```

Zde si můžeme všimnout, že systém pojmenoval referenční podmínu svým jménem, protože jsme jí neurčili vlastní jméno v příkazu **ALTER TABLE**.

Chceme-li podmínu přejmenovat, musíme ji nejprve zrušit příkazem

```
ALTER TABLE KOLEKCE.STAVY DROP FOREIGN KEY
    KOLEKCE.Q_KOLEKCE_STAVY_CZBOZI_00001 CASCADE
```

(**CASCADE** znamená zrušení všech případných závislých referenčních podmínek.) Pak vytvoříme podmínu znovu pod jiným jménem, např. **STAVY_REF**.

```
ALTER TABLE KOLEKCE.STAVY ADD CONSTRAINT KOLEKCE.STAVY_REF
    FOREIGN KEY (CZBOZI)
    REFERENCES KOLEKCE.CENY (CZBOZI)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
```

Kdyby byly v tabulce STAVY nějaké řádky již dříve a některé neodpovídaly referenčnímu omezení, zjistili bychom je příkazem

```
SELECT * FROM KOLEKCE.STAVY S
    LEFT EXCEPTION JOIN KOLEKCE.CENY C ON S.CZBOZI = C.CZBOZI
    ORDER BY S.CZBOZI
```

a zrušili bychom je příkazem

```
DELETE FROM KOLEKCE.STAVY WHERE CZBOZI IN
(
    SELECT S.CZBOZI FROM KOLEKCE.STAVY S
    LEFT EXCEPTION JOIN KOLEKCE.CENY C ON S.CZBOZI = C.CZBOZI
)
```

Při pokusu o vložení záznamu do tabulky STAVY (pořadí sloupců lze přehodit)

```
INSERT INTO KOLEKCE.STAVY
( CZBOZI, MNOZSTVI, ZAVOD, SKLAD )
VALUES( '99999', 50000, '01', '01' )
```

ohlásí systém chybu

```
Operation not allowed by referential constraint STAVY_REF in KOLEKCE.
```

Poznámka: Referenční omezení lze zrušit příkazem

```
ALTER TABLE KOLEKCE.STAVY DROP FOREIGN KEY
```

Tabulka OBRATY

Podmínka u tabulky OBRATY je, že sloupce ZAVOD, SKLAD, CZBOZI tabulky OBRATY tvoří tzv. *cizí klíč* (*foreign key*). Každá trojice jejich hodnot musí být přítomna ve stejnojmenných sloupcích právě jednoho řádku tabulky STAVY. Tyto tři sloupce se nazývají *rodičovský klíč* (*parent key*). Tabulka STAVY je nyní *rodičovská tabulka* (*parent table*), tabulka OBRATY je *závislá tabulka* (*dependent table*).

```
ALTER TABLE KOLEKCE.OBRATY ADD CONSTRAINT KOLEKCE.OBRATY_REF
FOREIGN KEY (ZAVOD, SKLAD, CZBOZI)
REFERENCES KOLEKCE.STAVY (ZAVOD, SKLAD, CZBOZI)
ON DELETE CASCADE
ON UPDATE NO ACTION
```

Fráze DELETE CASCADE zruší všechny závislé řádky v závislé tabulce.

Fráze UPDATE NO ACTION znamená totéž co UPDATE RESTRICT.

Tyto referenční podmínky se uplatňují i při vkládání nových řádků do tabulky příkazem INSERT. Např. příkaz

```
INSERT INTO KOLEKCE.OBRATY VALUES('01', '99', '00001', 789.222)
```

který vkládá řádek s neexistujícím skladem, způsobí chybové hlášení

```
Operation not allowed by referential constraint OBRATY_REF in KOLEKCE.
```

Poznámka 1: U tabulky OBRATY nepotřebujeme primární klíč, k jedné skladové položce (řádku souboru STAVY) může existovat libovolné množství obratů, tj. záznamů se stejnou hodnotou sloupců ZAVOD, SKLAD, CZBOZI.

Poznámka 2: Podmínky integrity lze kromě příkazu ALTER TABLE zadat i přímo při vytváření tabulky v příkazu CREATE TABLE, např.:

