

# **Použití SQL příkazů v RPG programu**

Vladimír Župka

# Obsah

<b>Obsah .....</b>	<b>2</b>
<b>Úvod .....</b>	<b>4</b>
<b>Vytvoření SQL objektů pro příklady .....</b>	<b>5</b>
<i>Vytvoření schematu (collection) čili knihovny.....</i>	<i>5</i>
<i>SQL skript pro vytvoření objektů ve schematu.....</i>	<i>5</i>
<b>Omezení kladená na tabulky .....</b>	<b>8</b>
<i>Omezení hodnot sloupců (check constraint).....</i>	<i>8</i>
<i>Unikátní klíč (unique key).....</i>	<i>8</i>
<i>Primární klíč (primary key) .....</i>	<i>8</i>
<i>Referenční integrita (referential integrity).....</i>	<i>8</i>
<i>SQL příkazy pro referenční integritu.....</i>	<i>9</i>
<i>Přidání referenčního omezení.....</i>	<i>9</i>
<i>Odebrání referenčního omezení .....</i>	<i>9</i>
<i>CL příkazy pro referenční integritu .....</i>	<i>9</i>
<i>Přidání referenčního omezení.....</i>	<i>9</i>
<i>Odebrání referenčního omezení .....</i>	<i>9</i>
<i>Změna stavu referenčního omezení .....</i>	<i>10</i>
<i>Spuštění a zastavení žurnálu .....</i>	<i>10</i>
<i>Pořadí CL příkazů pro žurnál a commitment.....</i>	<i>10</i>
<i>Omezení zobrazená v ACS.....</i>	<i>11</i>
<b>Použití SQL v jazyku RPG .....</b>	<b>12</b>
<i>Předkompilátor SQL pro jazyk RPG.....</i>	<i>12</i>
<i>Zápis příkazů v RPG programu.....</i>	<i>12</i>
<i>Statické SQL příkazy.....</i>	<i>13</i>
<i>Příkaz DECLARE s příkazem SELECT .....</i>	<i>13</i>
<i>Příkaz OPEN.....</i>	<i>13</i>
<i>Příkaz FETCH.....</i>	<i>13</i>
<i>Příkaz CLOSE.....</i>	<i>13</i>
<i>Příkaz UPDATE .....</i>	<i>13</i>
<i>Příkaz DELETE .....</i>	<i>13</i>
<i>Příkaz INSERT.....</i>	<i>13</i>
<i>Příklad: program STATSEL z knihovny VZSQLPGM, data z knihovny VZSQL .....</i>	<i>14</i>
<i>Dynamické SQL příkazy.....</i>	<i>15</i>
<i>Dynamický SELECT s pevným seznamem .....</i>	<i>15</i>
<i>Příklad: program DYNSELFIX z knihovny VZSQLPGM, data z knihovny VZSQL.....</i>	<i>15</i>
<i>Dynamický SELECT se značkami na místě proměnných .....</i>	<i>17</i>
<i>Příklad: program DYNSELMKF z knihovny VZSQLPGM, data z knihovny VZSQL.....</i>	<i>17</i>
<i>Dynamický SELECT s neznámým seznamem sloupců.....</i>	<i>18</i>
<i>Příklad: program DYNSELLSTF z knihovny VZSQLPGM, data z knihovny VZSQL .....</i>	<i>18</i>
<i>Ostatní dynamické příkazy.....</i>	<i>21</i>
<i>EXECUTE IMMEDIATE .....</i>	<i>21</i>
<i>PREPARE a EXECUTE se značkou .....</i>	<i>21</i>
<i>Testování výsledku SQL příkazů .....</i>	<i>22</i>
<i>Příklad: program TEST_CH, data z knihovny VZSQL .....</i>	<i>23</i>
<b>Srovnání tradičního přístupu s přístupem SQL v jazyku RPG .....</b>	<b>25</b>
<i>Popisy souborů – host variables .....</i>	<i>26</i>
<i>Datové struktury podle SQL tabulek a pohledů – host variables.....</i>	<i>26</i>
<i>Příkazy pro kompilátor - příkaz CTL-OPT .....</i>	<i>27</i>
<i>Příkazy pro předkompilátor - SQL příkaz SET OPTION .....</i>	<i>27</i>
<i>Plnění podsouboru hlaviček objednávek se jmény dodavatelů.....</i>	<i>28</i>
<i>Plnění podsouboru pro předchozí stránku zboží (po stisku Page Up) .....</i>	<i>29</i>

Výmaz hlavičky a všech odpovídajících detailů objednávky .....	30
Oprava detailních položek objednávky .....	31
Zápis nové hlavičky objednávky .....	32
<b>Rutiny SQL .....</b>	<b>33</b>
<b>Interní rutiny SQL .....</b>	<b>34</b>
SQL příkazy pro rutiny .....	34
Složený příkaz pro rutiny .....	34
Uložená procedura SQL (stored procedure) .....	34
Obecný tvar SQL procedury .....	34
Vytvoření procedury .....	35
Vyvolání procedury v RPG programu .....	37
Výsledek výpočtu .....	38
Uživatelská funkce SQL (UDF) .....	38
Obecný tvar uživatelské funkce SQL .....	38
Vytvoření tabulkové funkce .....	38
Volání tabulkové funkce v RPG programu .....	39
Spouštěč SQL (trigger) .....	41
Obecný tvar spouštěče .....	41
Příklady spouštěčů .....	41
Trigger BEFORE INSERT pro tabulku OBJDET_T .....	41
Trigger BEFORE UPDATE pro tabulku OBJDET_T .....	42
<b>Externí rutiny SQL .....</b>	<b>43</b>
Uložená procedura zapsaná v RPG .....	43
Uživatelská tabulková funkce zapsaná v RPG .....	45
Modul OBJDAT_F .....	46
Spouštěč zapsaný v RPG .....	49
Trigger BEFORE INSERT pro tabulku OBJDET_T .....	49
Trigger BEFORE UPDATE pro tabulku OBJDET_T .....	51
<b>Některé postupy při pořizování vydávaných faktur .....</b>	<b>53</b>
Převod dat ze starých souborů jiné knihovny .....	53
Číslování faktur .....	53
Získání ceny faktury z detailů do hlavičky .....	54
Odstranění osiřelých řádků před přidáním referenčního omezení .....	54
<b>Volba prostředí SQL v ACS a ve skriptu .....</b>	<b>55</b>
Nastavení jmenné konvence pro databázi .....	55
Nastavení českého prostředí pro SQL .....	57
<b>Opisy programů z příkladů .....</b>	<b>58</b>
Obrazovkový soubor OBJW .....	58
Program OBJ_RPG .....	63
Program OBJ_SQLF .....	77

# Úvod

Kurz je určen programátorům znalým jazyka RPG IV a databázového jazyka SQL. Aplikace psané v jazyku RPG obvykle přistupují k databázi prostřednictvím záznamů (record level access), např. READ, WRITE, SETLL, aj. Na rozdíl od tohoto přístupu jazyk SQL přistupuje k databázi prostřednictvím hromadných příkazů, např. SELECT, UPDATE, UNION, aj.

K překladači jazyka RPG (ale i dalších jazyků) existuje předkompilátor (preprocesor) dovolující zařazovat příkazy jazyka SQL a využívat tak veškeré jeho možnosti v aplikačním programu.

K tématu se vztahuje dokumentace IBM na stránce <https://www.ibm.com/docs/en/i/7.4?topic=programming-embedded-sql>. Další informace ke kurzu jsou v referenční příručce SQL na stránce <https://www.ibm.com/docs/en/i/7.4?topic=reference-sql>.

V kurzu se uvádí nejprve informace o jazyku SQL (vytvoření objektů a omezení kladená na tabulky) Použití v RPG programech je pak uváděno na komplexním příkladu, kde se srovnává tradiční přístup RPG s přístupem SQL. Nakonec se kurz věnuje tzv. rutinám SQL (stored procedure, user defined function, trigger), jednak vlastním - psaným v SQL, jednak externím - psaným v RPG.

Příklady RPG programů jsou umístěny v knihovně VZSQLPGM a data jsou v knihovně VZSQL.

# Vytvoření SQL objektů pro příklady

Všechny SQL objekty byly vytvořeny SQL skriptem ve jmenné konvenci \*SYS, která byla nastavena v grafickém prostředí programu ACS (*IBM i Access - Client Solutions*) – viz [dole](#).

## Vytvoření schematu (collection) čili knihovny

```
CREATE SCHEMA VZSQL WITH DATA DICTIONARY;
```

```
-- DROP SCHEMA VZSQL CASCADE;
```

## SQL skript pro vytvoření objektů ve schematu

Objekty pocházejí z knihovny VZRP GPOKR, kde byly původně vytvořeny jako tradiční soubory. Z této knihovny byl vytvořen text skriptu pomocí programu ACS, volby *Generovat SQL...* Výsledný text byl následně upraven a použit k vytvoření objektů SQL knihovny VZSQL, které jsou použity v příkladech.

Poznámka: Ve skutečnosti to v roce 2013 nebyl program ACS, ale program Navigator v systému Windows 7, který již neexistuje.

```
SET SCHEMA 'VZSQL'; -- příkaz dovoluje vynechat kvalifikaci objektů jménem schematu
```

```
-- Tabulka cen zboží od dodavatelů
```

```
-- -----
```

```
CREATE TABLE CENYD_T (  
  CDOD CHAR(6) NOT NULL DEFAULT '' ,  
  CZBOZID CHAR(5) NOT NULL DEFAULT '' ,  
  CENAJ DECIMAL(9, 2) NOT NULL DEFAULT 0 ,  
  NAZZBO CHAR(30) NOT NULL DEFAULT '' ,  
  SAZBA_DPH INTEGER NOT NULL DEFAULT 0 )  
  RCDFMT CENYD_T ;  
LABEL ON TABLE CENYD_T IS 'Ceník zboží od dodavatelů' ;  
LABEL ON COLUMN CENYD_T  
( CZBOZID IS 'Číslo          zboží.' ,  
  CENAJ IS 'Cena/j.' ,  
  NAZZBO IS 'Název zboží' ,  
  SAZBA_DPH IS 'Sazba          DPH' ) ;
```

```
-- Tabulka procent a sazeb DPH
```

```
-- -----
```

```
CREATE TABLE DPH_T (  
  PROC_DPH INTEGER NOT NULL DEFAULT 0 ,  
  SAZBA_DPH INTEGER NOT NULL DEFAULT 0 )  
  RCDFMT DPH_T ;  
LABEL ON TABLE DPH_T IS 'Daň z přidané hodnoty' ;  
LABEL ON COLUMN DPH_T  
( PROC_DPH IS 'Procento          DPH' ,  
  SAZBA_DPH IS 'Sazba          DPH' ) ;
```

```
-- Tabulka dodavatelů
```

```
-- -----
```

```
CREATE TABLE DODAV_T (  
  CDOD CHAR(6) NOT NULL DEFAULT '' ,  
  NAZDOD CHAR(30) NOT NULL DEFAULT '' ,  
  ADRDOD CHAR(20) NOT NULL DEFAULT '' )  
  RCDFMT DODAV_T ;
```

```

LABEL ON TABLE DODAV_T IS 'Dodavatelé materiálu' ;
LABEL ON COLUMN DODAV_T
( CDOD IS 'Číslo          dodav.' ,
  NAZDOD IS 'Název dodavatele' ,
  ADRDOD IS 'Adresa dodavatele' ) ;

-- Tabulka detailů objednávek
-- -----

CREATE TABLE OBJDET_T (
  COBJ CHAR(6) NOT NULL DEFAULT '' ,
  CDOD CHAR(6) NOT NULL DEFAULT '' ,
  CZBOZID CHAR(5) NOT NULL DEFAULT '' ,
  MNOBJ DECIMAL(9, 0) NOT NULL DEFAULT 0 )
RCDFMT OBJDET_T ;
LABEL ON TABLE OBJDET_T IS 'Objednávky - detaily' ;
LABEL ON COLUMN OBJDET_T
( COBJ IS 'Číslo          objed.' ,
  CZBOZID IS 'Číslo          zboží.' ,
  MNOBJ IS 'Množství       objed.' ) ;

-- Tabulka hlaviček objednávek
-- -----

CREATE TABLE OBJHLA_T (
  COBJ CHAR(6) NOT NULL DEFAULT '' ,
  CDOD CHAR(6) NOT NULL DEFAULT '' ,
  DTOBJ DATE NOT NULL WITH DEFAULT )
RCDFMT OBJHLA_T ;
LABEL ON TABLE OBJHLA_T IS 'Objednávky - hlavičky' ;
LABEL ON COLUMN OBJHLA_T
( COBJ IS 'Číslo          objed.' ,
  CDOD IS 'Číslo          dodav.' ,
  DTOBJ IS 'Datum          RRRR-MM-DD' ) ;

-- Pohledy na tabulky
-- -----

CREATE VIEW CENYD AS SELECT CDOD, CZBOZID, CENAJ, NAZZBO FROM CENYD_T ;
LABEL ON TABLE CENYD IS 'Ceník zboží od dodavatelů';

CREATE VIEW DODAV AS SELECT CDOD, NAZDOD, ADRDOD FROM DODAV_T ;
LABEL ON TABLE DODAV IS 'Dodavatelé materiálu' ;

CREATE VIEW OBJDET AS SELECT COBJ, CDOD, CZBOZID, MNOBJ FROM OBJDET_T ;
LABEL ON TABLE OBJDET IS 'Objednávky - detaily' ;

CREATE VIEW OBJHLA AS SELECT COBJ, CDOD, DTOBJ FROM OBJHLA_T ;
LABEL ON TABLE OBJHLA IS 'Objednávky - hlavičky' ;

-- Indexy s unikátními klíči podle čísel
-- -----

CREATE UNIQUE INDEX CENYD_IX ON CENYD_T (CDOD, CZBOZID) PAGESIZE 64 ;
LABEL ON INDEX CENYD_IX IS 'Index k cenám dodavatelů podle čísla';

CREATE UNIQUE INDEX DODAV_IX ON DODAV_T (CDOD) PAGESIZE 64 ;
LABEL ON INDEX DODAV_IX IS 'Index k dodavatelům podle čísla';

CREATE UNIQUE INDEX OBJDET_IX ON OBJDET_T (COBJ, CZBOZID) PAGESIZE 64 ;
LABEL ON INDEX OBJDET_IX IS 'Index k detailům objednávek';

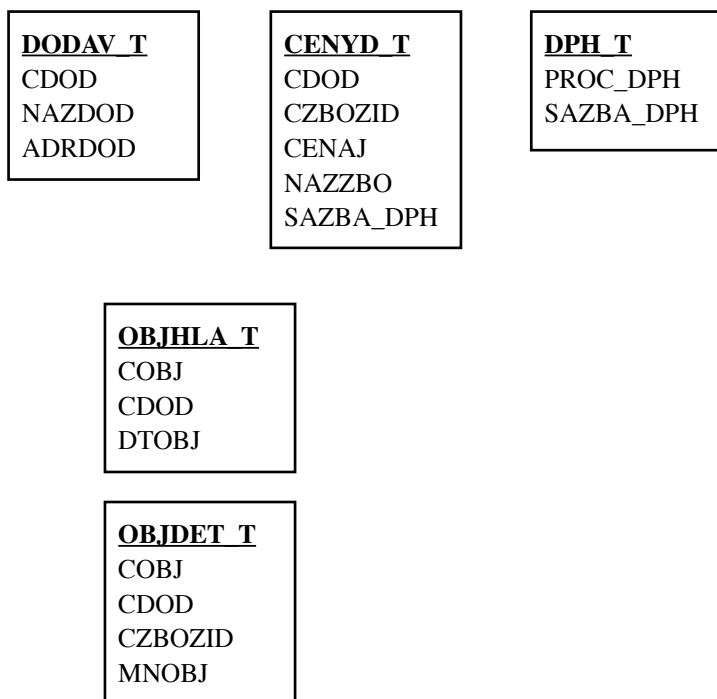
```

```
CREATE UNIQUE INDEX OBJHLA_IX ON OBJHLA_T (COBJ) PAGESIZE 64 ;
LABEL ON INDEX OBJHLA_IX IS 'Index k hlavičkám objednávek';
```

```
-- Indexy podle názvů
-- -----
```

```
CREATE INDEX CENYDN_IX ON CENYD_T (NAZZBO) PAGESIZE 64 ;
LABEL ON INDEX CENYDN_IX IS 'Index k cenám dodavatelů podle názvu';
```

```
CREATE INDEX DODAVN_IX ON DODAV_T (NAZDOD) PAGESIZE 64 ;
LABEL ON INDEX DODAVN_IX IS 'Index k dodavatelům podle názvu';
```



# Omezení kladená na tabulky

V SQL lze klást na tabulky jistá omezení, buď na jednotlivé tabulky nebo na skupinu tabulek. U jednotlivé tabulky jde o omezení hodnot sloupce nebo skupiny sloupců a o omezení na unikátní hodnoty klíčů. U skupiny tabulek jde o tzv. referenční integritu za pomoci “cizího klíče” a “rodičovského klíče”.

## Omezení hodnot sloupců (*check constraint*)

Příklad: cena za jednotku v tabulce CENYD\_T nesmí překročit hodnotu 9999.00.

```
ALTER TABLE CENYD_T ADD CONSTRAINT CST_CENAJ CHECK ( CENAJ <= 9999.00 ) ;
```

## Unikátní klíč (*unique key*)

V tabulce se definuje sloupec nebo skupina sloupců jako klíč. Hodnoty klíče musí být v tabulce unikátní (jednoznačné). Žádné dva řádky nesmí mít stejnou hodnotu klíče.

```
ALTER TABLE OBJHLA_T ADD CONSTRAINT OBJHLA_UNQ UNIQUE (COBJ) ;  
ALTER TABLE OBJDET_T ADD CONSTRAINT OBJDET_UNQ UNIQUE (COBJ, CZBOZID) ;  
ALTER TABLE DODAV_T ADD CONSTRAINT DODAV_UNQ UNIQUE (CDOD) ;  
ALTER TABLE CENYD_T ADD CONSTRAINT CENY_UNQ UNIQUE (CDOD, CZBOZID) ;
```

Omezení lze odebrat těmito příkazy:

```
ALTER TABLE OBJHLA_T DROP CONSTRAINT OBJHLA_UNQ ;  
ALTER TABLE OBJDET_T DROP CONSTRAINT OBJDET_UNQ ;  
ALTER TABLE DODAV_T DROP CONSTRAINT DODAV_UNQ ;  
ALTER TABLE CENY_T DROP CONSTRAINT CENY_UNQ ;
```

## Primární klíč (*primary key*)

Primární klíč se liší od unikátního klíče jen tím, že neumožňuje prázdné (null) hodnoty svých sloupců. Místo slova UNIQUE je zapsáno PRIMARY KEY. Nejdříve se však musí odstranit předchozí omezení unikátního klíče OBJHLA\_UNQ.

```
ALTER TABLE OBJHLA_T DROP CONSTRAINT OBJHLA_UNQ ;  
ALTER TABLE OBJHLA_T ADD CONSTRAINT OBJHLA_PRIM_KEY PRIMARY KEY (COBJ) ;
```

## Referenční integrita (*referential integrity*)

Hlavičky objednávky odkazují na tabulku DODAV\_T číslem dodavatele CDOD a detaily objednávky odkazují na tabulku CENYD\_T číslem dodavatele CDOD a číslem zboží CZBOZID.

Aplikační program vytvoří objednávku pro existujícího dodavatele a jeho existující druhy zboží. Někdy je třeba z databáze vyřadit určitého dodavatele nebo druh zboží. Použijí-li se k tomu účelu aplikační programy pro údržbu dodavatelů a zboží, které při vyřazení záznamu odstraní také odpovídající záznam objednávky, popř. celou objednávku, je vše v pořádku. Udrží se tak integrita aplikace.

Použije-li však někdo k vyřazení jiný nástroj, který nebere ohled na objednávky, např. program DFU, nebo interakční SQL, naruší se integrita tím, že v souborech objednávek zůstane osamocená objednávka - sirotek.

Tomu lze zabránit tak, že číslo dodavatele a číslo zboží označíme jako *cizí klíč* (*foreign key*) a zařadíme je přímo do tabulky jako *omezení* (*constraint*). Lze to udělat při vytvoření tabulky nebo doda-



tečně příkazem ALTER TABLE (ale také CL příkazem ADDPFCST - Add Physical File Constraint).

Jeden druh referenčního omezení spočívá v tom, že při zrušení řádku zboží se automaticky zruší i příslušné detailní řádky objednávky. Při zrušení řádku dodavatele se automaticky zruší hlavička objednávky a všechny detaily. Říká se tomu *kaskádové* rušení (cascade delete). Jiný druh je *restrikce*, kdy zrušit nebo změnit řádek nelze.

Poznámka: Tabulka obsahující cizí klíč se nazývá *závislá* (dependent); tabulka, na niž cizí klíč odkazuje, se nazývá *rodičovská* (parent) tabulka. Podmínkou pro existenci referenčního omezení je, aby rodičovská tabulka měla odpovídající *unikátní* nebo *primární* klíč. Nestačí existence unikátního indexu.

## SQL příkazy pro referenční integritu

### Přidání referenčního omezení

V tabulce OBJHLA\_T definujeme jedno omezení nazvané DODAV\_EXISTUJE.

```
ALTER TABLE OBJHLA_T ADD CONSTRAINT DODAV_EXISTUJE
  FOREIGN KEY (CDOD) REFERENCES DODAV_T (CDOD)
  ON DELETE CASCADE ON UPDATE NO ACTION ;
```

V tabulce OBJDET\_T definujeme dvě omezení:

ZBOZI\_DODAV\_EXISTUJE a HLAV\_OBJEDN\_EXISTUJE.

```
ALTER TABLE OBJDET_T ADD CONSTRAINT ZBOZI_DODAV_EXISTUJE
  FOREIGN KEY (CDOD, CZBOZID) REFERENCES CENYD_T (CDOD, CZBOZID)
  ON DELETE CASCADE ON UPDATE NO ACTION ;
ALTER TABLE OBJDET_T ADD CONSTRAINT HLAV_OBJEDN_EXISTUJE
  FOREIGN KEY (COBJ) REFERENCES OBJHLA_T (COBJ)
  ON DELETE CASCADE ON UPDATE NO ACTION ;
```

### Odebrání referenčního omezení

```
ALTER TABLE OBJHLA_T DROP FOREIGN KEY DODAV_EXISTUJE CASCADE ;
ALTER TABLE OBJDET_T DROP FOREIGN KEY ZBOZI_DODAV_EXISTUJE CASCADE ;
ALTER TABLE OBJDET_T DROP FOREIGN KEY HLAV_OBJEDN_EXISTUJE CASCADE ;
```

Poznámka: U operace DELETE i UPDATE lze volit druh omezení RESTRICT nebo NO ACTION (default), což znamená totéž, a sice že nastane chyba a akce se neprovede.

## CL příkazy pro referenční integritu

Stejného účinku lze dosáhnout také CL příkazy.

### Přidání referenčního omezení

```
ADDPFCST  FILE(OBJHLA_T) TYPE(*REFCST) KEY(CDOD)
          CST(DODAV_EXISTUJE) PRNFILE(DODAV_T) PRNKEY(CDOD)
          DLTRULE(*CASCADE) UPDRULE(*NOACTION)
ADDPFCST  FILE(OBJDET_T) TYPE(*REFCST) KEY(CDOD CZBOZI)
          CST(ZBOZI_DODAV_EXISTUJE) PRNFILE(CENY_T) PRNKEY(CDOD CZBOZI)
          DLTRULE(*CASCADE) UPDRULE(*NOACTION)
ADDPFCST  FILE(OBJDET_T) TYPE(*REFCST) KEY(COBJ)
          CST(HLAV_OBJEDN_EXISTUJE) PRNFILE(OBJHLA_T) PRNKEY(COBJ)
          DLTRULE(*CASCADE) UPDRULE(*NOACTION)
```

### Odebrání referenčního omezení

```
RMVPFCST  FILE(OBJHLA_T) CST(DODAV_EXISTUJE) TYPE(*REFCST)
```

```
RMVFCST FILE(OBJDET_T) CST(ZBOZI_DODAV_EXISTUJE) TYPE(*REFCST)
RMVFCST FILE(OBJDET_T) CST(HLAV_OBJEDN_EXISTUJE) TYPE(*REFCST)
```

## Změna stavu referenčního omezení

Referenční omezení lze zneschopnit (disable) nebo naopak uschopnit (enable), a to v programu ACS nebo pomocí CL příkazů

```
CHGFCST FILE(OBJHLA_T) CST(HLAV_OBJEDN_EXISTUJE) STATE(*DISABLED)
CHGFCST FILE(OBJHLA_T) CST(HLAV_OBJEDN_EXISTUJE) STATE(*ENABLED)
...
```

Můžeme také použít CL příkaz

```
WRKFCST FILE(VZSQL/*ALL) TYPE(*REFCST)
```

Work with Physical File Constraints						
Type options, press Enter.						
2=Change 4=Remove 6=Display records in check pending						
Opt	Constraint	File	Library	Type	State	Check Pending
—	ZBOZI_DODA	> OBJDET_T	VZSQL	*REFCST	EST/ENB	NO
—	HLAV_OBJED	> OBJDET_T	VZSQL	*REFCST	EST/ENB	NO
—	DODAV_EXIS	> OBJHLA_T	VZSQL	*REFCST	EST/ENB	NO

## Spuštění a zastavení žurnálu

K použití *kaskádové* referenční integrity pomocí cizího klíče (foreign key) je *zapotřebí spuštěný žurnál* pro rodičovskou i závislou tabulku. Zastavíme-li např. žurnál rodičovské tabulky, databázový systém SQL na ní zabráni operacím UPDATE a DELETE. Nedovolí ani spustit program DFU (UPDDTA).

Protože se jedná o transakce, které musí proběhnout bez přerušení, je *zapotřebí i potvrzování transakcí (commitment control)*. K provozu stačí spustit jen žurnálování; *řízení transakcí není třeba spouštět*, zahajuje a ukončuje se automaticky.

## Pořadí CL příkazů pro žurnál a commitment

- Spustit žurnál QSQRN pro čtyři tabulky

```
STRJRNPF FILE(OBJHLA_T OBJDET_T CENYD_T DODAV_T) JRN(VZSQL/QSQJRN)
IMAGES(*BOTH)
```

- Spustit commitment control (řízení transakcí je nepovinné)

```
STRCMTCTL LCKLVL(*CHG)
```

- Zastavit commitment control

```
ENDCMTCTL
```

- Zastavit žurnálování čtyř tabulek

```
ENDJRNPF FILE(OBJHLA_T OBJDET_T CENYD_T DODAV_T) JRN(VZSQL/QSQJRN)
```

- Změnit číslování přijímače a zrušit starý přijímač

```
CHGJRN JRN(VZSQL/QSQJRN)
JRNRCV(*GEN)
```

SEQOPT (\***RESET**)  
DLTRCV (\***YES**)  
JRNSTATE (\***SAME**/\*ACTIVE/\*INACTIVE)

## Omezení zobrazená v ACS

Schémat - 192.168.1.93

Soubor Upravit Zobrazit Akce Nástroje

Databáze

- C2074a0w
  - Schémat
    - QGPL
    - VZRPPOKR
    - VZSQL
      - Všechny databázové objekty
      - Alias
      - Balíky SQL
      - Funkce
      - Globální proměnné
      - Indexy
      - Maska sloupců
      - Omezení**
      - Oprávnění řádků
      - Posloupnosti
      - Procedury
      - Spouštěče
      - Tabulky
      - Typy
      - Zobrazení
      - Úložiště schémat XML (XSR)
      - Žurnálové zásobníky
      - Žurnály
    - VZSQLPGM
    - Údržba databáze
    - Transakce

Databáze ► C2074a0w ► Schémata ► VZSQL ► Omezení

Název	Typ	Název tabulky	Stav	Povoleno
CENY_UNQ	Omezení jedinečného klíče	CENYD_T	Zavedené	Ano
CST_CENAJ	Kontrolní omezení	CENYD_T	Zavedené	Ano
DODAV_EXISTUJE	Omezení cizího klíče	OBJHLA_T	Zavedené	Ano
DODAV_UNQ	Omezení jedinečného klíče	DODAV_T	Zavedené	Ano
HLAV_OBJEDN_EXISTUJE	Omezení cizího klíče	OBJDET_T	Definováno	
OBJDET_UNQ	Omezení jedinečného klíče	OBJDET_T	Zavedené	Ano
OBJHLA_PRIM_KEY	Omezení primárního klíče	OBJHLA_T	Zavedené	Ano
<b>ZBOZI_DODAV_EXISTUJE</b>	<b>Omezení cizího klíče</b>	<b>OBJDET_T</b>	<b>Zavedené</b>	<b>Ano</b>

Done: 8 rows retrieved.