```
CREATE TABLE KOLEKCE.OBRATY
( ZAVOD CHAR (2) NOT NULL,
SKLAD CHAR (2) NOT NULL,
CZBOZI CHAR (5) NOT NULL,
MNOBRATU NUMERIC (9, 3) NOT NULL,
CONSTRAINT KOLEKCE.OBRATY_REF FOREIGN KEY (ZAVOD, SKLAD, CZBOZI)
REFERENCES KOLEKCE.STAVY (ZAVOD, SKLAD, CZBOZI)
ON DELETE CASCADE
```

ON UPDATE NO ACTION)

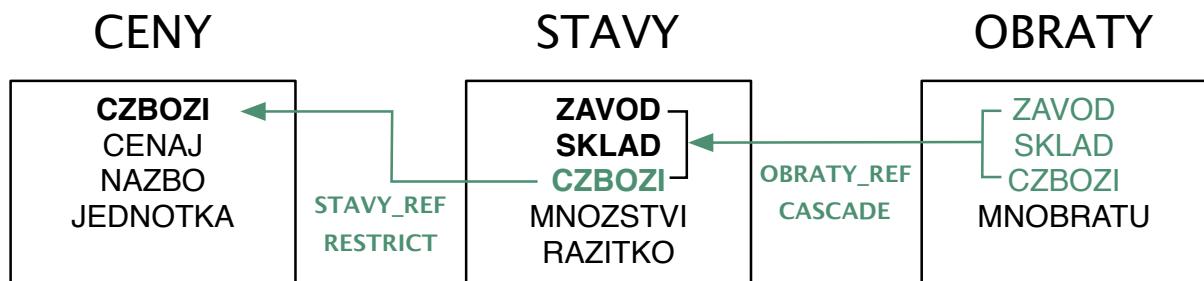
Poznámka 3: Pro uplatnění volby CASCADE je nutné zapnout žurnál, protože se jedná o transakce, které musí proběhnout bez přerušení. Je zapotřebí i potvrzování transakcí (commitment control), ale k provozu stačí spustit jen žurnálování; řízení transakcí není třeba spouštět, zahajuje a ukončuje se automaticky.

Poznámka 4: Referenční omezení lze zrušit příkazem

ALTER TABLE KOLEKCE.OBRATY DROP FOREIGN KEY

Schema referenčních omezení

Zeleně jsou vyznačeny cizí klíče (foreign keys) a šipkami závislost cizích klíčů na primárních klíčích a u nich názvy referenčních omezení (referential constraints).



Typy dat

Hodnoty sloupců jsou data předepsaných typů. Každý sloupec tabulky musí mít definován svůj typ dat. Seznam typů se stručným popisem následuje.

Upozornění. U různých plafórem se jazyk SQL trochu liší. Liší se však i u různých verzí operačního systému, např. v maximálních rozsazích velikosti dat.

INTEGER nebo INT

"Velké" celé číslo (4 bajty, 32 bitů) představující 10 dekadických číslic

SMALLINT

"Malé" celé číslo (2 bajty, 16 bitů) představující 5 dekadických číslic

DECIMAL(n,d) nebo DEC(n,d)

DECIMAL(n) nebo DEC(n) – stejně jako DECIMAL(n,0)

DECIMAL nebo DEC – stejně jako DECIMAL(5,0)

Dekadické pakované číslo. n znamená počet číslic celkem (precision) – 1 až 63, d znamená počet desetinných míst (scale) – 0 až n.

NUMERIC(n,d)

NUMERIC(n) – stejně jako NUMERIC(n,0)

NUMERIC – stejně jako NUMERIC(5,0)

Dekadické zónové (nepakované) číslo. n znamená počet číslic celkem (precision) – 1 až 63, d znamená počet desetinných míst (scale) – 0 až n.

FLOAT

Pohyblivá čárka dvojitě přesnosti.

FLOAT(n)

Pohyblivá čárka jednoduché nebo dvojitě přesnosti podle n. 1 až 25 znamená jednoduchou přesnost, 26 až 53 znamená dvojitou přesnost.

REAL

Pohyblivá čárka jednoduché přesnosti.

DOUBLE PRECISION or DOUBLE

Pohyblivá čárka dvojitě přesnosti.

CHARACTER(n) nebo CHAR(n)

CHARACTER nebo CHAR – stejně jako CHARACTER(1)

Znakový řetězec pevné délky. n je 1 až 32766 (32765, je-li NULL CAPABLE).

VARCHAR(n)

CHARACTER VARYING (n) nebo CHAR VARYING (n)

Znakový řetězec proměnné délky. n je 1 až 32740 (32739, je-li NULL CAPABLE).

DATE

Datum. Má určitý formát (DATFMT) a oddělovač (DATSEP). Určuje se CL příkazem STRSQL (popř. CRTSQLRPGI nebo RUNSQLSTM). Formáty datumu jsou uvedeny v následující tabulce.

DATFMT	Tvar data	DATSEP	Příklad konstanty
*ISO	yyyy-mm-dd		'2000-12-31'
*USA	mm/dd/yyyy		'12/31/2000'
*EUR	dd.mm.yyyy		'31.12.2000'
*JIS	yyyy-mm-dd		'2000-12-31'
*MDY	mm/dd/yy	/ - . , mezera	'12/31/00'
*DMY	dd/mm/yy	/ - . , mezera	'31 12 00'
*YMD	yy/mm/dd	/ - . , mezera	'00.12.31'
*JUL	yy/ddd	/ - . , mezera	'00-366'
*JOB	*MDY nebo *DMY nebo *YMD nebo *JUL	/ - . , mezera (viz DSPJOB)	'31/12/00'

TIME

Čas. Má určitý formát (TIMFMT) a oddělovač (TIMSEP). Určuje se příkazem STRSQL (popř. CRTSQLRPGI nebo RUNSQLSTM). Formáty času jsou uvedeny v následující tabulce.

TIMFMT	Tvar času	TIMSEP	Příklad konstanty
*HMS	hh:mm:ss	: . , mezera *JOB (viz DSPJOB)	'23:59:59'
*ISO	hh.mm.ss		'23.59.59'
*EUR	hh.mm.ss		'23.59.59'
*USA	hh:mm AM hh:mm PM		'00:00 AM '11:59 PM'
*JIS	hh:mm:ss		'23:59:59'

TIMESTAMP

Časové razítko. Má předepsaný tvar s oddělovači různé podoby:

Podle konvence	Tvar časového razítka	Příklad
IBM	' yyyy-mm-dd-hh.mm.ss.mmmmmm '	' 2000-04-05-23.59.59.999999 '
ISO	' yyyy-mm-dd hh:mm:ss.mmmmmm '	' 2000-04-05 23:59:59.999999 '
14 znaková	' yyymmddhhmmssmmmmmm '	' 20000405235959999999 '

Jeho délka je 26 nebo 14 znaků. Lze vynechat vedoucí nuly měsíce, dne, hodiny, minuty a sekundy u tvaru s oddělovači. Koncové nuly mikrosekund mohou být částečně nebo zcela odstraněny. Připouští se i tvar s koncovkou 24.00.00.000000.

BLOB [(délka [K | M | G])]

Binary Large Object je velký binární objekt zadáné maximální délky. Je-li údaj délky vynechán, rozumí se 1 MB. K znamená "kilobajt", tj. 1 024 bajtů. M znamená "megabajt", tj. 1 048 576 bajtů, G znamená "gigabajt", tj. 1 073 741 824 bajtů. Maximální délka musí být v rozsahu 1 až 2 147 483 647 (nebo 2 097 152 K nebo 2 048 M nebo 2G).

CLOB [(délka [K | M | G])]

Character Large Object je velký znakový objekt (string) zadáné délky. Je-li údaj délky vynechán, rozumí se 1 MB. K znamená "kilobajt", tj. 1 024 bajtů. M znamená "megabajt", tj. 1 048 576 bajtů, G znamená "gigabajt", tj. 1 073 741 824 bajtů. Maximální délka musí být v rozsahu 1 až 2 147 483 647 (nebo 2 097 152 K nebo 2 048 M nebo 2G). Obsahuje text kódovaný ve schematu SBCS (Single Byte Character Set), např. CCSID 1208, tj. UTF-8, CCSID 1250 (Windows Latin 2, CCSID 920 (ISO-5589-2 Latin 2) atd.

NCLOB a DBCLOB [(délka [K | M | G])]

Varianty typu CLOB používající dvojice bajtů pro kódování znaku. Proto jejich maximální délka musí být v rozsahu 1 až 1 073 741 823 (nebo 1 028 576 K nebo 1 024 M nebo 1G).

NCLOB National Character Large Object obsahuje text kódovaný CCSID 1200, tj. UTF-16.

DBCLOB Double Byte Character Large Object obsahuje text kódovaný CCSID 1200, tj. UTF-16 nebo CCSID 13488, tj. UCS-2).

DATALINK(délka) nebo DATALINK

Datová spojka (DataLink) zadáné maximální délky. Maximální délka musí být v rozsahu 1 až 32718 (32717 pro "null capable"). Délka musí zahrnovat největší očekávané URL a komentář datové spojky. Je-li údaj délky vynechán, dosadí se 200.

Hodnota typu DATALINK je zapouzdřena se sadou vestavěných (built-in) skladárních funkcí. Funkce DLVALUE vytváří hodnotu pro DATALINK. K extrakci atributů z hodnoty DATALINK lze použít těchto funkcí:

DLCOMMENT
 DLLINKTYPE
 DLURLCOMPLETE
 DLURLPATH
 DLURLPATHONLY
 DLURLSCHEME
 DLURLSERVER

Datová spojka (DataLink) nemůže být částí indexu. Proto ji nelze uvést jako sloupec pro primární klíč, cizí klíč nebo podmínky jedinečnosti.

Předvolené hodnoty datových typů:

Typ dat	Předvolená hodnota
číselný	0
znakový, pevné délky	mezery
znakový, proměnné délky	řetězec s délkou 0
datum	běžné datum z INSERT
čas	běžný čas z INSERT
časové razítko	běžné časové razítko z INSERT

Speciální registry

CURRENT_DATE
 CURRENT DEBUG MODE
 CURRENT_SCHEMA
 CURRENT_SERVER
 CURRENT_TIME
 CURRENT_TIMESTAMP
 CURRENT_TIMEZONE

Standardní funkce

Rozlišují se dva druhy funkcí:

- Agregátní (aggregate) funkce, které dodají *jednu* hodnotu z *množiny hodnot* (např. ze sloupce dat). Vyhodnocují počet řádků nebo s hodnoty sloupců v řádcích tabulky. Používají se v seznamu výsledných sloupců fráze SELECT nebo v podmínce fráze HAVING. Říká se jim také *sloupcové funkce*.
- Skalární (scalar) funkce, které dodají *jednu* hodnotu z jednoho nebo více *argumentů*.

Agregátní (sloupcové) funkce

Funkce COUNT představuje počet.

COUNT (*) hvězdička představuje množinu řádků tabulky

COUNT ([ALL] výraz) výraz představuje množinu hodnot

COUNT (DISTINCT výraz) výraz představuje množinu hodnot s vyloučením duplicit

```
select count(*) from obraty
COUNT ( * )
      5
```

```
select count(distinct czbozi) from obraty
COUNT
      4
```

```
select count(all czbozi) from obraty
COUNT
      5
```

Funkce **SUM** představuje součet. Výraz musí být číselný.

SUM ([ALL] výraz) výraz představuje množinu hodnot

SUM (DISTINCT výraz) výraz představuje množinu hodnot s vyloučením duplicit

Funkce **AVG** představuje aritmetický průměr. Výraz musí být číselný.

AVG ([ALL] výraz) výraz představuje množinu hodnot

AVG (DISTINCT výraz) výraz představuje množinu hodnot s vyloučením duplicit

Funkce **MAX** představuje maximální hodnotu. Výraz může být libovolný typ dat kromě **DATALINK**.

MAX ([ALL] výraz) výraz představuje množinu hodnot

MAX (DISTINCT výraz) výraz představuje množinu hodnot s vyloučením duplicit

Funkce **MIN** představuje minimální hodnotu. Výraz může být libovolný typ dat kromě **DATALINK**.

MIN ([ALL] výraz) výraz představuje množinu hodnot

MIN (DISTINCT výraz) výraz představuje množinu hodnot s vyloučením duplicit

Skalární funkce

Skalárních funkcí je velmi mnoho. Uvedeme jen nejběžnější.

Změny typu dat (cast scalar functions)

CHAR (výraz) vrací znakovou reprezentaci hodnoty.

DATE (výraz) vrací datum z hodnoty.

DECIMAL (výraz) vrací dekadickou reprezentaci čísla nebo znakového řetězce.

INTEGER (výraz) nebo **INT (výraz)** vrací celočíselnou binární reprezentaci čísla nebo znakového řetězce.

TIME (výraz) vrací hodnotu typu TIME z výrazu typu TIME, TIMESTAMP, CHARACTER.

TIMESTAMP (výraz) vrací hodnotu typu TIMESTAMP z výrazu typu TIMESTAMP nebo CHARACTER.

TIMESTAMP_ISO (výraz) vrací hodnotu typu TIMESTAMP z výrazu typu TIMESTAMP, DATE, TIME, CHARACTER.

VARCHAR (výraz) vrací znakovou reprezentaci hodnoty.

Datum a čas (datetime scalar functions)

ADD_MONTHS (výraz, číselný-výraz) přičítá počet měsíců k datumové hodnotě.

`SELECT CHAR(CURRENT_DATE, ISO) FROM OBRATY`

```
CHAR conversion
2014-02-09
```

```
SELECT CHAR(ADD_MONTHS(CURRENT_DATE, 1 ), ISO)
FROM OBRATY
```

```
CHAR conversion
2014-03-09
```

EXTRACT (YEAR | MONTH | DAY | HOUR | MINUTE | SECOND FROM výraz) vrací číslo příslušné části z hodnoty typu TIMESTAMP, DATE nebo TIME.

```
EXTRACT ( DAY FROM DATE('2014-02-09') )
```

LAST_DAY (výraz) vrací datum posledního dne v měsíci.

```
SELECT CHAR(LAST_DAY('2014-02-07'), ISO) FROM OBRATY
```

```
CHAR conversion
2014-02-28
```

QUARTER (výraz) vrací číslo čtvrtletí.