# Použití SQL v jazyku RPG

## Předkompilátor SQL pro jazyk RPG

Zdrojový typ programu je SQLRPGLE. Příkaz volby 14 i 15 pro kompilaci je CRTSQLRPGI (Create SQL ILE RPG Object). Uvádíme některé parametry předkompilátoru.

```
CRTSQLRPGI OBJ(*CURLIB/OBJ_SQL) SRCFILE(*LIBL/QRPGLESRC) SRCMBR(*OBJ)
  COMMIT(*CHG) *NONE *ALL ...
  OBJTYPE(*PGM) *MODULE *SRVPGM
  TGTRLS(V5R3M0) V6R1M0
  CLOSQLCSR(*ENDMOD) *ENDACTGRP kdy se uzavře kurzor
  DATFMT(*JOB) *ISO ...
  DATSEP(*JOB) / . , - *BLANK
  TIMFMT(*JOB) *ISO ...
  TIMSEP(*JOB) : . , *BLANK
  DFTRDBCOL(*NONE) VZSQL ... předvolené SQL schema
  DYNDFTCOL(*NO) *YES - DFTRDBCOL také pro dynamické příkazy
  DBGVIEW(*NONE) *SOURCE
  SRTSEQ(*JOB) CSY ...
  LANGID(*JOB) *LANGIDSHR ...
  TOSRCFILE(QTEMP/QSQLTEMP1) výstup zdroje - při OPTION(*NOGEN) zůstává
  DECRESULT(31 31) max. počet číslic, des. míst
  OPTION(*SYS/*SQL
    *JOB/*SYSVAL/*PERIOD/*COMMA
    ...)
```

Příkaz SET OPTION představuje alternativní a doplňkové volby pro SQL příkazy. Některé jsou shodné s parametry předkompilátoru. Volby jsou odděleny čárkou.

```
SET OPTION
  COMMIT = *CHG *NONE *ALL ...
  DATFMT = *JOB *ISO ...
  DATSEP = *JOB *PERIOD *COMMA *DASH *BLANK
  DFTRDBCOL = *NONE 'VZSQL'
  LANGID = *JOB *JOBRUN CSY ENU DEU ...
  NAMING = *SYS *SQL
  SRTSEQ = *JOB *HEX *JOBRUN *LANGIDUNQ *LANGIDSHR
  TIMFMT = *HMS *ISO *EUR *USA *JIS
  TIMSEP = *JOB *COLON *PERIOD *COMMA *BLANK
```

SQL příkaz SET OPTION musí být (je-li použit) zapsán před prvním výkonným SQL příkazem. Příklad:

```
Exec SQL set option LANGID = CSY, SRTSEQ = *LANGIDSHR,
          DATFMT = *ISO, COMMIT = *NONE ;
```

## Zápis příkazů v RPG programu

Ve volném formátu (/free) je tvar SQL příkazu

```
EXEC SQL příkaz ;
```

V pevném formátu RPG III i RPG IV je tvar SQL příkazu

```
C/EXEC SQL
C+ příkaz
C/END-EXEC
```

## Statické SQL příkazy

Statické SQL příkazy jsou ty, které při výpočtu mají stejnou formu jako při kompilaci. K výpočtu je připraví (a předběžně optimalizuje) předkompilátor.

Příkazy s výhodou používají jména tabulek/pohledů (views) *bez kvalifikace* jménem schematu. Při standardní volbě OPTION(\*SYS) se jméno schematu (knihovny) dosadí ze seznamu knihoven (\*LIBL).

Jména proměnných uvozená dvojtečkou – například :ZNAZD – se nazývají *host variables* a slouží k zadání hodnot pro SQL příkazy nebo příjmu hodnot z výsledku SQL příkazů.

Výsledek příkazu je vyjádřen kódem v proměnné SQLSTATE (standard ISO) nebo SQLCODE (IBM).

### Příkaz DECLARE s příkazem SELECT

```
Exec SQL declare CS3 scroll cursor for
      select CDOD, NAZDOD, ADRDOD from DODAV
      where NAZDOD >= :ZNAZD
      order by NAZDOD asc
      fetch first 7 rows only;
```

deklaruje *nastavovací (scroll)* kurzor pro dotazovací příkaz SELECT.

### Příkaz OPEN

```
Exec SQL open CS3;
```

otevře kurzor, tj. provede příkaz SELECT.

### Příkaz FETCH

```
Exec SQL fetch first from CS3 into :CDOD, :NAZDOD, :ADRDOD;
```

přečte první záznam výsledné tabulky, kterou vyprodukuje příkaz SELECT, je-li kurzor označen jako SCROLL.

### Příkaz CLOSE

```
Exec SQL close CS3;
```

uzavře kurzor.

### Příkaz UPDATE

```
Exec SQL update OBJHLA
      set OBJHLA.CDOD = :CDOD,
        OBJHLA.DTOBJ = :DTOBJ
      where COBJ = :COBJ;
```

změní dva sloupce v řádku hlavičky objednávky.

### Příkaz DELETE

```
Exec SQL delete from OBJDET where COBJ = :COBJ;
```

Hromadný příkaz ruší všechny detailní řádky se zadaným číslem objednávky.

### Příkaz INSERT

```
Exec SQL insert into OBJHLA ( COBJ, CDOD, DTOBJ )
      values ( :COBJ, :CDOD, :DTOBJ );
```

zapiše jeden záznam hlavičky objednávky do tabulky přes pohled OBJHLA.

Příklad: program STATSEL z knihovny VZSQLPGM, data z knihovny VZSQL  
Program přečte tabulku dodavatelských cen a všechny řádky vypíše příkazem DSPLY.

```
**free

DCL-DS *N ExtName('*LIBL/CENYD_T') END-DS;  // host variables

Exec SQL declare CS cursor for
      select CDOD, CZBOZID, CENAJ, NAZZBO
      from   CENYD_T
      order by CDOD, CZBOZID ;
Exec SQL open CS;
Exec SQL fetch from CS into :CDOD, :CZBOZID, :CENAJ, :NAZZBO ;

dow sqlstate < '02000';

      dsply (CDOD + ' ' + CZBOZID + ' ' + %char(CENAJ) + ' '
            + %subst(NAZZBO: 1: 25));

      Exec SQL fetch from CS into :CDOD, :CZBOZID, :CENAJ, :NAZZBO ;

enddo;

Exec SQL close CS;

return;
```

## Dynamické SQL příkazy

Dynamické SQL příkazy jsou ty, které jsou umístěny ve znakové proměnné a zpracují se až v době výpočtu. Zpracování spočívá v přípravě, optimalizaci a provedení příkazu.

Poznámka: Příprava a předběžná optimalizace u statických příkazů se děje v době kompilace.

Dynamické zpracování příkazu SELECT se liší od dynamického zpracování ostatních příkazů.

### Dynamický SELECT s pevným seznamem

Dynamický příkaz SELECT s *pevným seznamem* sloupců se zpracovává podobně jako statický, ale s použitím příkazu PREPARE.

Příklad: program DYNSELFIX z knihovny VZSQLPGM, data z knihovny VZSQL

```
Ctl-Opt dftactgrp(*no) actgrp(*new) decedit('0,');

Dcl-F QPRINT PRINTER(200) OFLIND(*INOA);

//   Paměť pro sestavení příkazu SELECT
Dcl-S SQL_STATEMENT          Char(500) INZ;

//   Datové struktury pro databázové záznamy podle pohledů na tabulky
Dcl-DS CENYD                  EXT INZ;
      CDODCEN                  EXTFLD('CDOD');
End-DS;
Dcl-DS DODAV                  EXT INZ;
End-DS;
Dcl-DS OBJDET                  EXT INZ;
      COBJDET                  EXTFLD('COBJ');
      CDODDET                  EXTFLD('CDOD');
      CZBOZIDET                EXTFLD('CZBOZID');
End-DS;
Dcl-DS OBJHLA                 EXT INZ;
      CDODOBJ                  EXTFLD('CDOD');
End-DS;

//   Pracovní proměnné
Dcl-S CENAC                    Packed(12:2);

Exec SQL set option DATFMT = *ISO, COMMIT = *NONE;

SQL_STATEMENT =
'select H.COBJ, H.CDOD, DTOBJ, D.CZBOZID, MNOBJ, CENAJ, +
      NAZZBO, NAZDOD, ADRDOD, (CENAJ * MNOBJ) as CENA_CELKEM +
from OBJHLA as H +
join OBJDET as D on  H.COBJ = D.COBJ +
      and H.CDOD = D.CDOD +
join CENYD  as C on  D.CDOD = C.CDOD +
      and D.CZBOZID = C.CZBOZID +
join DODAV as DOD on H.CDOD = DOD.CDOD +
where (CENAJ * MNOBJ) between 0.00 and 99999.00 +
      and DTOBJ <=  '2022-12-31' +
order by COBJ asc +
For read only';

Exec SQL prepare SEL from :SQL_STATEMENT ;
Exec SQL declare CUR cursor for SEL;
```

```

Exec SQL open CUR;

// přečíst a zpracovat data z dotazu
Exec SQL
    fetch CUR into :COBJ, :CDOD, :DTOBJ, :CZBOZID, :MNOBJ,
                    :CENAJ, :NAZZBO, :NAZDOD, :ADRDOD, :CENAC;

// Tisknout nadpis sloupců
Except NADPIS;

// Přečíst a vytisknout výslednou tabulku v cyklu
DoW sqlstate < '02000'; // dokud jsou nějaké záznamy

    // Tisknout vybrané údaje přečteného řádku tabulky
    If *inOA;
        Except NADPIS;
        *inOA = *off;
    EndIf;
    Except DETAIL;

    // Přečíst další záznam výsledné tabulky
    Exec SQL
        fetch CUR into :COBJ, :CDOD, :DTOBJ, :CZBOZID, :MNOBJ,
                        :CENAJ, :NAZZBO, :NAZDOD, :ADRDOD, :CENAC;

EndDo;

Exec SQL close CUR;

*inlr = *on;

```

```

OQPRINT      E              NADPIS              2
O              4 'COBJ'
O              12 'DTOBJ'
O              22 'CDOD'
O              31 'NAZDOD'
O              61 'CZBOZID'
O              72 'MNOBJ'
O              84 'CENAJ'
O              99 'CENAC'
O             106 'NAZZBO'
OQPRINT      E              DETAIL              1
O              COBJ
O              DTOBJ              + 1
O              CDOD              + 1
O              NAZDOD              + 1
O              CZBOZID              + 1
O              MNOBJ              Q + 1
O              CENAJ              Q + 1
O              CENAC              Q + 1
O              NAZZBO              + 1

```

Výsledky jsou znázorněny v následujícím výpisu.

COBJ	DTOBJ	CDOD	NAZDOD	CZBOZID	MNOBJ	CENAJ	CENAC NAZZBO
000001	2020-09-20	000006	Hračky v.o.s.	01062	4	557,00	2228,00 Koloběžka střední bez brzdy
000001	2020-09-20	000006	Hračky v.o.s.	01063	12	357,00	4284,00 Koloběžka malá bez brzdy
000002	2014-02-27	000003	Laktos mlékárna	00023	40	598,00	23920,00 Mléko pasteurisované tučné
000002	2014-02-27	000003	Laktos mlékárna	00239	30	569,00	17070,00 Jogurt bílý plnotučný
...							



## ***Dynamický SELECT se značkami na místě proměnných***

Jde o příkaz s pevným seznamem sloupců, ale v textu příkazu SELECT jsou otazníky (značky, markers). Značka může být obecně tam, kde může být programová proměnná (host variable). Zde značky stojí na místě hodnot, které dosadíme v příkazu OPEN ze tří proměnných CENA\_OD, CENA\_DO, DATUM\_OBJ. Pořadí proměnných v příkazu OPEN musí odpovídat pořadí otazníků v příkazu SELECT.

Příklad: program DYNSELMKF z knihovny VZSQLPGM, data z knihovny VZSQL

```
... jako u DYNSELFIX

// Pracovní proměnné
Dcl-S Cena_od          Packed(18:2) Inz(0.00);
Dcl-S Cena_do          Packed(18:2) Inz(99999.00);
Dcl-S Datum_obj       Date Inz(D'2022-12-31');
Dcl-S CENAC            Packed(12:2);

Exec SQL set option DATFMT = *ISO, COMMIT = *NONE;

SQL_STATEMENT =
'select H.COBJ, H.CDOD, DTOBJ, D.CZBOZID, MNOBJ, CENAJ, +
      NAZZBO, NAZDOD, ADRDOD, (CENAJ * MNOBJ) as CENA_CELKEM +
from OBJHLA as H +
join OBJDET as D on  H.COBJ = D.COBJ +
                  and H.CDOD = D.CDOD +
join CENYD  as C on  D.CDOD  = C.CDOD +
                  and D.CZBOZID = C.CZBOZID +
join DODAV as DOD on H.CDOD = DOD.CDOD +
where (CENAJ * MNOBJ) between ? and ? +
      and DTOBJ <= ? +
order by COBJ asc +
For read only';

Exec SQL prepare SEL from :SQL_STATEMENT ;
Exec SQL declare CUR cursor for SEL;
Exec SQL open CUR using :Cena_od, :Cena_do, :Datum_obj;

// přečíst a zpracovat data z dotazu
Exec SQL
      fetch CUR into :COBJ, :CDOD, :DTOBJ, :CZBOZID, :MNOBJ,
                    :CENAJ, :NAZZBO, :NAZDOD, :ADRDOD, :CENAC;

... jako u DYNSELFIX
```

Výsledky jsou znázorněny v následujícím výpisu.