```
QUARTER('2014-07-07')
```

```
QUARTER
3
```

WEEK (výraz) vrací číslo týdne v roce: 1 až 54. Týden začíná nedělí.

WEEK_ISO (výraz) vrací číslo týdne v roce: 1 až 53. Týden začíná pondělkem.

```
WEEK_ISO ('2014-07-07')
```

```
WEEK_ISO
28
```

Různé skalární funkce

COALESCE (výraz, výraz, ...) vrací hodnotu prvního výrazu, který není NULL.

Slovo COALESCE znamená sloučit, splynout. Následující dotaz vyprodukuje prázdné hodnoty u tří sloupců, které nebudeme chtít.

```
SELECT DISTINCT O.CZBOZI, O.ZAVOD, O.SKLAB, S.CZBOZI
FROM STAVY S
      EXCEPTION JOIN OBRATY O
      ON S.CZBOZI = O.CZBOZI
```

CZBOZI	ZAVOD	SKLAB	CZBOZI
-	-	-	00002
-	-	-	00005
-	-	-	00006
-	-	-	00008
-	-	-	00019
-	-	-	00011
-	-	-	00014
-	-	-	00018

Další dotaz nás prázdných sloupců zbaví.

```
SELECT DISTINCT COALESCE (O.CZBOZI, O.ZAVOD, O.SKLAB, S.CZBOZI)
FROM STAVY S
      EXCEPTION JOIN OBRATY O
      ON S.CZBOZI = O.CZBOZI
```

```
COALESCE
```

```
00002  
00005  
00006  
00008  
00019  
00011  
00014  
00018
```

HEX (výraz) vrací hexadecimální reprezentaci hodnoty výrazu libovolného typu.

```
SELECT CZBOZI, HEX(CZBOZI) FROM OBRATY
```

```
CZBOZI  HEX ( CZBOZI )  
00001    F0F0F0F0F1  
00003    F0F0F0F0F3  
00010    F0F0F0F1F0  
00010    F0F0F0F1F0  
00009    F0F0F0F0F9
```

LENGTH (výraz) vrací délku hodnoty výrazu libovolného typu.

MAX (výraz, výraz, ...) vrací maximální hodnotu ze seznamu výrazů slučitelných typů (např. CHAR a DATE jsou slučitelné).

MIN (výraz, výraz, ...) vrací minimální hodnotu ze seznamu výrazů slučitelných typů (např. CHAR a DATE jsou slučitelné).

RRN (označení-tabulky) vrací relativní číslo řádku.

```
SELECT RRN (OBRATY), ZAVOD, SKLAD, CZBOZI FROM OBRATY
```

```
RRN ( OBRATY )  ZAVOD  SKLAD  CZBOZI  
1               01      01      00001  
2               01      02      00003  
3               01      01      00010  
4               01      02      00010  
5               01      02      00009
```

Číselné skalární funkce

Jsou to goniometrické, logaritmické, exponenciální a jiné funkce, např. SIN(), COS(), LN(), EXP(), LOG10(), atd.

MOD (výraz1, výraz2) vrací zbytek po dělení výrazu1 výrazem2. Funkce je určena vzorcem

$$\text{MOD}(x, y) = x - (x/y) * y.$$

ROUND (výraz, výraz) zaokrouhlí číslo z prvního výrazu podle čísla z druhého výrazu. Je-li druhé číslo kladné, zaokrouhuje na udaný počet desetinných míst, je-li nula, zaokrouhuje na celky, je-li záporné, zaokrouhuje na mocniny 10. (-2 na stovky: 10^{-2}).

```
SELECT ROUND(255.545, 2),  
       ROUND(255.545, 0),  
       ROUND(255.545,-2)
```

```
FROM OBRATY
```

ROUND	ROUND	ROUND
255.550	256.000	300.000
255.550	256.000	300.000
255.550	256.000	300.000
255.550	256.000	300.000
255.550	256.000	300.000

Znakové skalární funkce

CONCAT (výraz, výraz) spojí hodnoty obou výrazů do jednoho řetězce. Výrazy musí mít slučitelné typy (např. znaky a čísla). Stejný výsledek poskytuje spojovací *operátor* || nebo *operátor* CONCAT.

```
SELECT CONCAT (CONCAT(CZBOZI, ' '), NAZZBO) FROM CENY
SELECT CZBOZI || ' ' || NAZZBO FROM CENY
SELECT CZBOZI CONCAT ' ' CONCAT NAZZBO FROM CENY
```

```
00002 Zubní pasta Kalodont
00001 PIŠKOTY OPAVIA
00003 Prádelní šňůra
```

...
INSERT (původní-řetězec , start , délka , vkládaný-řetězec) vkládá nový řetězec do původního řetězce. Start je číslo od 1, délka určuje počet vynechaných znaků.

```
SELECT INSERT(NAZZBO, 1, 0, '****') FROM CENY
```

```
INSERT
****Zubní pasta Kalodont
****PIŠKOTY OPAVIA
****Prádelní šňůra
...
```

LCASE | LOWER (výraz) převede hodnotu výrazu do malých písmen.

UCASE | UPPER (výraz) převede hodnotu výrazu do velkých písmen.

LTRIM (výraz) odstraní z levé strany hodnoty výrazu mezery (u znaků) nebo hexadecimální nuly (u binárního výrazu).

RTRIM (výraz) odstraní z pravé strany hodnoty výrazu mezery (u znaků) nebo hexadecimální nuly (u binárního výrazu).

TRIM (výraz) odstraní z obou stran hodnoty výrazu mezery (u znaků) nebo hexadecimální nuly (u binárního výrazu).

POSITION (hledaný-řetězec IN původní-řetězec) vrací číslo pozice (počítané od 1) nalezeného řetězce nebo nulu. Tato funkce počítá pozice ve znacích a bere ohled na CCSID.

POSSTR (původní-řetězec , hledaný-řetězec) vrací číslo pozice (počítané od 1) nalezeného řetězce nebo nulu. Tato funkce počítá pozice v bajtech.

LOCATE (hledaný-řetězec , původní-řetězec , start) vrací číslo pozice (počítané od 1) nalezeného řetězce nebo nulu. Hledání začíná na pozici start. Tato funkce počítá pozice ve znacích a bere ohled na CCSID.

```
SELECT NAZZBO, LOCATE ('šňů', NAZZBO, 5) FROM CENY
```

NAZZBO	LOCATE
Zubní pasta Kalodont	0
PIŠKOTY OPAVIA	0
Prádelní šňůra	10
...	

REPLACE (původní-řetězec , hledaný řetězec , náhradní-řetězec) nahradí všechny výskytu hledaného řetězce v původním řetězci náhradním řetězcem. Jestliže nic nenajde, původní řetězec zůstane nezměněn.

```
SELECT REPLACE (NAZZBO, ' ', ', ') FROM CENY
```

```
REPLACE
```

```
Zubní,pasta,Kalodont,.....  
PIŠKOTY,OPIAVIA.....  
Prádelní,šňůra.....  
...
```

SPACE (výraz) vrací znakový řetězec jednobajtových mezer v délce zadané výrazem.

STRIP (výraz [, BOTH | B], znak)

|, LEADING | L

|, TRAILING | T

odstraňuje zadaný znak z obou stran (B) nebo zleva (L) nebo zprava (T). Výsledky funkce jsou stejné jako u funkcí TRIM, LTRIM a RTRIM.

SUBSTR (výraz, start [, délka]) vrátí podřetězec z hodnoty výrazu od pozice start v zadane délce (je-li zadána) nebo v délce hodnoty výrazu (není-li zadána). Výraz je číselný nebo znakový.

Číselný se nejprve převede na znakové vyjádření. Údaje start a délka značí počty bajtů u typu CHAR nebo VARCHAR. Je-li třeba výraz kódován v UTF-8, může být znak delší než jeden bajt.

SUBSTRING (výraz, start [, délka]) vrátí podřetězec z hodnoty výrazu od pozice start v zadane délce (je-li zadána) nebo v délce hodnoty výrazu (není-li zadána). Výraz je číselný nebo znakový. Číselný se nejprve převede na znakové vyjádření. Údaje start a délka značí počty znaků. Například u typu GRAPHIC je znak dvoubajtový, u typu CHAR 1 až 6bajtový (v kódování UTF-8).

SUBSTRING (výraz FROM start [FOR délka]) totéž

RIGHT (výraz, počet) vrátí zadaný počet znaků zprava.

TRANSLATE (výraz [, cílové-znaky [, zdrojové-znaky [, výplň]]]) vrací hodnotu, kde jeden nebo více znaků může být převedeno na jiné znaky. Je to překódování řetězce podle dvou překódovacích tabulek (řetězců se stejnolehlými znaky).

Výraz může být číselný nebo znakový. Znaky v hodnotě výrazu, které se shodují se zdrojovými znaky, se převedou na znaky uvedené v cílových znacích. Přitom se berou cílové znaky na pozicích shodných s pozicemi zdrojových znaků. Je-li počet cílových znaků menší než zdrojových, doplní se cílové znaky do stejné délky znakem výplně (je-li zadán) nebo mezerou. Je-li počet cílových znaků větší, přebytek se ignoruje.