COBJ	DTOBJ	CDOD	NAZDOD	CZBOZID	MNOBJ	CENAJ	CENAC NAZZBO
000001	2020-09-20	000006	Hračky v.o.s.	01063	12	357,00	4284,00 Koloběžka malá bez brzdy
000001	2020-09-20	000006	Hračky v.o.s.	01062	4	557,00	2228,00 Koloběžka střední bez brzdy
000002	2014-02-27	000003	Laktos mlékárna	00239	30	569,00	17070,00 Jogurt bílý plnotučný
000002	2014-02-27	000003	Laktos mlékárna	00023	40	598,00	23920,00 Mléko pasteurisované tučné
...							

## **Dynamický SELECT s neznámým seznamem sloupců**

Dynamický příkaz SELECT s neznámým seznamem sloupců používá informace obsažené v systémové struktuře “SQL descriptor”. Existuje několik variant použití této struktury, z nichž použijeme datovou strukturu v programu výslovně pojmenovanou **SQLDA** – *SQL descriptor area*. Použijeme příkazy

```
exec SQL prepare SEL from :SQL_STATEMENT ;
exec SQL describe SEL into :SQLDA;
exec SQL fetch next from CUR using descriptor :SQLDA ;
```

Příklad: program DYNSELLSTF z knihovny VZSQLPGM, data z knihovny VZSQL

Tento program funguje tak, že nezná sloupce v příkazu SELECT. Všechny informace o nich zjistí teprve z datové struktury SQLDA. Pak vytiskne zjištěné názvy sloupců a jejich data. Text příkazu je zde sice zapsán v programu, ale mohl by být přijat třeba jako vstupní parametr.

```
Ctl-Opt dftactgrp(*no) actgrp(*new) decedit('0,');

Dcl-F QPRINT PRINTER(200) OFLIND(*INOA);

// Pomocné proměnné
Dcl-S DataSpace Char(1000);
Dcl-S Data Char(1000) Based(Ptr);
Dcl-S Ptr Pointer;
Dcl-S SQL_STATEMENT Char(500);
Dcl-S CenaProTisk Packed(12:2);
Dcl-S Idx Int(10:0);
Dcl-C MaxPocet Const(20);
Dcl-S JmenoSloupce Char(12);
Dcl-S Hodnota VarChar(100);
Dcl-S Digits Int(10:0);
Dcl-S DecPos Int(10:0);
Dcl-S Size Int(10:0);
// Překrytí pakovaného čísla se znaky
Dcl-DS *N;
    Packed Packed(39:9) INZ;
    PackedChar Char(20) OVERLAY(Packed);
End-DS;

// SQL Descriptor Area
Dcl-DS SQLDA ALIGN QUALIFIED INZ;
    SQLDAID Char(8);
    SQLDABC Int(10:0);
    SQLN Int(5:0) Inz(MaxPocet);
    SQLD Int(5:0);
    SQL_VAR DIM(MaxPocet) LIKEDS(SQLVAR);
End-DS;

// Šablona datové struktury SQL_VAR pro proměnné
Dcl-DS SQLVAR TEMPLATE;
    SQLTYPE Int(5:0);
    SQLLEN Int(5:0);
    SQLRES Char(12);
    SQLDATA Pointer;
    SQLIND Pointer;
    SQLNAME VarChar(30);
End-DS;

Exec SQL set option DATFMT = *ISO, COMMIT = *NONE;

SQL_STATEMENT =
    'select H.COBJ, H.CDOD, DTOBJ, D.CZBOZID, MNOBJ, CENAJ, +
```

```

        NAZZBO, NAZDOD, ADRDOD, (CENAJ * MNOBJ) as CENA_CELKEM +
from OBJHLA as H +
join OBJDET as D on H.COBJ = D.COBJ +
        and H.CDOD = D.CDOD +
join CENYD as C on D.CDOD = C.CDOD +
        and D.CZBOZID = C.CZBOZID +
join DODAV as DOD on H.CDOD = DOD.CDOD +
where (CENAJ * MNOBJ) between 0.00 and 99999.00 +
        and DTOBJ <= ''2022-12-31'' +
order by COBJ asc +
For read only' ;

Exec SQL prepare SEL from :SQL_STATEMENT ;
Exec SQL describe SEL into :SQLDA;
Exec SQL declare CUR cursor for SEL;
Exec SQL open CUR ;

Exsr Adresy_SQLDA; // dosadit adresy proměnných do SQLDA

// přečíst a zpracovat data z dotazu
Exec SQL fetch next from CUR using descriptor :SQLDA ;
DoW SQLSTATE < '02000';
    Exsr ZpracData; // zpracovat data z proměnných
    Exec SQL fetch next from CUR using descriptor :SQLDA ;
EndDo;

*inlr = *on;

//-----
// Adresy_SQLDA - dosazení adres proměnných do SQLDA
//-----
BegSr Adresy_SQLDA;
    // dosadím adresy budoucích hodnot do SQLDA
    Ptr = %addr(DataSpace); // ukazatel na přijímaná data
    For Idx = 1 to SQLDA.SQLD; // skutečný počet sloupců
        SQLDA.SQL_VAR(Idx).SQLDATA = Ptr ; // ukazatel na položku dat
        Ptr += %size(Hodnota); // stejná délka pro všechny typy dat
    EndFor;
endsr; // Adresy_SQLDA

//-----
// ZpracData
//-----
BegSr ZpracData;
    Ptr = %addr(DataSpace);
    For Idx = 1 to SQLDA.SQLD; // skutečný počet sloupců
        JmenoSloupce = SQLDA.SQL_VAR(Idx).SQLNAME;
        // typ CHAR nebo DATE se přesune přímo
        If SQLDA.SQL_VAR(Idx).SQLTYPE = 452 or
            SQLDA.SQL_VAR(Idx).SQLTYPE = 453 or
            SQLDA.SQL_VAR(Idx).SQLTYPE = 384;
            Hodnota = Data;
            // typ DECIMAL je třeba převést do znakové podoby
        ElseIf SQLDA.SQL_VAR(Idx).SQLTYPE = 484 or
            SQLDA.SQL_VAR(Idx).SQLTYPE = 485;
            Digits = %div(SQLDA.SQL_VAR(Idx).SQLLEN : 256);
            DecPos = %rem(SQLDA.SQL_VAR(Idx).SQLLEN : 256);
            Size = %div(Digits: 2) + 1; // počet bajtů pakovaného čísla
            Packed = 0; // inicializovat pomocnou proměnnou
            PackedChar = %replace( %subst(Data: 1: Size): PackedChar:
                %size(PackedChar) - Size + 1: %size(PackedChar));
            Eval(H) Packed = Packed * 10 ** (%decpos(Packed) - Decpos);
            Hodnota = %editc(Packed: 'P');
        EndIf;
    EndFor;

```

```

        Except DETAIL;
        Ptr += %size(Hodnota);
    EndFor;
    Except ODDELOVAC;

    endsr; // ZpracData

    QQPRINT      E          DETAIL          1
    O              JmenoSloupce
    O              Hodnota              +2
    QQPRINT      E          ODDELOVAC       1
    O              '-----'

```

Výsledky jsou znázorněny v následujícím výpisu se jmény sloupců vlevo a jejich hodnotami vpravo.

```

*...+....1....+....2....+....3....+....4....+....5....+
COBJ      000001
CDOD      000006
DTOBJ     2020-09-20
CZBOZID   01062
MNOBJ                                4,000000000
CENAJ                                557,000000000
NAZZBO     Koloběžka střední bez brzdy
NAZDOD     Hračky v.o.s.
ADRDO      Ostrava
CENA_CELKEM                                2228,000000000
-----
COBJ      000001
CDOD      000006
DTOBJ     2020-09-20
CZBOZID   01063
MNOBJ                                12,000000000
CENAJ                                357,000000000
NAZZBO     Koloběžka malá bez brzdy
NAZDOD     Hračky v.o.s.
ADRDO      Ostrava
CENA_CELKEM                                4284,000000000
-----
COBJ      000002
CDOD      000003
DTOBJ     2014-02-27
CZBOZID   00023
MNOBJ                                40,000000000
CENAJ                                598,000000000
NAZZBO     Mléko pasteurisované tučné
NAZDOD     Laktos mlékárna
ADRDO      Praha
CENA_CELKEM                                23920,000000000
-----
COBJ      000002
CDOD      000003
DTOBJ     2014-02-27
CZBOZID   00239
MNOBJ                                30,000000000
CENAJ                                569,000000000
NAZZBO     Jogurt bílý plnotučný
NAZDOD     Laktos mlékárna
ADRDO      Praha
CENA_CELKEM                                17070,000000000
-----
...

```

## Ostatní dynamické příkazy

Dynamické příkazy jiné než SELECT, které nepoužívají značky (otazníky, markery) na místě proměnných, lze provádět příkazem EXECUTE IMMEDIATE, kdežto příkazy se značkami je nutné nejprve připravit příkazem PREPARE a pak provést příkazem EXECUTE.

### EXECUTE IMMEDIATE

První ukázka představuje dynamický příkaz UPDATE, jehož text je složen teprve při výpočtu připojením hodnoty proměnné `Limit` (číslo 500.00 ve znakové podobě). Příkaz zvýší cenu v určených řádcích o 10 %.

```
//*****  
//   Program DYNEX - Dynamický příkaz UPDATE s EXECUTE IMMEDIATE  
//   Kompilace volbou 14  
//   Volá se bez parametrů  
//*****  
  
Dcl-S DYNST                      Char(500) INZ;  
Dcl-S Limit                     Packed(9:2) inz(500.00);  
  
Exec SQL set option COMMIT = *NONE;  
  
DYNST = 'update CENYD_T set CENAJ = CENAJ * 1.10 +  
        where CENAJ < '  
DYNST = %trim(DYNST) + ' ' + %char(Limit) ;  
  
Exec SQL execute immediate :DYNST;  
  
*inlr = *on;
```

### PREPARE a EXECUTE se značkou

Druhá ukázka představuje podobný příkaz, který je sice zapsaný celý, ale obsahuje otazník na místě limitní ceny. Hodnota limitní ceny se dosadí teprve v příkazu EXECUTE z proměnné. Příkaz sníží cenu ve stejně určených řádcích nazpět do původní hodnoty.

```
//*****  
//   Program DYNEX_MK - Dynamický příkaz s PREPARE a EXECUTE  
//   Kompilace volbou 14  
//   Volání příkazem (*CMD) DYNEX_MK  
//*****  
  
Dcl-S DYNST                      Char(500) Inz;  
  
Dcl-PI *n;  
    Limit Packed(9:2); // parametr z CMD příkazu  
End-PI;  
  
Exec SQL set option COMMIT = *NONE;  
  
DYNST = 'update CENYD_T set CENAJ = CENAJ / 1.10 +  
        where CENAJ < ? '  
Exec SQL prepare STMT from :DYNST;  
Exec SQL execute STMT using :Limit;  
  
*inlr = *on;
```

## Testování výsledku SQL příkazů

Výsledkem přípravy i provedení příkazu je také kód výsledku dosazený do proměnné SQLSTATE (v souladu se standardem ISO) nebo číselné proměnné SQLCODE (v souladu s produkty DB2).

Jednu z těchto proměnných testujeme, abychom zjistili formální správnost příkazu – po příkazu PREPARE, OPEN, EXECUTE IMMEDIATE – nebo stav výpočtu, např. nenalezení řádku výsledné tabulky u příkazu FETCH.

SQLSTATE je pětimístný kód, jehož první dva znaky označují třídu (class):

```
00  Unqualified Successful Completion
01  Warning
02  No Data
07  Dynamic SQL Error
08  Connection Exception
09  Triggered Action Exception
...
42  Syntax Error or Access Rule Violation
...
```

SQLCODE je v relaci s kódem SQLSTATE a představuje kladné nebo záporné číslo. Kladná čísla jsou spíše upozornění nebo oznámení, záporná čísla představují chybu. Dokumentace IBM je na stránce <https://www.ibm.com/docs/en/i/7.4?topic=reference-sql-messages-codes>. Obě proměnné jsou také součástí datové struktury SQLCA – **SQL Communication Area**.

Ukázky kódů jsou uvedeny v následující tabulce.

SQLSTATE		SQLCODE
00000	Execution of the operation was successful and did not result in any type of warning or exception condition.	+000
01503	The number of result columns is larger than the number of variables provided.	+000, +030
02000	One of the following exceptions occurred: <ul style="list-style-type: none"><li>- The result of the SELECT INTO statement or the subselect of the INSERT statement was an empty table.</li><li>- The number of rows identified in the searched UPDATE or DELETE statement was zero.</li><li>- The position of the cursor referenced in the FETCH statement was after the last row of the result table.</li><li>- The fetch orientation is invalid.</li></ul>	+100
07001	The number of variables is not correct for the number of parameter markers.	-313
09000	A triggered SQL statement failed.	-723
42703	An undefined column or parameter name was detected.	-205, -206, -213, -5001

Často stačí zjišťovat, zda SQLSTATE je roven '00000' nebo menší než '02000'. Podobně SQLCODE, zda je roven 0 nebo menší než 100. Ve výpisu paměti lze oba výsledné kódy nalézt v rámci struktury SQLCA. Lze tam ovšem nalézt i jiné informace o provedeném příkazu.

Příklad: program TEST\_CH, data z knihovny VZSQL

```
// Ceny objednávaného zboží
Dcl-DS CENYD ext;
End-DS;

Exec SQL set option COMMIT = *NONE;

Exec SQL declare CSR cursor for
      select CZBOZID, CENAJ, NAZZBO from CENYD_T
      where NAZZBO >= 'P' and CENA_CELKEM > 100
      order by NAZZBO asc;

Exec SQL open CSR;
      dump(a) 'Test Dump'; // testovací výpis po OPEN

      except NADPIS;
Exec SQL fetch from CSR into :CZBOZID, :CENAJ, :NAZZBO;

DoW sqlstate < '02000'; // do počtu dat
      Exec SQL fetch from CSR into :CZBOZID, :CENAJ, :NAZZBO;
      except DETAIL;
EndDo;

Exec SQL close CSR;
*InLR = *On;

OQPRINT      E              NADPIS              2
O
O
O
...
```

Po příkazu OPEN nastane chyba SQLSTATE '42703', tj. SQLCODE -206, a v proměnné SQLERRMC je text **CENA\_CELKEM** v diagnostické zprávě v protokolu úlohy:

```
Message ID . . . . . : SQL0206
Date sent . . . . . : 04/01/22      Time sent . . . . . : 14:50:30

Message . . . . . : Column or global variable CENA_CELKEM not found.

Cause . . . . . : CENA_CELKEM was not found as a column of table *N in *N
and was not found as a global variable in *N. If the table is *N,
CENA_CELKEM is not a column of any table or view that can be referenced, or
CENA_CELKEM is a special register that cannot be set in an atomic compound
statement.
```

Část výpisu paměti z příkazu DUMP:

```
...

SQLCA          DS
SQLABC         BIN(9,0)      000000136.      '00000088'X
SQLAID        CHAR(8)       'SQLCA      '      'E2D8D3C3C1404040'X
```

SQLCABC	INT(10)	136	'00000088'X
SQLCAID	CHAR(8)	'SQLCA '	'E2D8D3C3C1404040'X
SQLCOD	BIN(9,0)	-000000206.	'FFFFFF32'X
SQLCODE	INT(10)	-206	'FFFFFF32'X
SQLERL	BIN(4,0)	0021.	'0015'X
SQLERM	CHAR(70)	' CENA_CELKEM	*N *N

,

VALUE IN HEX

'000BC3C5D5C16DC3C5D3D2C5D400025CD500025CD5000000000000000000000000000000000'X

41

'00'X

SQLERP	CHAR(8)	'QSQRCHK '	'D8E2D8D9C3C8D240'X
SQLERR	CHAR(24)	,	,

VALUE IN HEX

'000'X

SQLERRD	INT(10)	DIM(6)	
	(1-6)	0	'00000000'X

**SQLERRMC** CHAR(70) ' CENA\_CELKEM \*N \*N

,

VALUE IN HEX

'000BC3C5D5C16DC3C5D3D2C5D400025CD500025CD5000000000000000000000000000000000'X

41

'00'X

SQLERRML	INT(5)	21	'0015'X
SQLERRP	CHAR(8)	'QSQRCHK '	'D8E2D8D9C3C8D240'X
SQLER1	BIN(9,0)	000000000.	'00000000'X
SQLER2	BIN(9,0)	000000000.	'00000000'X
SQLER3	BIN(9,0)	000000000.	'00000000'X
SQLER4	BIN(9,0)	000000000.	'00000000'X
SQLER5	BIN(9,0)	000000000.	'00000000'X
SQLER6	BIN(9,0)	000000000.	'00000000'X
<b>SQLSTATE</b>	CHAR(5)	'42703'	'F4F2F7F0F3'X
SQLSTT	CHAR(5)	'42703'	'F4F2F7F0F3'X

...



# Srovnání tradičního přístupu s přístupem SQL v jazyku RPG

Na příkladu dvou RPG programů pro pořizování a údržbu objednávek porovnáme tradiční přístup s přístupem SQL k práci s databází.

Programy jsou v knihovně VZSQLPGM a data jsou v knihovně VZSQL.

Oba programy řeší tutéž úlohu - pořízení a údržbu objednávek zboží od dodavatelů.

První program - **OBJ\_RPG** - pracuje s tradičními fyzickými a logickými soubory a příkazy SETTLL, SETGT, READ, READE, CHAIN, UPDATE, WRITE, DELETE.

Druhý program - **OBJ\_SQLF** - pracuje jen s tabulkami/pohledy SQL a příkazy SELECT, OPEN, FETCH, UPDATE, INSERT, CLOSE.

Opisy celých programů jsou uvedeny [na konci publikace](#).

## Popisy souborů – host variables

```
// Objednávky - hlavička                klíč: číslo objednávky
Dcl-F OBJHLA_IX                        Usage(*UPDATE:*DELETE:*OUTPUT) Keyed;
//-----
// Objednávky - detail (zboží)          klíč: číslo objednávky
//                                     číslo zboží
Dcl-F OBJDET_IX                        Usage(*UPDATE:*DELETE:*OUTPUT) Keyed;
//-----
// Ceny objednávaného zboží            klíč: číslo dodavatele
//                                     číslo zboží
Dcl-F CENYD_IX                        Keyed;
// Ceny objednávaného zboží            klíč: název zboží
Dcl-F CENYDN_IX                        Keyed
                                     Rename(CENYD_T: CENYD_T2);
//-----
// Dodavatelé                          klíč: číslo dodavatele
Dcl-F DODAV_IX                        Keyed;
// Dodavatelé                          klíč: název dodavatele
Dcl-F DODAVN_IX                        Keyed
                                     Rename(DODAV_T: DODAV_T2);
```

## Datové struktury podle SQL tabulek a pohledů – host variables

```
// Objednávky - hlavička - přejmenování proti pohledu DODAV
//                                     inicializace je nutná kvůli datumu
Dcl-DS OBJHLA                        extname('OBJHLA_T') INZ;
      CDODHLA                        EXTFLD ('CDOD');
End-DS;

// Objednávky - detail (zboží)
// - přejmenování proti pohledům DODAV, OBJHLA, CENYD
Dcl-DS OBJDET                        extname('OBJDET_T');
      CDODDET                        extfld ('CDOD');
      COBJDET                        extfld ('COBJ');
      CZBOZIDET                     extfld ('CZBOZID');
End-DS;

// Ceny objednávaného zboží - přejmenování proti pohledu DODAV
Dcl-DS CENYD                        extname('CENYD');
      CDODCEN                        extfld ('CDOD');
End-DS;

// Dodavatelé
Dcl-DS DODAV                        extname('DODAV');
End-DS;
```

## ***Příkazy pro kompilátor - příkaz CTL-OPT***

```
Ctl-Opt LangID('CSY') SrtSeq(*LANGIDSHR) DatFmt(*ISO);
```

## ***Příkazy pro předkompilátor - SQL příkaz SET OPTION***

```
// Řazení znaků podle českých pravidel (pro názvy zboží a dodavatelů)
exec SQL set option LANGID = CSY, SRTSEQ = *LANGIDSHR,
// Určit typ pro datum: ISO
           DATFMT = *ISO,
// Zrušit sledování transakcí (aby byl umožněn UPDATE bez žurnálování)
           COMMIT = *NONE ;
```

Poznámka 1: Předkompilátor CRTSQLRPGI nebere v úvahu údaje z příkazu CTL-OPT, místo nich je k dispozici SQL příkaz SET OPTION. Příkaz SET OPTION, je-li použit, musí být zapsán před všemi dalšími SQL příkazy v programu, protože řídí proces jejich předkompilace.

Poznámka 2: Sledování transakcí je standardně nastaveno na \*CHG. Pro účely ladění je vhodné je výslovně vypnout parametrem COMMIT(\*NONE).

## ***Plnění podsouboru hlaviček objednávek se jmény dodavatelů***

```
// Přečíst první záznam z databáze hlaviček objednávek
SetLL *LoVal OBJHLA_IX;
Read(N) OBJHLA_IX; // Číst první záznam hlaviček objednávek
DoW Not %EOF And IO < MAXIMUM_SF;
    Chain CDOD DODAV_IX; // Číst dodavatele podle čísla z obrazovky
    IO += 1; // Zvýšit čítač záznamů podsouboru
    Write OBJWS0; // Zapsat záznam do podsouboru podle čítače
    Read(N) OBJHLA_IX; // Číst další záznam hlaviček objednávek
EndDo;
```

```
Exec SQL declare CS0 cursor for
    select COBJ, H.CDOD, DTOBJ, NAZDOD
    from OBJHLA as H
    join DODAV as D on H.CDOD = D.CDOD
    order by COBJ asc
    fetch first 9998 rows only; // MAXIMUM_SF je 9998
    for read only;
Exec SQL open CS0;
Exec SQL fetch from CS0 into :COBJ, :CDOD, :DTOBJ, :NAZDOD;
DoW sqlstate < '02000' And IO < MAXIMUM_SF;
    IO += 1; // Zvýšit čítač záznamů podsouboru
    Write OBJWS0; // Zapsat záznam do podsouboru podle čítače
    Exec SQL fetch from CS0 into :COBJ, :CDOD, :DTOBJ, :NAZDOD;
EndDo;
Exec SQL close CS0;
```

## ***Plnění podsouboru pro předchozí stránku zboží (po stisku Page Up)***

```
Chain(E) 1 OBJWS2; // Najít první dosavadní název zboží v podsouboru
... vymazat podsoubor
// Nastavit ukazatel do databáze ZA nalezený název (nebo na konec)
SetGT NAZZBO CENYDN_IX;
// Přechází Pag2 (nebo méně) databázových vět pozpátku
ReadP CENYDN_IX;
I2 = 0;
DoW Not %Eof And I2 < Pag2;
    I2 += 1;
    ReadP CENYDN_IX;
EndDo;

// Vynulovat čítač záznamů podsouboru 2
I2 = 0;
// Nastavit ukazatel do databáze PŘED naposledy přečtený název
SetLL NAZZBO CENYDN_IX;
// Číst nastavený záznam z databáze (dopředu)
Read CENYDN_IX;
// Cyklus, dokud není konec databáze nebo není vyčerpán
// počet řádků obrazkové stránky (Pag2)
DoW Not %Eof And I2 < Pag2;
    I2 += 1; // Zvýšit čítač záznamů podsouboru
    Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
    Read CENYDN_IX; // Číst další záznam z databáze zboží
EndDo;
```

```
Chain(E) 1 OBJWS2; // Najít první dosavadní název zboží v podsouboru
... vymazat podsoubor
// Přechází nejvýše 7 záznamů menších či rovných naposledy přečtenému názvu
exec SQL declare CS2U scroll cursor for
    select CZBOZI, CENAJ, NAZZBO from CENY
    where NAZZBO <= :NAZZBO // menší nebo rovno
    order by NAZZBO desc // seřadit sestupně pro čtení "pozpátku"
    fetch first 7 rows only; // Pag2 = 7

// Vynulovat čítač záznamů podsouboru 2
I2 = 0;
Exec SQL open CS2U;
// Číst poslední záznam z výsledné tabulky
Exec SQL fetch last from CS2U into :CZBOZI, :CENAJ, :NAZZBO;
DoW sqlstate < '02000' And I2 < Pag2;
    I2 += 1; // Zvýšit čítač záznamů podsouboru
    Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
    // Číst předchozí záznam výsledné tabulky
    exec SQL fetch prior from CS2U into :CZBOZI, :CENAJ, :NAZZBO;
EndDo;
Exec SQL close CS2U;
```

## ***Výmaz hlavičky a všech odpovídajících detailů objednávky***

```
Delete COBJ OBJHLA_IX; // Zruším hlavičku, na niž ukazuje kurzor

// Zruším odpovídající detaily, jestliže nějaké existují
// Najdu první záznam v databázi OBJDET_T s daným číslem objednávky
SetLL COBJ OBJDET_IX;
// Existuje-li, zruším všechny detaily s tímto číslem objednávky
If %Equal;
    ReadE COBJ OBJDET_IX; // Čtu první záznam se stejným číslem objednávky
    DoW Not %EOF; // Cyklus, dokud není konec skupiny záznamů se stejným
        // číslem objednávky
        Delete OBJDET_T; // Zruším přečtený záznam
        ReadE COBJ OBJDET_IX; // Čtu další záznam se stejným číslem obj.
    EndDo;
EndIf; // konec %equal
```

```
// Zruším hlavičku, na niž ukazuje kurzor
Exec SQL delete from OBJHLA where COBJ = :COBJ;

// Zruším odpovídající detaily, jestliže nějaké existují
Exec SQL delete from OBJDET where COBJ = :COBJ;
```

## ***Oprava detailních položek objednávky***

```
ReadC OBJWS1;  
DoW Not %EOF;  
  Exsr Uloz1;  
  Chain (COBJ: CZBOZID) OBJDET_IX;  
  If %Found(OBJDET_IX);  
    Exsr Obnov1;  
    Update OBJDET_T;  
  EndIf;  
  ReadC OBJWS1;  
EndDo;
```

```
ReadC OBJWS1;  
DoW Not %EOF;  
  exec SQL update OBJDET set MNOBJ = :MNOBJ  
    where COBJ = :COBJ and CZBOZID = :CZBOZID;  
  ReadC OBJWS1;  
EndDo;
```

## ***Zápis nové hlavičky objednávky***

```
// Zapsat novou hlavičku, není-li již v databázi
SetLL COBJ OBJHLA_IX;
If %Equal;
    Eval *In85 = *On; // Už tam je - chyba
    Iter; // Opakovat
Else;
    Eval *In85 = *Off;
EndIf;
Write OBJHLA_T; // Zapsat
```

```
// Zkusit, zda záznam existuje
Exec SQL select COBJ into :COBJ from OBJHLA
    where COBJ = :COBJ;

If sqlstate = '00000'; // když se našel
    *In85 = *on; // indikace chyby
    Iter; // Opakovat
Endif;
*In85 = *off;

// Zapsat nový záznam do hlavičky
Exec SQL insert into OBJHLA ( COBJ, CDOD, DTOBJ )
    values ( :COBJ, :CDOD, :DTOBJ );
```



## Rutiny SQL

Rutiny SQL jsou interní nebo externí programy se zvláštními pravidly pro předávání parametrů.