```
SELECT TRANSLATE (NAZZBO, 'zscrtnouua', 'žščřdťňóúáéíý', '*')  
FROM CENY
```

```
TRANSLATE  
Zubn* pasta Kalodont  
PIŠKOTY OPIAVIA  
Pradeln* snura  
Ponomzky pansk* tmav*  
Tricko b*l*  
...
```

Výraz (expression)

Výraz v SQL má tuto obecnou definici a poskytuje hodnotu.

```
.-operator-----.
  V
>>---+---+---+function-invocation-----+-----><
  +- + -+ +(expression)-----+
  '- - -' +-constant-----+
    +-column-name-----+
    +-variable-----+
    +-special-register-----+
    +-scalar-fullselect-----+
    +-labeled-duration-----+
    +-array-constructor-----+
    +-array-element-specification-
    +-case-expression-----+
    +-cast-specification-----+
    +-OLAP-specification-----+
    +-row-change-expression-----+
    +-sequence-reference-----+
    '-XMLECAST-specification-----'
```

operator:

```
>>-+CONCAT+-----><
  +- || ----+
  +- / ----+
  +- * ----+
  +- ** ----+
  +- + ----+
  '- - ----'
```

labeled-duration:

```
>>-+function----+--YEAR-----><
  +- (expression)-+ +-YEARS-----+
  +-constant-----+ +-MONTH-----+
  +-column-name--+ +-MONTHS-----+
  '-variable----' +-DAY-----+
    +-DAYS-----+
    +-HOUR-----+
    +-HOURS-----+
    +-MINUTE-----+
    +-MINUTES-----+
    +-SECOND-----+
    +-SECONDS-----+
    +-MICROSECOND--+
    '-MICROSECONDS-'
```

Všimneme si jen některých speciálních výrazů.

Scalar fullselect

je fullselect uzavřený v závorkách poskytující jedinou hodnotu (jediného sloupce z jediného řádku) nebo NULL.

Case expression

je rozhodovací výraz tvaru

CASE WHEN podmínka THEN výraz-pro-výsledek-nebo ELSE výraz-pro-výsledek END

CASE výraz WHEN výraz THEN výraz -pro-výsledek ELSE výraz-pro-výsledek END

Definice:

```
>>-CASE--+-searched-when-clause-+-----+--END-><
      '-simple-when-clause---'   '-ELSE--result-expression-'
```

searched-when-clause

```
.-----.
 | V
|----WHEN--search-condition--THEN--+--result-expression-+-----+
 |           '-NULL-----'
```

simple-when-clause

```
|--expression----->
 .-----.
 | V
|----WHEN--expression--THEN--+--result-expression-+-----+
 |           '-NULL-----'
```

Příklad 1:

```
SELECT S.ZAVOD, S.SKLAD,
       DEC(AVG(S.MNOZSTVI*C.CENAJ), 9,2) PRUM_CENA,
       CASE
         WHEN AVG(S.MNOZSTVI*C.CENAJ) > 5000 THEN 'NADPRŮMĚRNÝ'
         ELSE 'PODPRŮMĚRNÝ'
       END HODNOCENI
FROM STAVY S
JOIN CENY C USING (CZBOZI)
GROUP BY S.ZAVOD, S.SKLAD
ORDER BY S.ZAVOD, S.SKLAD

ZAVOD SKLAD     PRUM_CENA    HODNOCENI
 01     01        10,280.59- PODPRŮMĚRNÝ
 01     02          6,601.66 NADPRŮMĚRNÝ
 01     03            1.25 PODPRŮMĚRNÝ
 02     01        1,220.36 PODPRŮMĚRNÝ
 02     02        1,982.50 PODPRŮMĚRNÝ
```

Příklad 2:

```
SELECT ZAVOD,
       CASE ZAVOD
         WHEN '01' THEN 'Místní závod'
         WHEN '02' THEN 'Vzdálený závod'
       END NAZEV_ZAVODU
FROM STAVY
GROUP BY ZAVOD

ZAVOD NAZEV_ZAVODU
 01    Místní závod
 02    Vzdálený závod
```

Cast expression (přetypování)

je výraz pro transformaci datových typů tvaru

CAST (výraz AS typ-dat)

Vrací hodnotu výrazu převedenou do zadанého typu. Nahrazuje v podstatě skalární funkce pro změnu typu dat.

Definice:

```
>>-CAST--(--+expression-----+--AS--data-type--)-----><
      +-NULL-----+
      '-parameter-marker-'
```

data-type

```
| --+built-in-type+-----|
 | +-distinct-type+-
 | '-array-type---'
```

Příklad změny znaků na celé číslo:

```
SELECT CZBOZI, CAST (CZBOZI AS INT) FROM CENY
```

```
CZBOZI  CAST function
00002          2
00001          1
00003          3
...
```

Příklad změny dekadického čísla na znaky:

```
SELECT CENAJ, CAST (CENAJ AS CHAR(9)) FROM CENY
```

```
CENAJ  CAST function
459.00  459.00
  8.99   8.99
  1.25   1.25
 10.50   10.50
120.00  120.00
...
```

Predikáty

Predikát je podmínka, která se vztahuje k řádku nebo ke skupině řádků a je pravdivá, nepravdivá nebo neznámá.

Základní predikáty

```
>>-+expression--+- = -+---expression-----+----><
|           +- <> -+
|           +- < --+
|           +- > --+
|           +- <= -+
|           '- >= -'
+-row-value-expression--+- = -+---row-value-expression-+
|           '- <> -'
+-(--fullselect--)---+ = -+---row-value-expression-----+
|           '- <> -'
'-row-value-expression--+- = -+---(--fullselect--)-----'
|           '- <> -'
```

row-value-expression má tvar (výraz, výraz, ...) a jeho hodnota je jediný řádek s jedním nebo více sloupcí.

Kvantifikované predikáty

```
>>-+expression--+- = -+-----+-SOME---(--fullselect--)---+----><
|           +- <> -+      +-ANY--+
|           +- < --+      '-ALL--'
|           +- > --+
|           +- <= -+
|           '- >= -'
+-row-value-expression-- = -+---SOME---(--fullselect--)---+
|           '-ANY--'
'-row-value-expression-- <> --ALL---(--fullselect--)-----'
```

Predikát **ANY** (totéž jako **SOME**) zjišťuje, zda podmínku splňuje *alespoň jedna hodnota* z množiny.

Predikát **ALL** zjišťuje, zda podmínku splňují *všechny hodnoty* z množiny.

Predikát **BETWEEN**

```
>>-expression--+-----+--BETWEEN--expression--AND--expression---><
|           '-NOT-'
```

Porovnává hodnotu, zda je v rozsahu hodnot (včetně).

Predikát **DISTINCT** (odlišný)

Testuje na odlišnost hodnot.

Definice:

```
>>---expression--IS---+-----+--DISTINCT FROM--expression-----><
|           '-NOT-'
```

Výraz *hodnota1 IS NOT DISTINCT FROM hodnota2* zjišťuje, zda se hodnoty liší. Dvě prázdné hodnoty (NULL) se nikdy neliší (jsou si rovny), a prázdná hodnota se vždy liší od neprázdné. U dvou neprázdných hodnot je porovnání jasné.

Podmínka

hodnota1 IS NOT DISTINCT FROM hodnota2

je ekvivalentní této podmínce:

```
( hodnota1 IS NOT NULL AND hodnota2 IS NOT NULL AND hodnota1 = hodnota2 )
OR
( hodnota1 IS NULL AND hodnota2 IS NULL )
```

Podmínka

hodnota1 IS DISTINCT FROM hodnota1

je ekvivalentní této podmínce:

```
NOT (hodnota1 IS NOT DISTINCT FROM hodnota1)
```

Predikát EXISTS

EXISTS (*fullselect*)

Testuje na přítomnost jednoho nebo více řádků.

Predikát IN

Porovnává hodnotu nebo hodnoty s množinou hodnot.

Definice:

```
>>-+expression1--+----+--IN--+-(--fullselect1--)-----+--><
      |           '-NOT-'          |   .-,-----.
      |           |       V      |   |
      +-(-----expression2---+---)-+
      '---expression3-----'     |
'-row-value-expression--+----+--IN--(--fullselect2--)----'
      '-NOT-'
```

Predikát LIKE

Hledá řetězec se vzorkem v zadáném řetězci.

Definice:

```
>>-match-expression--+----+--LIKE--pattern-expression--+-----+--><
      ' -NOT-'                      ' -ESCAPE--escape-expression-'
```

match-expression je zadaný řetězec.

pattern-expression je vzorek, podle nějž se porovnávají znaky v zadáném řetězci.

Ve vzorku mohou být znaky

- podtržítko _ představuje libovolný jednotlivý znak,
- procento % představuje řetězec o více znacích nebo prázdný řetězec,
- libovolné znaky představují sebe.

Příklad

Název zboží je AAAA+%BBB%_CCC

```
SELECT NAZZBO
FROM CENY WHERE NAZZBO LIKE 'AAAA%BBB%'
```

```
NAZZBO
AAAA+%BBB%_CCC
```

Příkaz najde celý název:

hledá všechno mezi AAAA a BBB a vše za BBB.

escape-expression je tzv. únikový znak, který modifikuje význam speciálních znaků _ a %.

Dovoluje v zadáném řetězci vyhledat znak _ a %. Tento znak ruší speciální význam speciálního

znaku, který bezprostředně následuje. Ve vzorku se může objevit jen ve dvojici se sebou samým, s následujícím znakem _ nebo znakem %. Je-li únikový znak např. +, mohou být ve vzorku uvedeny dvojice ++, +_ nebo +%.

Vzorek Hledá se

+% znak procento

++% znak plus následovaný libovolným počtem libovolnch znaků

+++% znak plus následovaný znakem procento

- Název zboží je AAAA%BBB_CCC

```
SELECT NAZZBO  
FROM CENY WHERE NAZZBO LIKE 'AAAAA+%BBB%' ESCAPE '+'
```

NAZZBO
AAAA%BBB CCC

- Název zboží je AAAA%BBB%_CCC

```
SELECT NAZZBO
FROM CENY WHERE NAZZBO LIKE 'AAAAA+'%BBB+'%' ESCAPE '+'
```

NAZZBO
AAAA% BBB% CCC

- Název zboží je AAAA+%BBB%_CCC

```
SELECT NAZZBO  
FROM CENY WHERE NAZZBO LIKE 'AAAA++%BBB+%%' ESCAPE '+'
```

NAZZBO

Pradíkát NLLL

wíkaz IS [NOT] NULL

Testuje, zda výraz má či nemá hodnotu NUU I

Syntaktické definice příkazů

CREATE TABLE

```
>>--CREATE TABLE--table-name--+
   'FOR SYSTEM NAME--system-object-identifier'-->

   .--,-----.
   V | |
>>--(----+column-definition-----+---+-->
   | +LIKE--+table-name--+-----+-
   | | '-view-name--' '-copy-options-' |
   | +unique-constraint-----+
   | +referential-constraint-----+
   | '-check-constraint-----'
   +LIKE--+table-name-----+
   | '-view-name--' '-copy-options-' |
   +-as-result-table-----+
   | '-copy-options-' |
   '-materialized-query-definition-----'

>>+-----+----->
   '-NOT LOGGED INITIALLY-'
```

```

>-+-----+-----+-->
| .-CARDINALITY-. | '-RCDFMT--format-name-
'-VOLATILE--+-----'

>-+-----+-----+--><
' -media-preference-' +distribution-clause+
' -partitioning-clause-' 

media-preference

.-UNIT ANY..
| --+UNIT SSD+-----| 

built-in-type

|---+---+---+SMALLINT---+---+
| | +--+INTEGER---+
| | | 'INT-----'
| | '--BIGINT-----'
| | | .-(5,0)---.
+---+---+DECIMAL---+---+
| | | 'DEC-----' | | .-,0-----.
| | '+--NUMERIC---' | '-(--integer---+---+)
| | | 'NUM-----' | | ', integer-
| | | .-(--53--)-----.
+---+---+FLOAT---+---+
| | | '-(--integer---)' |
| | '+REAL-----+
| | | .-PRECISION-.
| | '-DOUBLE---+-----'
| | | .-(--34--)-. 
+---+DECFLOAT---+---+
| | | '-(--16--)'
| | | .-(--1--)-----.
+---+---+CHARACTER---+---+
| | | 'CHAR-----' | '-(--integer---)' | '+FOR BIT DATA---+
| | '+--+CHARACTER---+VARYING---+--(--integer---+---+)
| | | 'CHAR-----' | | '-allocate-clause-' | '+FOR SBCS DATA---+
| | '| 'CHAR-----' | | | '+FOR MIXED DATA---+
| | '| 'VARCHAR-----' | | | '-ccsid-clause---'
| | | .-(--1M--)-----.
+---+---+CHARACTER---+LARGE OBJECT---+---+
| | | '-CHAR-----' | '-(--integer---+---+)' | '-allocate-clause-' | '+FOR SBCS DATA---+
| | '| CLOB-----' | | '+K+ | '+FOR MIXED DATA---+
| | | | '+M+ | '| '-ccsid-clause---'
| | | | '| 'G-' | 
| | | .-(--1--)-----.
+---+---+GRAPHIC---+---+
| | | '-(--integer---)' | '| '-ccsid-clause-' 
| | '+--GRAPHIC VARYING---+--(--integer---+---+)
| | | '| VARGRAPHIC-----' | '| '-allocate-clause-' | 
| | | | .-(--1M--)-----.
| | '| DBCLOB-----' | '| '-(--integer---+---+)' | '| '-allocate-clause-' 
| | | | '+K+ | |
| | | | '+M+ | 
| | | | '| 'G-' | 
| | | | .-(--1--)-----.
+---+---+NATIONAL CHARACTER---+---+
| | | '+NATIONAL CHAR-----+' | '| '-(--integer---)' | '| '-normalize-clause-' 
| | '| NCHAR-----' | 
| | '| '+--NATIONAL CHARACTER---+VARYING---+--(--integer---+---+)
| | | | '+NATIONAL CHAR-----+' | '| '-allocate-clause-' | 
| | '| '| NCHAR-----' | 
| | '| '| NVARCHAR-----' | 
| | | .-(--1M--)-----.
+---+---+NATIONAL CHARACTER---+LARGE OBJECT---+---+
| | | '| NCLOB-----' | '| '-(--integer---+---+)' | '| '-allocate-clause-' 
| | '| '| NCLOB-----' | | '+K+ | 
| | '| '| '+M+ | 
| | '| '| '| 'G-' | 
| | | .-(--1--)-----.
+---+---+BINARY---+---+
| | | '| BINARY VARYING---+--(--integer---+---+)
| | '| '| VARBINARY-----' | '| '-allocate-clause-' | 
| | | | .-(--1M--)-----.
| | '| '| BLOB-----' | '| '-BINARY LARGE OBJECT-' | '| '-(--integer---+---+)' | '| '-allocate-clause-' 
| | | | '+K+ | |
| | | | '+M+ | 
| | | | '| 'G-' | 
| | | .-(--0--)-. 
+---+---+DATE---+---+
| | | .-(--0--)-. | 

```

```

| +-TIME-----+-----+
| | .--(6)--.- |
| '-TIMESTAMP---+-----+
| | .--(200)--.
+---DATALINK---+-----+-----+-----+
| | '-(--integer--)-' '-allocate-clause-' '-ccsid-clause-'
+---ROWID-----+
'---XML-----+-----+-----+
| | '-allocate-clause-' '-ccsid-clause-'

allocate-clause

| --ALLOCATE--(--integer--)-----| 

ccsid-clause

| --CCSID--integer-----+-----+
| | '-normalize-clause-' 

normalize-clause

. -NOT NORMALIZED -.
| --+--NORMALIZED-----| 

default-clause

. -WITH -.
| --+----+--DEFAULT--+-----+-----+
| | +--constant-----+
| | +--USER-----+
| | +--NULL-----+
| | +--CURRENT_DATE-----+
| | +--CURRENT_TIME-----+
| | +--CURRENT_TIMESTAMP-----+
| | '-cast-function-name--(--+constant-----+--)-'
| | | +--USER-----+
| | | +--CURRENT_DATE-----+
| | | +--CURRENT_TIME-----+
| | | '-CURRENT_TIMESTAMP-' 

identity-options

| --AS IDENTITY-----+-----+
| | .-----.
| | | V .-1-----.
| | | | (1) |
| | '-(----+--START WITH--+--numeric-constant-----+--)-'
| | | .-1-----.
| | '+INCREMENT BY--+--numeric-constant--+
| | | .-NO MINVALUE-----.
| | '+--MINVALUE--numeric-constant-----+
| | | .-NO MAXVALUE-----.
| | '+--MAXVALUE--numeric-constant-----+
| | | .-NO CYCLE-.
| | '+--CYCLE-----+
| | | .-CACHE--20-----.
| | '+--NO CACHE-----+
| | | '-CACHE--integer-'
| | | .-NO ORDER-.
| | '-+--ORDER-----+-----| 

as-row-change-timestamp-clause

| --FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP-----| 

column-constraint

| --+-----+-----+-----+-----+-----+-----+
| | '-CONSTRAINT--constraint-name-' | '| '-UNIQUE-----' |
| | | '+references-clause-----+
| | | '-CHECK--(--check-condition--)-' 

```

datalink-options

```

    .-LINKTYPE URL-.  .-NO LINK CONTROL-----.
|---+-----+-----+-----+-----+-----+-----+
|      '-FILE LINK CONTROL--+file-link-options+-'
|          '-MODE DB2OPTIONS---'

```

file-link-options

```

    .-----.
    V           | (1)
|-----+INTEGRITY ALL-----+-----+
+--+READ PERMISSION FS-----+
| '-READ PERMISSION DB-'   |
+--+WRITE PERMISSION FS-----+
| '-WRITE PERMISSION BLOCKED-' |
+--RECOVERY NO-----+
'-+ON UNLINK RESTORE-----'
  '-ON UNLINK DELETE--'

```

as-result-table

```

|---+-----+-----+-----+-----+-----+-----+-->
|      '-(--column-name--+-----+--+--)'   |
|          | .-COLUMN-.
|          | '-FOR--+-----+--system-column-name--'
|-----+-----+-----+-----+-----+-----+-----+
>--AS--(--select-statement--)--+WITH NO DATA-----+
|          '-WITH DATA---'

```

copy-options