- *Uložená procedura* (stored procedure) je program (\*PGM). Volá se SQL příkazem CALL z aplikačního programu nebo interaktivně (STRSQL nebo Query Manager).
- *Uživatelská funkce* (user defined function UDF) je podprocedura v servisním programu (\*SRVPGM). Volá se jménem v SQL příkazu a vrací jednoduchou hodnotu, souhrnnou hodnotu nebo tabulku.
- *Spouštěč* (trigger) je program (\*PGM). Volá se automaticky v databázovém řídicím systému.

Rozlišujeme interní a externí rutiny. *Interní* rutiny se programují a registrují pomocí SQL příkazů.

*Externí* rutiny se programují v programovacím jazyku, v našem případě RPG, a registrují se pomocí SQL příkazů.

Uložené procedury nebo uživatelské funkce se ukládají do zvolené knihovny a pak se vyvolávají v aplikačních programech, v našem případě v RPG programech. Spouštěče se vztahují ke konkrétní tabulce, ukládají se do zvolené knihovny a vyvolávají se automaticky v SQL příkazech INSERT, DELETE, UPDATE.

# Interní rutiny SQL

Interní rutiny se vytvářejí SQL příkazy, nejlépe sestavenými do skriptu. Skript je textový soubor s koncovkou `.sql` obsahující SQL příkazy ukončené středníkem. Může se spouštět pomocí ACS volby *Spustit skripty SQL*.

Skript může být také zapsán v datovém členu zdrojového databázového souboru, např. `QSQLSRC`, tentokrát bez středníků na konci příkazů. Spouští se CL příkazem `RUNSQLSTM`.

## SQL příkazy pro rutiny

<code>label:</code>	návěští u příkazu
<code>BEGIN .. END</code>	složený příkaz
<code>SET proměnná = výraz</code>	přiřazovací příkaz
<code>CALL procedura ( parametry )</code>	volání procedury
<code>IF podmínka .. ELSEIF .. ENDIF .. END IF</code>	podmiňovací příkaz
<code>CASE [výraz] WHEN podmínka .. WHEN podmínka ..     ELSE .. END CASE</code>	výběr alternativ
<code>FOR kurzor FOR select DO .. END FOR</code>	cyklus přes kurzor
<code>LOOP .. END LOOP</code>	neomezený cyklus
<code>WHILE podmínka DO .. END WHILE</code>	cyklus dokud ano
<code>REPEAT .. UNTIL podmínka END REPEAT</code>	cyklus dokud ne
<code>ITERATE label</code>	nové opakování cyklu s návěstím
<code>LEAVE label</code>	opuštění cyklu nebo bloku s návěstím
<code>RETURN výraz</code>	návrat z rutiny
<code>SIGNAL</code>	signalizuje zprávu
<code>RESIGNAL</code>	zaslání zprávy do vyšší úrovně
<code>GET DIAGNOSTICS</code>	získání informace o stavu výpočtu
<code>GOTO label</code>	skok na návěští

## Složený příkaz pro rutiny

Příkazy uvnitř složeného příkazu musí být ukončeny středníkem.

```
BEGIN
  DECLARE variable|condition|retcode
  DECLARE cursor|handler

  řídící SQL příkazy

  + další SQL příkazy:
  SELECT INTO, INSERT, UPDATE, DELETE,
  CREATE, ALTER, DROP aj.
END
```

## Uložená procedura SQL (stored procedure)

### Obecný tvar SQL procedury

```
CREATE PROCEDURE jméno procedury
  ( parametry - in | out | inout )

MODIFIES|READS SQL DATA

volby-pro-vytvoření

jediný příkaz - jednoduchý nebo složený
```

## Vytvoření procedury

Uložená procedura **CENA\_DODAVEK** má formu SQL skriptu (soubor s koncovkou .sql). Spočítá cenu objednaného zboží od dodavatelů v rozmezí počátečního a koncového data. Cena je spočítána jak s daní tak bez daně. Procedura navíc zjistí počet objednávek z tohoto období. Parametry procedury jsou dva vstupní (počáteční a koncové datum) a tři výstupní (cena s daní, cena bez daně a počet objednávek). Příkaz SET SCHEMA dovoluje vynechat kvalifikátor VZSQL v příkazech. Procedura bude uložena v knihovně VZSQLPGM jako program pod jménem **CENA\_00001**, které vznikne transformací tak, aby mělo nejvýše 10 znaků. Bez ohledu na to se procedura volá z RPG programu jménem CENA\_DODAVEK v SQL příkazu CALL (viz dále).

```
set schema VZSQL;
create or replace procedure VZSQLPGM/CENA_DODAVEK
( in DATUM1 date,                -- počáteční datum
  in DATUM2 date,                -- koncové datum
  out CASTKA_CELKEM decimal (11, 2), -- Součet částek včetně DPH
  out CASTKA_BEZ_DPH decimal (11, 2) , -- Součet částek bez DPH
  out POCET_OBJEDNAVEK integer      -- Počet objednávek v rozmezí datumů
)

language sql reads sql data

begin
declare SUMA_S_DPH decimal (11, 2) default 0.00;
declare SUMA_BEZ_DPH decimal (11, 2) default 0.00;
declare POCET_OBJ integer default 0;

-- Cyklus přes kurzor pro spojení tabulek hlaviček, detailů, cen a dph
-- pro zjištění ceny za dodávky
for CYKLUS1 as CUR1 cursor for
  select CENAJ, MNOBJ, PROC_DPH
    from OBJHLA as H
    join OBJDET as D on H.COBJ = D.COBJ
    join CENYD_T as C on D.CZBOZID = C.CZBOZID
                        and D.CDOD = C.CDOD
    join DPH_T as DPH on C.SAZBA_DPH = DPH.SAZBA_DPH
  where DTOBJ between DATUM1 and DATUM2
do
  set SUMA_S_DPH = SUMA_S_DPH +
    ( CENAJ * MNOBJ ) * ( 1.00 + PROC_DPH / 100.00 ) ;
  set SUMA_BEZ_DPH = SUMA_BEZ_DPH + ( CENAJ * MNOBJ ) ;
end for;

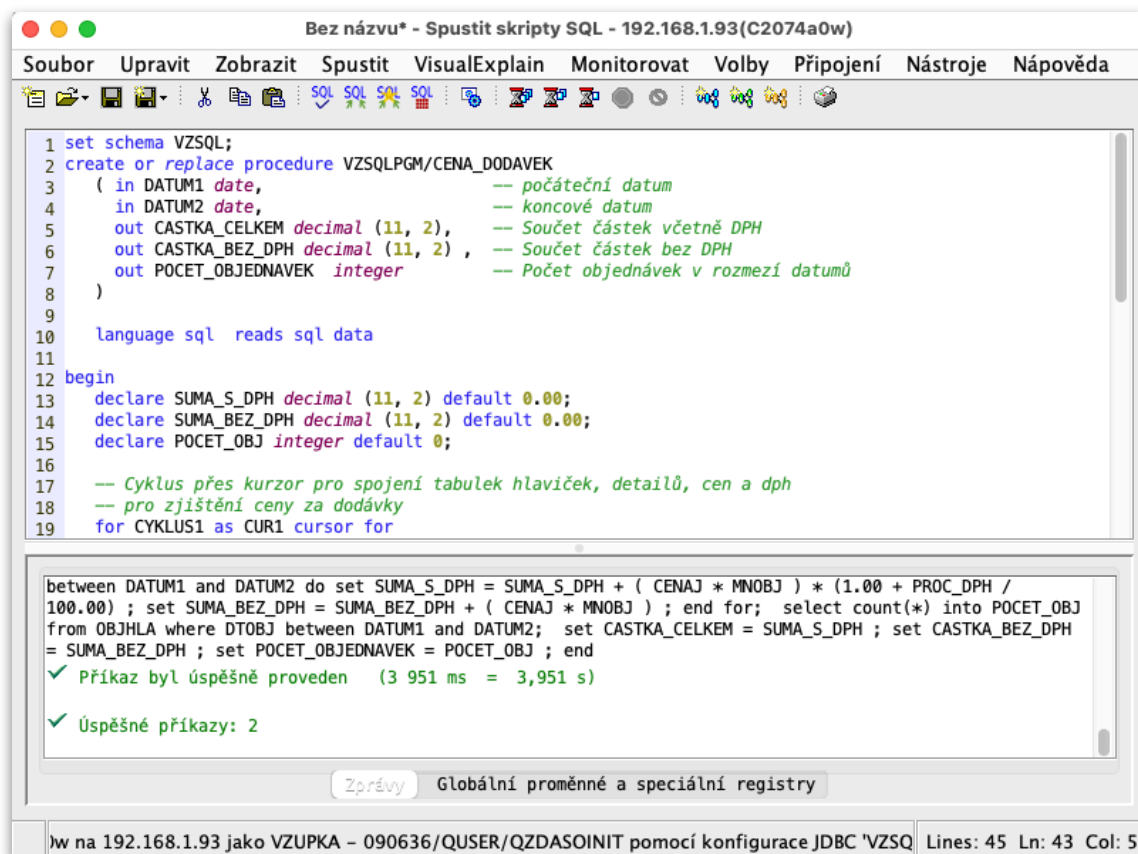
-- Zjištění počtu objednávek v daném rozmezí datumů
select count(*) into POCET_OBJ
  from OBJHLA
  where DTOBJ between DATUM1 and DATUM2;

-- Naplnění výstupních parametrů
set CASTKA_CELKEM = SUMA_S_DPH ;
set CASTKA_BEZ_DPH = SUMA_BEZ_DPH ;
set POCET_OBJEDNAVEK = POCET_OBJ ;

end;
```

SQL příkazem DROP PROCEDURE CENA\_DODAVEK můžeme proceduru odstranit.

Jeden způsob, jak definici procedury uložit, je spustit skript v ACS:



```
1 set schema VZSQL;
2 create or replace procedure VZSQLPGM/CENA_DODAVEK
3 ( in DATUM1 date,           -- počáteční datum
4   in DATUM2 date,           -- koncové datum
5   out CASTKA_CELKEM decimal (11, 2), -- Součet částek včetně DPH
6   out CASTKA_BEZ_DPH decimal (11, 2), -- Součet částek bez DPH
7   out POCET_OBJEDNAVEK integer      -- Počet objednávek v rozmezí datů
8 )
9
10 language sql reads sql data
11
12 begin
13   declare SUMA_S_DPH decimal (11, 2) default 0.00;
14   declare SUMA_BEZ_DPH decimal (11, 2) default 0.00;
15   declare POCET_OBJ integer default 0;
16
17   -- Cyklus přes kurzor pro spojení tabulek hlaviček, detailů, cen a dph
18   -- pro zjištění ceny za dodávky
19   for CYKLUS1 as CUR1 cursor for
20
21 between DATUM1 and DATUM2 do set SUMA_S_DPH = SUMA_S_DPH + ( CENAJ * MNOBJ ) * (1.00 + PROC_DPH /
22 100.00) ; set SUMA_BEZ_DPH = SUMA_BEZ_DPH + ( CENAJ * MNOBJ ) ; end for; select count(*) into POCET_OBJ
23 from OBJHLA where DTOBJ between DATUM1 and DATUM2; set CASTKA_CELKEM = SUMA_S_DPH ; set CASTKA_BEZ_DPH
24 = SUMA_BEZ_DPH ; set POCET_OBJEDNAVEK = POCET_OBJ ; end
25
26 ✓ Příkaz byl úspěšně proveden (3 951 ms = 3,951 s)
27
28 ✓ Úspěšné příkazy: 2
```

Zprávy Globální proměnné a speciální registry

hw na 192.168.1.93 jako VZUPKA - 090636/QUSER/QZDASOINIT pomocí konfigurace JDBC 'VZSQ Lines: 45 Ln: 43 Col: 5

Jiný způsob je použít CL příkaz RUNSQLSTM. Skript umístíme do zdrojového členu CENA\_DODAV v souboru QSQLSRC. Pak skript spustíme příkazem

**RUNSQLSTM SRCFILE(VZSQLPGM/QSQLSRC) SRCMBR(CENA\_DODAV)**

Uložená procedura CENA\_DODAVEK bude uložena v knihovně VZSQLPGM.