```

    .-----.
    V           | .-COLUMN ATTRIBUTES-.
    |   .-EXCLUDING IDENTITY--+-----+-----+
    |       .-COLUMN ATTRIBUTES-.
|-----+--+INCLUDING IDENTITY-----+-----+-----+-----+
|           .-COLUMN-.
|   .-EXCLUDING--+-----+--DEFAULTS-.
|       .-COLUMN-.
+--+INCLUDING--+-----+--DEFAULTS--+-----+
|   '-USING TYPE DEFAULTS-----'
|           .-COLUMN ATTRIBUTES-.
|   .-EXCLUDING IMPLICITLY HIDDEN -+-----+-----+
|       .-COLUMN ATTRIBUTES-.
+--+INCLUDING IMPLICITLY HIDDEN -+-----+-----+-----+
|           .-COLUMN ATTRIBUTES-.
|   .-EXCLUDING ROW CHANGE TIMESTAMP -+-----+-----+
|       .-COLUMN ATTRIBUTES-.
|   .-INCLUDING ROW CHANGE TIMESTAMP -+-----+-----+

```

unique-constraint

```

|---+-----+-----+-----+-----+-----+-----+-->
|      '-CONSTRAINT--constraint-name-'  '-UNIQUE-----'
|          .-,
|          V
|-----+-----+-----+-----+-----+-----+
>--column-name--+-->

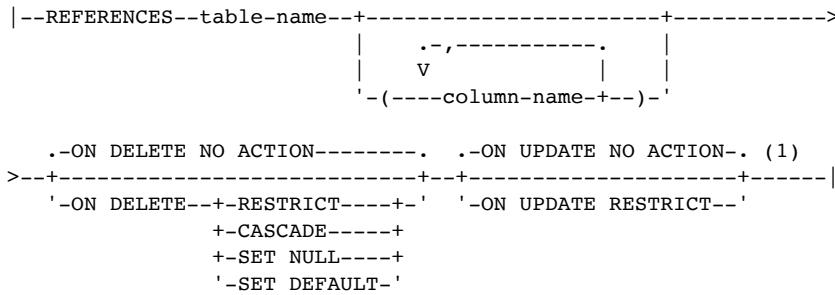
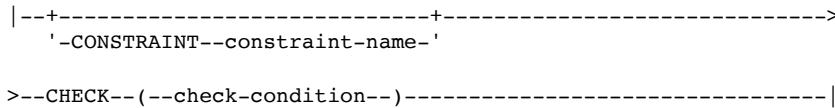
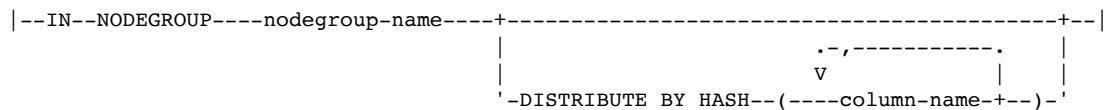
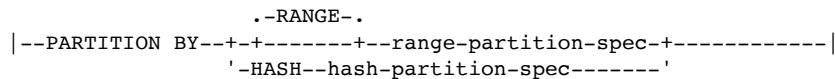
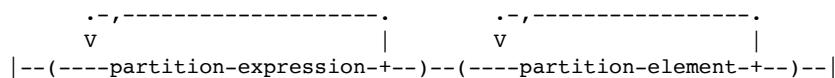
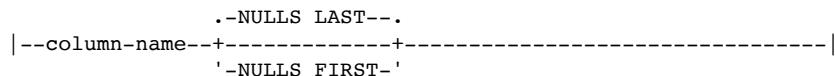
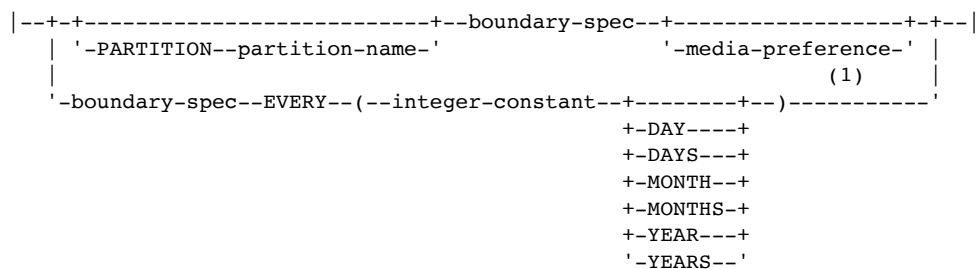
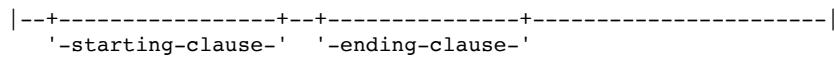
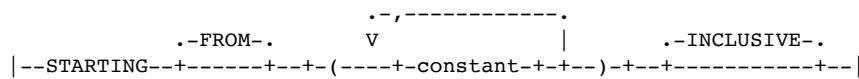
```

referential-constraint

```

|---+-----+-----+-----+-----+-----+-----+-->
|      '-CONSTRAINT--constraint-name-' 
|          .-,
|          V
|-----+-----+-----+-----+-----+-----+
>--(--column-name+--)--references-clause-----+

```

references-clause**check-constraint****distribution-clause****partitioning-clause****range-partition-spec****partition-expression****partition-element****boundary-spec****starting-clause**

```

|      +-MINVALUE-+      |  '-EXCLUSIVE-
|      '-MAXVALUE-'   |
+-constant+-----+
+-MINVALUE-
'-MAXVALUE-'
```

ending-clause

```

. , ----- .
.-AT-.      V      |  .-INCLUSIVE-.
|--ENDING---+----+---+(-+constant+-+---)-+---+-----+---+
|      +-MINVALUE-+      |  '-EXCLUSIVE-
|      '-MAXVALUE-'   |
+-constant+-----+
+-MINVALUE-
'-MAXVALUE-'
```

hash-partition-spec

```

. , ----- .
V      |
|--(--column-name---)--INTO--integer--PARTITIONS-----|
```

materialized-query-definition

```

|--+-----+-----+-->
'-(--column-name---+-----+---)-
|      .-COLUMN-.
|      '-FOR--+-----+--system-column-name-'

>--AS--(--select-statement--)--refreshable-table-options-----|
```

refreshable-table-options

```

. , ----- .
V  (1)      |
|--+DATA INITIALLY DEFERRED---+--REFRESH DEFERRED-----+--MAINTAINED BY USER-----+---+
|  '-DATA INITIALLY IMMEDIATE-'      |  ..-ENABLE QUERY OPTIMIZATION--. |
|                                         '-+--DISABLE QUERY OPTIMIZATION-+'
```

SELECT

select-clause

from-clause

table-reference:


```
|--CONNECT BY---+---search-condition-----|  
    '-NOCYCLE-'
```

pseudo columns

CONNECT_BY_ISCYCLE - 1 row is in a cycle, 0 row is not in a cycle

CONNECT_BY_ISLEAF - 1 row is a leaf, 0 row is not a leaf

LEVEL - number of the recursive step

unary operators

```
>>-CONNECT_BY_ROOT--expression-----><
```

For every row in the hierarchy, this operator returns the expression for the row's root ancestor.

```
>>-PRIOR--expression-----><
```

The expression refers to the previous level of recursion.

function

```
>>-SYS_CONNECT_BY_PATH--(--string-expression1--,--string-expression2--)><
```

Builds a string representing a path from the root row to this row.

fullselect

```
>>-+subselect----->  
  +-<(fullselect)>----+  
  '-|- values-clause |-'
```

```
.-----.  
V           |  
>--+-----+--->  
  |       .-DISTINCT-.  
  |   +-UNION---+-----+---+<subselect-----+  
  |   ' -ALL-----'     |   +-<(fullselect)>----+  
  |   .-DISTINCT-.     |   '-|- values-clause |-'  
  +-EXCEPT---+-----+  
  |       .-DISTINCT-.  
  '-INTERSECT---+--'
```

```
>>-----+---+-----><  
  '-order-by-clause-'  '-fetch-first-clause-'
```

values-clause

```
.-----.  
V           |  
|--VALUES---| values-row |---+-----|
```

values-row

```
|--+-+expression+-----+-----|  
  |  '-NULL-----'  
  |  .-----.  
  |  V  
  '-(<---+expression+---+---)-'  
      '-NULL-----'
```

select-statement

```
>>------+----->  
  |           .-----.  
  |           V           |  
  '-WITH---+-----+---common-table-expression---+'  
      '-RECURSIVE-'
```

```
>>-fullselect----->
```

```

.
|-----| (1) (2)
>----+-----+-----><
  +--update-clause-----+
  +--read-only-clause-----+
  +--optimize-clause-----+
  +--isolation-clause-----+
  '-concurrent-access-resolution-clause-'


common-table-expression

>>-table-identifier-----+----->
  |   .,-,-----.
  |   V
  '-(---column-name---)-'

>>-AS---(--fullselect--)-----+-----+-----+-----><
  '-search-clause-'   '-cycle-clause-'


search-clause:

      .,-,-----.
      V
| --SEARCH--+DEPTH FIRST--+BY---column-name---+SET--seq-column-name--|
  '-BREADTH FIRST-'


cycle-clause:

      .,-,-----.
      V
| --CYCLE---column-name---+----->

>>-SET--cycle-column-name--TO--constant--DEFAULT--constant----->

>>+-----+-----+-----|-----+
  '-USING--using-column-name-'


update-clause:

>>-FOR UPDATE-----+-----><
  |   .,-,-----.
  |   V
  '-OF---column-name---'


read-only-clause:

>>-FOR READ ONLY-----><


optimize-clause:

>>-OPTIMIZE FOR---+--integer---+--ROW---+-----><
  '-ALL----'   '-ROWS-'


isolation-clause:
>>-WITH---+NC-----+-----><
  +-UR-----+
  +-CS---+-----+---+
  |   '-KEEP LOCKS-'   |
  +-RS---+-----+---+
  |   '-lock-clause-'   |
  '-RR---+-----+---+
  '-lock-clause-'


lock-clause:

| --USE AND KEEP EXCLUSIVE LOCKS-----|
```

concurrent-access-resolution-clause:

```
>>-+SKIP LOCKED DATA-----+
    +USE CURRENTLY COMMITTED-
    '-WAIT FOR OUTCOME-----'
```

CREATE VIEW

CREATE INDEX

```
>>>CREATE --+-----+----->
    +UNIQUE--+-----+--+
    |          '-WHERE NOT NULL-' |
    '|-ENCODED VECTOR-----' |

>>>INDEX--index-name--+-----+--+
                           '|-FOR SYSTEM NAME--system-object-identifier-' |

                           .-, -----.
                           V           .-ASC--. |
>>>ON--table-name--(----key-expression----+----+----)----->
                           '|-DESC-' |

>>>+-----+--index-options----->
    '| WHERE--search-condition-' |
```

key-expression

index-options

```
|-----+-----+----->
|           .-DISTINCT-.          |
|   -WITH--integer--+-----+--VALUES-
|-----+-----+----->
>--+-----+-----+----->
+--NOT PARTITIONED+-----+
'--PARTITIONED-----'
>--+-----+-----+----->
|           .-, -----.
|           V
|-INCLUDE--(----aggregate-function-name -(--expression--)---)-'
>--+-----+-----+----->
. -PAGESIZE--64-----.
>--+-----+-----+----->
```

```

'--PAGESIZE--+-8---+'
    +-16---+
    +-32---+
    +-128---+
    +-256---+
    '-512-'


>--+-----+----->
|          .-ADD ALL COLUMNS-----. |
'-RCDFMT--format-name-----+-----+
    +-ADD KEYS ONLY-----+
    |      .-,-----.
    |      V
    '-ADD-----column-name---+'


(1)
>--+-----+-----|
'--media-preference-'


media-preference

    .-UNIT ANY..
|--+UNIT SSD+-----|



```

CREATE SEQUENCE

```

>>-CREATE--+-----+--SEQUENCE--sequence-name----->
    'OR REPLACE-'


    .-----.
    V          (1) |
>--+-----+-----+-----><
    |      .-INTEGER---.
    +-AS---+data-type+-----+
    +-START WITH--numeric-constant-----+
    |      .-1-----.
    +-INCREMENT BY--+numeric-constant+++
    |      .-NO MINVALUE-----.
    +-+MINVALUE--numeric-constant+----+
    |      .-NO MAXVALUE-----.
    +-+MAXVALUE--numeric-constant+----+
    |      .-NO CYCLE-.
    +-+CYCLE--+-----+
    |      .-CACHE--20-----.
    +-+NO CACHE-----+
    |      '-CACHE--integer-constant-'
    |      .-NO ORDER-.
    '-+ORDER---+'



```

data-type

```
|--+built-in-type-----+-----|
    '-distinct-type-name-'
```

built-in-type

```
|--+---SMALLINT---+-----+-----|
|  +-+INTEGER-+++
|  |  '-INT----'
|  |  '--BIGINT--'
|  |      .-(5,0)-----.
|  '-+DECIMAL-+++
|  |  '-DEC----'   |  .-,0-.
|  '-+NUMERIC-++'  '-(--integer--+---+-)'
    '|  '-NUM----'
```

INSERT

```
>>-INSERT INTO--+-table-name-+-----+----->
      '-view-name-' |     .-,-----.
                      V          |
      '(-column-name---)'|-----'

>--+-----+-----+-----+----->
  '-include-columns-'  +-OVERRIDING SYSTEM VALUE-
    '-OVERRIDING USER VALUE-'


      .-,-----.
      V          |
>--+VALUES---+---expression+-----+---+-----+-----> <-
  |           +-DEFAULT----+
  |           '-NULL-----'
  |
  |           .-,-----.
  |           V          |
  |   '(-+expression+-+---)'
  |
  |           +-DEFAULT----+
  |           '-NULL-----'

+insert-multiple-rows--+
  '-isolation-clause-'


'-----+-----+-----+-----+----->
  |           .-,-----.
  |           V          |
  |   '-----+-----+-----+-----+-----'
  |
  |           '-fullselect-' |           '-isolation-clause-' '
  |
  |           .-,-----.
  |           V          |
  |
  '-WITH--+-----+-----common-table-expression+-----'
    '-RECURSIVE-'
```

include-columns

```
--INCLUDE--(----column-name---+-----+--data-type---)---  
|                                |  
|          .-COLUMN-.           |  
|          '-FOR---+-----+--system-column-name---'|
```

insert-multiple-rows

```
|---+--integer--+--ROWS--VALUES--(--host-structure-array--)-----|  
|     '-variable-'|
```

isolation-clause

```
|--WITH--+NC-----+-----+  
|      +UR-----+  
|      +CS-----+--+  
|          |'KEEP LOCKS-'|  
|      +RS-----+--+  
|          |'_lock-clause-'|  
|      '-RR-----+-'  
|          |'_lock-clause-'|
```

lock-clause

|--USE AND KEEP EXCLUSIVE LOCKS-----|

UPDATE

```
>>-UPDATE--+table-name+-+-----+----->
      '-view-name-'  '-correlation-clause-'

>-+-----+SET--assignment-clause----->
  +-OVERRIDING SYSTEM VALUE-
  '-OVERRIDING USER VALUE--'

>-+-----+
  '-WHERE--search-condition-'
```

```

V
>-----+-----><
+-isolation-clause-----+
'-concurrent-access-resolution-clause-'



ALTER TABLE

>>-ALTER TABLE--table-name----->

.
V      .-COLUMN-.
>----+ADD-----+--column-definition-----+-----+-----+><
|           |
|           ..COLUMN..
|           +-ALTER-----+--column-alteration-----+
|           |       .-COLUMN-.          .-CASCADE--.
|           +-DROP-----+--column-name-----+-----+
|           |           'RESTRICT'
|           +-ADD-----+unique-constraint-----+
|           |       +-referential-constraint-+
|           |       '-check-constraint-----'
|           |           .-CASCADE--.
|           +-DROP-----+PRIMARY KEY-----+-----+-----+
|           |       '+-UNIQUE-----+--constraint-name-'  '-RESTRICT-'
|           |       +-FOREIGN KEY-
|           |       +-CHECK-----+
|           |       '-CONSTRAINT--'
|           +-ADD-----+partitioning-clause-----+
|           +-DROP PARTITIONING-----+
|           +-ADD PARTITION--add-partition-----+
|           +-ALTER PARTITION--partition-name--boundary-spec-----+--+
|                           |           '-media-preference-'
|           +-DROP PARTITION--partition-name--+DELETE ROWS-----+-----+
|                           |           '-PRESERVE ROWS-'
|                           .-MATERIALIZED-.
|                           .-+-----+--QUERY-.
|           +-ADD-----+-----+--materialized-query-definition-----+
|           |       .-MATERIALIZED-.
|           +-ALTER-----+-----+--QUERY--materialized-query-table-alteration--+
|           |       .-MATERIALIZED-.
|           +-DROP-----+-----+--QUERY-----+
|           +-ACTIVATE--NOT LOGGED INITIALLY-----+-----+
|                           |           '-WITH EMPTY TABLE-'
|                           .-CARDINALITY-.
|           +-+VOLATILE-----+-----+
|           |   '-NOT VOLATILE-'
|           '-ALTER--media-preference-----'

media-preference

.-UNIT ANY-.
|---+UNIT SSD-----+-----|


column-definition

|--column-name-----+----->
|           .-COLUMN-.
|           '-FOR--+-----+--system-column-name-'



(1) (2)
>>data-type----->

.
V
>-----+-----+-----+-----+><
+-default-clause-----+
|   .-GENERATED ALWAYS-----. (3)
|   +-+-----+-----+-----+><

```

```

|   '-GENERATED BY DEFAULT-'      '+-identity-options-----+ |
|                               '-as-row-change-timestamp-clause-' |
+-NOT NULL-----+
|   .-NOT HIDDEN-----.
+-+-----+
|   '-IMPLICITLY HIDDEN-'      |
+-column-constraint-----+
+-FIELDPROC--external-program-name--+-----+
|   |   .-, -----.
|   |   V
|   |   '-(----constant---)-'
|   (4)
`-datalink-options-----+

```

data-type