## Vyvolání procedury v RPG programu

Program **OBJ\_SUMF** se dvěma vstupními parametry (datum ve formátu ISO zadané znakově) volá uloženou proceduru **CENA\_DODAVEK** *SQL příkazem CALL*.

```
Dcl-F QPRINT    printer(120);

//   Parametry volání programu
Dcl-PI *n;
      DATUM1A          Char(10);
      DATUM2A          Char(10);
End-PI;

//   Parametry volání procedury (první dva jsou vstupní, ostatní výstupní)
Dcl-S Datum1          Date;
Dcl-S Datum2          Date;
Dcl-S suma_s_dph      Packed(11:2);
Dcl-S suma_bez_dph    Packed(11:2);
Dcl-S pocet_objedn     Int(10:0);

Exec SQL SET OPTION DATFMT = *ISO, COMMIT = *NONE;

Test(DE) *ISO DATUM1A;
If %Error;
    Datum1 = *Loval;
Else;
    Datum1 = %date(DATUM1A);
EndIf;
Test(DE) *ISO DATUM2A;
If %Error;
    Datum2 = *Hival;
Else;
    Datum2 = %date(DATUM2A);
EndIf;
Dump(a);
// volám uoženou proceduru CENA_DODAVEK
Exec SQL call CENA_DODAVEK ( :datum1, :datum2, :suma_s_dph,
                             :suma_bez_dph, :pocet_objedn );

Except DETAIL;

*inlr = *on;

OQPRINT      E          DETAIL          1
O
O
O          datum1
O
O          datum2
OQPRINT      E          DETAIL          1
O
O          ' S DPH '
O          suma_s_DPH    P
OQPRINT      E          DETAIL          1
O
O          'Bez DPH '
O          suma_bez_DPH  P
OQPRINT      E          DETAIL          1
O
O          'Počet objednávek '
O          pocet_objedn  P
```

Program se volá přes *CL příkaz* **OBJ\_SUMF** se dvěma parametry.

```
CMD      PROMPT('Volání SQL procedury OBJ_SUMF')
PARM     KWD(DATUM1) TYPE(*CHAR) LEN(10) +
          DFT('2013-01-01') +
          PROMPT('Datum OD')
PARM     KWD(DATUM2) TYPE(*CHAR) LEN(10) +
          DFT('2022-12-31') +
          PROMPT('Datum OD')
```

### Výsledek výpočtu

```
Celková cena objednaného zboží od 2013-01-01 do 2022-12-31
  S DPH      484546.80
Bez DPH      458061.00
Počet objednávek          6
```

## Uživatelská funkce SQL (UDF)

### Obecný tvar uživatelské funkce SQL

Uživatelská funkce (user defined function UDF) má jen vstupní parametry a vrací hodnotu. *Volá se jménem* v SQL příkazu a *vrací* buď jednoduchou či souhrnnou *hodnotu* nebo vrací *tabulku* (UDTF - user defined table function).

```
CREATE FUNCTION jméno funkce
  ( vstupní parametry )
  RETURNS výraz | TABLE ( definice-sloupců )
  MODIFIES|READS SQL DATA
  SET OPTION volby-pro-vytvoření
```

*jediný příkaz - jednoduchý nebo složený*

### Vytvoření tabulkové funkce

Funkce **OBJDAT\_F** přijímá dva parametry určující rozmezí datumů. Vybírá objednávky z daného rozmezí a *vrací tabulku* s čísly objednávek a s daty. Seznam bude uspořádán podle čísla objednávky vzestupně.

```
set schema VZSQL;
CREATE OR REPLACE FUNCTION VZSQLPGM/OBJDAT_F (DATUM1 DATE, DATUM2 DATE)
  RETURNS TABLE
  ( COBJ CHAR(6) CCSID 870,
    DTOBJ DATE )
  LANGUAGE SQL
  BEGIN
    RETURN SELECT COBJ, DTOBJ FROM OBJHLA_T
           WHERE DTOBJ BETWEEN DATUM1 AND DATUM2
           ORDER BY COBJ;
  END
```

SQL příkazem DROP odstraníme funkci.

```
DROP FUNCTION OBJDAT_F
```

Definici můžeme uložit spuštěním skriptu z okna *Spustit skripty SQL* v ACS nebo CL příkazem

```
RUNSQLSTM SRCFILE(VZSQLPGM/QSQLSRC) SRCMBR(OBJDAT_F)
```

Uživatelská funkce OBJDAT\_F bude vytvořena jako *servisní program* (typ \*SRVPGM) v knihovně VZSQLPGM.

## Volání tabulkové funkce v RPG programu

Program **OBJDAT\_P** zdrojového typu SQLRPGLE ukazuje použití funkce **OBJDAT\_F** v příkazu **SELECT**. Protože uživatelská funkce je vytvořena jako *servisní program* (typ **\*SRVPGM**), je nutné jej připojit k programu. Předpokládáme, že běžná knihovna (**\*CURLIB**) je **VZSQLPGM**. Zdrojový text **OBJDAT\_P** tedy zkompilujeme příkazem

```
CRTSQLRPGI OBJ(OBJDAT_P) SRCFILE(VZSQLPGM/QRPGLESRC) SRCMBR(OBJDAT_P)
              OBJTYPE(*MODULE)
```

do modulu **OBJDAT\_P** a k tomu pak připojíme servisní program **OBJDAT\_F** příkazem **CRTPGM**, čímž vznikne program **OBJDAT\_P**:

```
CRTPGM PGM(OBJDAT_P) MODULE(*PGM) BNDSRVPGM((OBJDAT_F))
```

V programu **OBJDAT\_P** se provádí výběr příkazem **SELECT** z tabulky, kterou vrací funkce **OBJDAT\_F**.

```
Dcl-F QPRINT    printer(120);

//   Parametry volání programu
Dcl-PI *n;
      DATUM1A          Char(10);
      DATUM2A          Char(10);
End-PI;

Dcl-S COBJ          Char(6);
Dcl-S DTOBJ          Date;

//   Pracovní data
Dcl-S Datum1          Date;
Dcl-S Datum2          Date;

Exec SQL SET OPTION DATFMT = *ISO, COMMIT = *NONE;

Test(DE) *ISO DATUM1A;
If %Error;
    Datum1 = *Loval;
Else;
    Datum1 = %date(DATUM1A);
EndIf;
Test(DE) *ISO DATUM2A;
If %Error;
    Datum2 = *Hival;
Else;
    Datum2 = %date(DATUM2A);
EndIf;

// Volám tabulkovou funkci OBJDAT_F.
// Vybírá čísla objednávek od data do data.
// - parametry musí být proměnné, nikoliv konstanty,
// - korelace AS xxx je povinná
Exec SQL declare CUR cursor for
      SELECT * FROM TABLE ( OBJDAT_F(:datum1 , :datum2 ) ) AS DAT_F;

Dsply sqlstate;
Exec SQL open CUR ;

// tisknu všechny záznamy vybrané tou funkcí
```

```

Exec SQL fetch CUR into :COBJ, :DTOBJ ;
Dsply sqlstate;
DoW sqlstate = '00000'; // dokud jsou nějaké záznamy
    Except DETAIL;
    Exec SQL fetch CUR into :COBJ, :DTOBJ ;
EndDo;
Exec SQL close CUR;
Dump(a) 'Koncový výpis';

*inlr = *on;

OQPRINT      E              DETAIL          1
O              COBJ
O              DTOBJ          +  1

```

Program OBJDAT\_P se volá přes *CL příkaz* OBJDAT\_P se dvěma parametry.

```

CMD          PROMPT('Volání SQL programu OBJDAT_P')
PARM         KWD(DATUM1) TYPE(*CHAR) LEN(10) +
              DFT('2013-01-01') +
              PROMPT('Datum OD')
PARM         KWD(DATUM2) TYPE(*CHAR) LEN(10) +
              DFT('2022-12-31') +
              PROMPT('Datum OD')

```

Program tiskne čísla objednávek v rozmezí zadaných datumů.

```

000001 2020-09-20
000002 2014-02-27
000003 2014-01-29
000005 2014-01-10
000006 2020-09-20
444444 2015-04-25

```



## Spouštěč SQL (trigger)

### Obečný tvar spouštěče

Spouštěč (trigger) se váže na určitou tabulku a vyvolává se automaticky, nastane-li určitá událost. Událost je určena příkazem INSERT, DELETE nebo UPDATE. Spouštěč se aktivuje (spustí) buď *před* nebo *po* provedení dotyčného příkazu.

```
CREATE TRIGGER jméno spouštěče
  BEFORE | AFTER | INSTEAD OF
  INSERT | DELETE | UPDATE
  ON table|view
  REFERENCING NEW|OLD AS zkratka
  FOR EACH STATEMENT|ROW MODE DB2SQL|DB2ROW

  SET OPTION volby-pro-vytvoření
  WHEN ( podmínka )
  jediný příkaz - jednoduchý nebo složený
```

### Příklady spouštěčů

Dva spouštěče pro tabulku detailů OBJDET\_T zajišťují, že objednané množství nepřekročí určitou hodnotu. Jeden je aktivován *před* operací INSERT, druhý *před* operací UPDATE.

Trigger BEFORE INSERT pro tabulku OBJDET\_T

```
CREATE OR REPLACE TRIGGER TRG_MNOBJI BEFORE INSERT ON OBJDET_T
  REFERENCING NEW ROW AS new_row -- odkazuje na nově vkládaný řádek
  FOR EACH ROW                    -- spustí se u každého řádku
  MODE DB2ROW
  BEGIN
    IF new_row.MNOBJ > 1000 THEN
      SET new_row.MNOBJ = 0;
    END IF;
  END
```

Pokusí-li se někdo vložit nový řádek s množstvím větším než 1000, spouštěč dosadí nulu.

Spouštěč můžeme vytvořit a zaregistrovat spuštěním skriptu z okna *Spustit skripty SQL* v ACS. Skript také můžeme spustit CL příkazem ze zdrojového členu.

```
RUNSQLSTM SRCFILE(VZSQLPGM/QSQLSRC) SRCMBR(TRG_MNOBJI)
```

Spouštěč bude uložen v knihovně VZSQL.

CL příkaz CHGPFTRG zneschopní nebo uschopní trigger.

```
CHGPFTRG FILE(VZSQL/OBJDET_T) TRG(TRG_MNOBJI) TRGLIB(VZSQL) STATE(*DISABLED)
CHGPFTRG FILE(VZSQL/OBJDET_T) TRG(TRG_MNOBJI) TRGLIB(VZSQL) STATE(*ENABLED)
```

SQL příkazem DROP odstraníme trigger.

```
DROP TRIGGER TRG_MNOBJI
```

Spouštěč můžeme demonstrovat na programu OBJ\_SQL z knihovny VZSQLPGM.

## Trigger BEFORE UPDATE pro tabulku OBJDET\_T

```
CREATE OR REPLACE TRIGGER TRG_MNOBJU BEFORE UPDATE ON OBJDET_T
  REFERENCING OLD ROW AS old_row
               NEW ROW AS new_row
  FOR EACH ROW
  MODE DB2ROW
  WHEN (new_row.MNOBJ > 100 and
        new_row.MNOBJ > old_row.MNOBJ * 1.10 )
  BEGIN
    SET new_row.MNOBJ = old_row.MNOBJ * 1.10;
    SIGNAL SQLSTATE VALUE '01H01' SET MESSAGE_TEXT =
      'Navýšení množství přesahuje 10 %. Ponechá se navýšení 10 %';
  END
```

Spouštěč zkoumá, zda nově vkládané množství nepřesahuje zvýšení o 10 procent proti starému. Přesahuje-li, zvýší se právě o 10 % proti starému. Spouštěč zároveň signalizuje zprávu, že je chybné množství.

Poznámka: V programu, který způsobil vyvolání spouštěče, však nebude v proměnné SQLSTATE kód '01H01', ale kód '00000'. Text zprávy se objeví jen v protokolu úlohy (job log).

Spouštěč můžeme vytvořit a zaregistrovat spuštěním skriptu z okna *Spustit skripty SQL* v ACS. Skript také můžeme spustit CL příkazem ze zdrojového členu.

```
RUNSQLSTM SRCFILE(VZSQLPGM/QSQLSRC) SRCMBR(TRG_MNOBJU)
```

CL příkaz CHGPFTRG zneschopní nebo uschopní trigger.

```
CHGPFTRG FILE(VZSQL/OBJDET_T) TRG(TRG_MNOBJU) TRGLIB(VZSQL) STATE(*DISABLED)
CHGPFTRG FILE(VZSQL/OBJDET_T) TRG(TRG_MNOBJU) TRGLIB(VZSQL) STATE(*ENABLED)
```

SQL příkazem DROP odstraníme trigger.

```
DROP TRIGGER TRG_MNOBJU
```

Spouštěč můžeme demonstrovat na programu OBJ\_SQLF z knihovny VZSQLPGM.

# Externí rutiny SQL

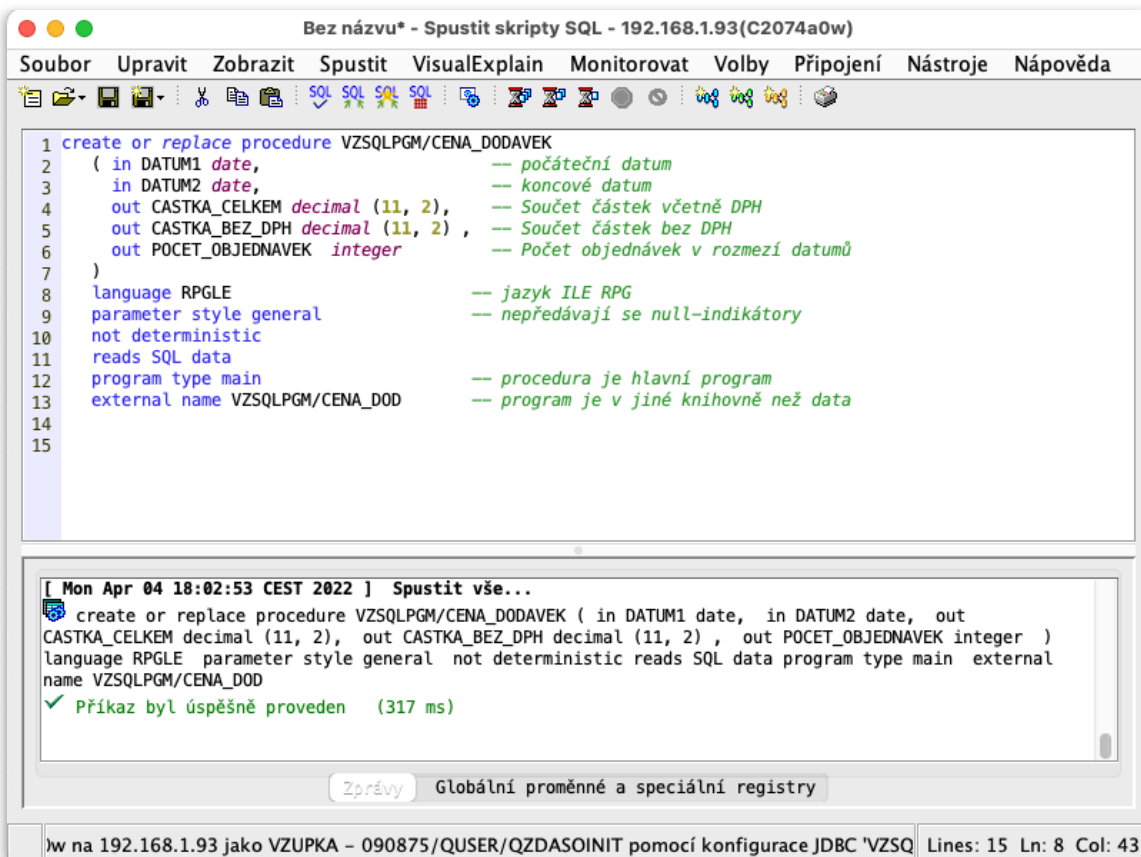
Uložené procedury a uživatelské funkce mohou být zapsány také jako programy nebo servisní programy a registrují se pomocí SQL příkazů pro rutiny. Vyvolávají se stejně jako ty zapsané v SQL.