```

|---+built-in-type-----+-----|
|   '-distinct-type-name-' |

```

built-in-type

```

|---+---+SMALLINT-----+-----+
|   | +-+INTEGER---+
|   |   '-INT-----'
|   |   '--BIGINT-----'
|   |       .-(5,0)-----.
+-+--+DECIMAL-----+-----+
|   |   '-DEC-----'   |   .-,0-----.
|   |   '-NUMERIC---'  '-(--integer-----+--)'
|   |       '-NUM-----'   ', integer'
|   |       .-(--52--)-----.
+-+--+FLOAT-----+-----+
|   |   '-(--integer---)' |
|   |   '+REAL-----+
|   |       .-PRECISION..
|   |   '-DOUBLE-----+
|   |       .-(--34--)..
+-+--+DECFLOAT-----+
|   |   '-(--16--)'
|   |       .-(--1--)-----.
+-+--+CHARACTER-----+-----+
|   |   '-CHAR-----'   '-(--integer---)'
|   |   '+-+CHARACTER---VARYING-----+--'
|   |       '-(--integer---+--)'
|   |       '-allocate-clause-'   '+FOR BIT DATA---+
|   |       '-VARCHAR-----'   '+FOR SBCS DATA---+
|   |           |           '+FOR MIXED DATA---+
|   |           |           '-ccsid-clause---'
|   |       .-(--1M--)-----.
'-+--+CHARACTER---LARGE OBJECT-----+-----+
|   |   '-CHAR-----'   '|   '-(--integer---+--)'   '-allocate-clause-'   '+FOR SBCS DATA---+
|   |   '-CLOB-----'   |           '+K+'
|   |           |           '+M+'
|   |           |           '-G'
|   |       .-(--1--)-----.
+-+--+GRAPHIC-----+-----+
|   |   '-(--integer---)'   '|   '-ccsid-clause-'   '
|   |   '+-+GRAPHIC VARYING-----+--'
|   |       '-(--integer---)'   '|   '-allocate-clause-'   '+FOR MIXED DATA---+
|   |       '-VARGRAPHIC-----'   '|   '-G-'
|   |       .-(--1M--)-----.
'-+--+DBCLOB-----+-----+
|   |   '-(--integer---+--)'   '|   '-allocate-clause-'   '
|   |       '+K+
|   |       '+M+
|   |       '-G'
|   |       .-(--1--)-----.
+-+--+NATIONAL CHARACTER-----+-----+
|   |   '+-NATIONAL CHAR-----'   '|   '-(--integer---)'   '|   '-normalize-clause-'   '
|   |   '-NCHAR-----'
|   |   '+-+NATIONAL CHARACTER---VARYING-----+--'
|   |       '|   '+-NATIONAL CHAR-----+'   '|   '-allocate-clause-'   '
|   |       '|   '-NCHAR-----'   '|   '-NVARCHAR-----'
|   |       .-(--1M--)-----.
'-+--+NATIONAL CHARACTER---LARGE OBJECT-----+-----+
|   |   '|   '-NCHAR-----'   '|   '-(--integer---+--)'   '|   '-allocate-clause-'   '
|   |   '|   '-NCLOB-----'   |           '+K+
|   |           |           '+M+
|   |           |           '-G'
|   |       .-(--1--)-----.
+-+--+BINARY-----+-----+
|   |   '|   '-(--integer---)'   '|   '
|   |   '|   '+-+BINARY VARYING-----+--'
|   |       '|   '-VARBINARY-----'   '|   '-allocate-clause-'   '
|   |       '|   .-(--1M--)-----.
`-+--+BINARY LARGE OBJECT-----+-----+

```

```

|   '-BLOB-----'      '-(--integer---+---+--)-'  '-allocate-clause-'
|   |           +-K-
|   |           +-M-
|   |           '-G-
+
|   +-DATE-----+
|   |   .-(-0--). |
|   |   +-TIME-----+
|   |   |   .-(-6--). |
|   |   '-TIMESTAMP--+-----+
|   |   .-(-200--).
+
|   +-DATALINK-----+
|   |   '-(--integer--)'  '-allocate-clause-'  '-ccsid-clause-'
+
|   +-ROWID-----+
|   '-XML-----+
|   |   '-allocate-clause-'  '-ccsid-clause-'

allocate-clause

| --ALLOCATE--(integer)-----|


ccsid-clause

| --CCSID--integer-----+
|   |   '-normalize-clause-'


normalize-clause

. -NOT NORMALIZED-.
| --+NORMALIZED-----|


default-clause

. -WITH-.
| --+-----+--DEFAULT--+-----+|
|   |   +-constant-----+
|   |   +-USER-----+
|   |   +-NULL-----+
|   |   +-CURRENT_DATE-----+
|   |   +-CURRENT_TIME-----+
|   |   +-CURRENT_TIMESTAMP-----+
|   |   '|-cast-function-name--(---+constant-----+--)-'
|   |   |   +-USER-----+
|   |   |   +-CURRENT_DATE-----+
|   |   |   +-CURRENT_TIME-----+
|   |   |   '|-CURRENT_TIMESTAMP-'


identity-options

| --AS IDENTITY-----+-----+|
|   |   .-----.
|   |   V       .-1-----. (1) |
|   |   '|-(----+START WITH--+numeric-constant+---+---+--)-'
|   |   |   .-1-----.
|   |   +-INCREMENT BY--+numeric-constant+--+
|   |   |   .-NO MINVALUE-----.
|   |   +-+MINVALUE--numeric-constant+---+
|   |   |   .-NO MAXVALUE-----.
|   |   +-+MAXVALUE--numeric-constant+---+
|   |   |   .-NO CYCLE-.
|   |   +-+CYCLE-----+
|   |   |   .-CACHE--20-----.
|   |   +-+NO CACHE-----+
|   |   |   '|-CACHE--integer-
|   |   |   .-NO ORDER-.
|   |   '-+ORDER-----|


as-row-change-timestamp-clause

| --FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP-----|


column-constraint

```

```
|---+-----+---+---+PRIMARY KEY+-----+---+|  
|'-CONSTRAINT--constraint-name-' | '|-UNIQUE-----'|  
| |'+-references-clause-----+'  
| '|-CHECK--(--check-condition--)-'
```

datalink-options

```
.-LINKTYPE URL-. .-NO LINK CONTROL-----.
|-----+-----+-----+-----|
|-----'-FILE LINK CONTROL--+-file-link-options-+-'
|-----'-----'-MODE DB2OPTIONS---'
```

file-link-options

```
-----+-----+-----+-----+-----+-----+
| V | (1) |  
|----+INTEGRITY ALL-----+-----+-----+-----+  
| +--+READ PERMISSION FS--+-----+  
| | '-READ PERMISSION DB-' |  
| +--+WRITE PERMISSION FS-----+-----+  
| | '-WRITE PERMISSION BLOCKED-' |  
| +--RECOVERY NO-----+-----+  
| | '-ON UNLINK RESTORE-----+'  
| | '-ON UNLINK DELETE--'  
-----+-----+-----+-----+-----+
```

column-alteration

```
--column-name--+-SET--+-default-clause--+(2)+--+  
| V | V | (2) |  
| --DATA TYPE--data-type-' | .-GENERATED ALWAYS-. (1) |  
| +-+-----+ +--+-----+ +--+-----+  
| | '-GENERATED BY DEFAULT-' '-identity-options-' |  
| +-NOT NULL-----+  
| | .-NOT HIDDEN-. |  
| +-+-----+ +--+-----+  
| | '-IMPLICITLY HIDDEN-' |  
| '-FIELDPROC--external-program-name--+'  
| | .-,-----.|  
| | V |  
| '-(---constant---)-'  
-----+(2)-----+  
| V | (2) |  
| --DROP--+-DEFAULT-----+  
| | +-NOT NULL-----+  
| | +-IDENTITY-----+  
| | +-ROW CHANGE TIMESTAMP+  
| | '-FIELDPROC-----+'  
| '-identity-alteration-
```

identity-alteration

DROP

```
>>--DROP--+-INDEX--index-name-----+
   +-SCHEMA--schema-name--+
   |                         +-RESTRICT-+
   |                         '-CASCADE--'
   |                         .-RESTRICT-.
   +-SEQUENCE--sequence-name--+
   +-TABLE--table-name--+
```



```

' -NCLOB-----'
+--K-+
+-M-+
'-G-'  

    .-(-1--)-----.  

+---+--BINARY--+-----+-----+-----+-----+  

| | | '-(--integer--)-' | | |  

| | '+-BINARY VARYING---+---(--integer--)-' | |  

| | '|-VARBINARY-----'| |  

| | | | .-(-1M--)-----.| |  

| | | | '-BLOB-----+-----+-----+-----+  

| | | | '|-BINARY LARGE OBJECT-' | '-(--integer--+---+---+---)-' |  

| | | | | | +--K-+  

| | | | | | +--M-+  

| | | | | | '|-G-'  

+---+--DATE-----+-----+-----+-----+-----+  

| | | | .-(-0--).. | |  

| | +-TIME--+-----+-----+-----+-----+  

| | | | .-(-6--).. | |  

| | '|-TIMESTAMP-----+-----+-----+  

+---+--XML-----+-----+-----+-----+-----+  

| | '|-ccsid-clause-' |  

| | | .-(-200--)-----.  

+---+--DATALINK-----+-----+-----+-----+-----+  

| | '|-(--integer--)-' | '|-ccsid-clause-' |  

+---+--ROWID-----+-----+-----+-----+-----+  

' ---+--XML-----'  


```

ccsid-clause

```
| --CCSID--integer-----|
```