## Uložená procedura zapsaná v RPG

Zde uvedená procedura CENA\_DODAVEK provádí stejnou činnost jako její ekvivalent v SQL (viz [výše](#)). Máme-li již k dispozici RPG program CENA\_DOD, který realizuje tuto činnost, můžeme proceduru registrovat. Registrace se zapisuje stejným příkazem jako u interní SQL procedury. Popis vstupních a výstupních parametrů je shodný, další údaje se liší.

```
create or replace procedure VZSQLPGM/CENA_DODAVEK
( in DATUM1 date,                -- počáteční datum
  in DATUM2 date,                -- koncové datum
  out CASTKA_CELKEM decimal (11, 2), -- Součet částek včetně DPH
  out CASTKA_BEZ_DPH decimal (11, 2) , -- Součet částek bez DPH
  out POCET_OBJEDNAVEK integer    -- Počet objednávek v rozmezí datumů
)
language RPGLE                -- jazyk ILE RPG
parameter style general         -- nepředávají se null-indikátory
not deterministic
reads SQL data
program type main              -- procedura je hlavní program
external name VZSQLPGM/CENA_DOD -- program je v jiné knihovně než data
```

Jeden způsob, jak definici uložit, je spustit skript v ACS:



Jiný způsob je spustit příkaz

```
RUNSQLSTM SRCFILE(VZSQLPGM/QSQLSRC) SRCMBR(CENA_DODA)
```

Výkonnou část procedury realizuje RPG program **CENA\_DOD** uložený v knihovně VZSQLPGM, která je určena pro *nedatové* objekty.

```
// Parametry procedury
Dcl-PI *n;
    Datum1          Date;
    Datum2          Date;
    Castka_s_DPH    Packed(11:2);
    Castka_bez_DPH  Packed(11:2);
    Poc_objednavek  Int(10:0);
End-PI;

// Pracovní proměnné
Dcl-S Suma_s_DPH    Packed(11:2);
Dcl-S Suma_bez_DPH  Packed(11:2);
Dcl-S Poc_obj       Int(10:0);

// Host variables
Dcl-S CENAJ          Like(Castka_s_DPH);
Dcl-S MNOBJ          Like(Castka_bez_DPH);
Dcl-S PROC_DPH       Like(Poc_objednavek);

Exec SQL set option DATFMT = *ISO, COMMIT = *NONE;

Exec SQL declare CUR cursor for
    select CENAJ, MNOBJ, PROC_DPH
    from OBJHLA as H
    join OBJDET as D on H.COBJ = D.COBJ
    join CENYD_T as C on D.CZBOZID = C.CZBOZID
    and D.CDOD = C.CDOD
    join DPH_T as DPH on C.SAZBA_DPH = DPH.SAZBA_DPH
    where DTOBJ between :Datum1 and :Datum2 ;
Exec SQL open CUR;

Exec SQL fetch from CUR into :CENAJ, :MNOBJ, :PROC_DPH;

DoW sqlstate = '00000'; // dokud jsou nějaké záznamy
    SUMA_S_DPH = SUMA_S_DPH + (CENAJ * MNOBJ) * (1.00 + PROC_DPH/100.00);
    SUMA_BEZ_DPH = SUMA_BEZ_DPH + ( CENAJ * MNOBJ ) ;
    Exec SQL fetch from CUR into :CENAJ, :MNOBJ, :PROC_DPH;
EndDo;
Exec SQL select count(*) into :POC_OBJ
    from OBJHLA
    where DTOBJ between :Datum1 and :Datum2;

// Naplnění výstupních parametrů
Castka_s_DPH = Suma_s_DPH ;
Castka_bez_DPH = Suma_bez_DPH ;
Poc_objednavek = Poc_obj ;

Return;
```

Příkazy SELECT jsou shodné s těmi, které byly zapsány v proceduře psané jazykem SQL. Datové objekty (pohledy a tabulky) se čerpají z knihovny VZSQL, která je zapsána v seznamu knihoven \*LIBL.

Externí uloženou proceduru CENA\_DODAVEK vyzkoušíme vyvoláním stejného *CL příkazu* OBJ\_SUMF, který volá *program* OBJ\_SUMF se dvěma vstupními parametry (datum ve formátu ISO zadané znakově). Ten volá uloženou proceduru CENA\_DODAVEK *SQL příkazem* CALL.

Celková cena objednaného zboží od 2013-01-01 do 2022-12-31

S DPH 484546.80

Bez DPH 458061.00

Počet objednávek 6

## ***Uživatelská tabulková funkce zapsaná v RPG***

Externě realizovaná funkce OBJDAT\_F provádí totéž co její SQL ekvivalent (viz [výše](#)). Je zde realizována v RPG namísto v SQL. Definiční část se zapisuje stejným příkazem jako u SQL funkce. Popis vstupních parametrů a vrácené tabulky je shodný, další údaje se liší.

```
CREATE OR REPLACE FUNCTION VZSQLPGM/OBJDAT_F (DATUM1 DATE, DATUM2 DATE)
  RETURNS TABLE
  ( COBJ CHAR(6) CCSID 870,
    DTOBJ DATE )
  LANGUAGE RPGLE
  PARAMETER STYLE DB2SQL          -- umožňuje open, fetch, close
  NOT DETERMINISTIC
  READS SQL DATA
  PROGRAM TYPE SUB                -- funkce je ILE podprocedura
  SCRATCHPAD                      -- paměť přetrvávající jednotlivá volání
  NOT FENCED                     -- není vázána na stejnou úlohu
  NO FINAL CALL                  -- není první a poslední volání
  CARDINALITY 1000               -- přibližné omezení počtu výsledných řádků
  EXTERNAL NAME VZSQLPGM/OBJDAT_F(OBJDAT_F) -- servisní program a procedura
```

Tato definice funkce se uloží do knihovny VZSQLPGM jen tehdy, existuje-li uvedený servisní program.

Předpokládáme, že běžná knihovna (\*CURLIB) je VZSQLPGM. K vytvoření servisního programu vytvoříme nejprve zdrojový soubor QSRVSRG a v něm člen OBJDAT\_F s typem BND (binder source):

```
STRPGMEXP SIGNATURE('VER1')
EXPORT SYMBOL(OBJDAT_F)
ENDPGMEXP
```

Potom vytvoříme servisní program kompilačním (a zároveň spojovacím) příkazem

```
CRTSQLRPGI OBJ(VZSQLPGM/OBJDAT_F) SRCFILE(VZSQLPGM/QRPGLESRC) SRCMBR(*OBJ)
  COMMIT(*NONE) OBJTYPE(*SRVPGM) TGTRLS(V7R3M0) CLOSQLCSR(*ENDACTGRP)
```

v knihovně VZSQLPGM určené pro nedatové objekty.

Alternativně bychom mohli zdroj samostatně kompilovat do modulu OBJTYPE(\*MODULE) a pak vytvořit servisní program příkazem CRTSRVPGM. Modul však nesmí mít stejné jméno jako exportovaná procedura, museli bychom jej pojmenovat jinak.

Jakmile máme vytvořený servisní program, můžeme spustit skript s definicí funkce buď pomocí ACS nebo příkazem

```
RUNSQLSTM SRCFILE(VZSQLPGM/QSQLSRC) SRCMBR(OBJDAT_FR)
```

Poznámka: Servisní program musíme připojit k nějakému hlavnímu modulu, zde opět OBJDAT\_P:

```
CRTPGM PGM(OBJDAT_P) MODULE(*PGM) BNDSRVPGM((OBJDAT_F))
```

Následuje výpis modulu OBJDAT\_F, který realizuje UDF funkci OBJDAT\_F. Ve skutečnosti nejde o funkci (ta by musela vracet hodnotu), ale o podproceduru s výstupními parametry.

### **Modul OBJDAT\_F**

Procedura OBJDAT\_F realizuje výkonnou část funkce. Servisní program OBJDAT\_F je vytvořený v ILE RPG jako *modul s jednou procedurou*. Procedura musí vytvořit výslednou tabulku a vrátit ji aplikačnímu programu. Databázový systém ji volá několikrát podle toho, kolik je záznamů ve výsledné tabulce po vyvolání typu Open.

```
Ctl-Opt nomain;  
/*****  
//   Procedura nevrací jednu hodnotu, "funkce" vrací hodnoty postupně  
//   prostřednictvím výstupních parametrů  
//   Procedura je volána vícekrát, vždy s určitým typem volání:  
//       - Open = -1  
//       - Fetch = 0  
//       - Close = 1  
/*****  
  
Dcl-Proc OBJDAT_F Export;  
  
//   Datové struktury pro databázové soubory (pohledy)  
Dcl-DS OBJHLA_T           Ext           Template ;  
    DTOBJDET              Extfld('DTOBJ');  
End-DS;  
Dcl-DS OBJDET_T           Ext           Template ;  
    COBJDET              Extfld('COBJ');  
    CDODDET              Extfld('CDOD');  
End-DS;  
  
//   Datová struktura dat přetrvávajících mezi voláními funkce  
Dcl-DS ScratchDS          Template;  
    ScrLen                Int(10:0) Inz(%Size(ScratchDS));  
    Cntr                  Packed(6:0) Inz(0);  
End-DS;  
  
//   Symbolické konstanty  
Dcl-C OPEN                -1;  
Dcl-C FETCH               0;  
Dcl-C CLOSE               1;  
  
//   Procedure interface  
Dcl-PI *N;  
    Datum1                Date;  
    Datum2                Date;  
  
    COBJ_PAR              Like(COBJDET);  
    DTOBJ_PAR             Like(DTOBJDET);  
  
    Datum1_ind            Int(5:0);  
    Datum2_ind            Int(5:0);  
    COBJ_PAR_ind          Int(5:0);  
    DTOBJ_PAR_ind         Int(5:0);  
  
    SQLSTATE_PAR          Char(5);
```

```

Func_name          VarChar(517);
Spec_name          VarChar(128);
Message_text       VarChar(1000);
Scratchpad         LikeDS(ScratchDS);
CallType           Int(10:0);
End-PI;

```

```

Exec SQL set option DATFMT = *ISO, COMMIT = *NONE;

```

```

If CallType = Open;

```

```

    Exec SQL declare CUR cursor for
        select COBJ, DTOBJ from OBJHLA_T
        where DTOBJ between :Datum1 and :Datum2
        order by COBJ;
    Exec SQL open CUR;

```

```

// Pro srovnání složitosti úryvek z SQL příkazu interní rutiny
//RETURN SELECT COBJ, DTOBJ FROM OBJHLA_T
//      WHERE DTOBJ BETWEEN DATUM1 AND DATUM2
//      ORDER BY COBJ;

```

```

    Scratchpad.Cntr = 0;      // inicializace čítače
    SQLSTATE_PAR = SQLSTATE;
    // dump(a) 'Calltype -1';

```

```

ElseIf CallType = Fetch;

```

```

    // Aby kurzor přetrval jednotlivá volání, je nutné při kompilaci
    // zadat parametr CLOSQLCSR(*ENDACTGRP)
    Exec SQL fetch next from CUR into
        :COBJ_PAR :COBJ_PAR_ind,
        :DTOBJ_PAR :DTOBJ_PAR_ind ;

    Scratchpad.Cntr += 1;
    SQLSTATE_PAR = SQLSTATE;

```

```

ElseIf CallType = Close;

```

```

    Exec SQL close CUR;
    SQLSTATE_PAR = SQLSTATE;

```

```

EndIf;

```

```

End-Proc OBJDAT_F;

```

Poznámka 1: Čítač *Scratchpad.Cntr* zde není použit, mohl by sloužit např. k omezení nebo tisku počtu vrácených řádků.

Poznámka 2: Velmi důležitý je parametr CLOSQLCSR s hodnotou \*ENDACTGRP zadaný při kompilaci modulu; zachovává otevřený kurzor mezi jednotlivými voláními procedury (Open, Fetch, Close). Kurzor se zavře až při ukončení aktivační skupiny. Předvolená hodnota \*ENDMOD by způsobila, že kurzor by se zavřel po každém volání procedury (nejen při volání Close).

Poznámka 3: Jednotlivá volání jsou realizována jako vlákna (threads). V nich nejsou dostupné hodnoty proměnných pro výpis paměti (dump). Ladění je ale možné pomocí programu STRDBG.

Program **OBJDAT\_P** se volá *CL příkazem* **OBJDAT\_P** se dvěma parametry.

```
CMD      PROMPT('Volání SQL programu OBJDAT_P')
PARM     KWD(DATUM1) TYPE(*CHAR) LEN(10) +
          DFT('2013-01-01') +
          PROMPT('Datum OD')
PARM     KWD(DATUM2) TYPE(*CHAR) LEN(10) +
          DFT('2022-12-31') +
          PROMPT('Datum OD')
```

V programu se funkce volá v příkazu **SELECT**, kde vrací tabulku do výrazu **TABLE(...)**:

```
Exec SQL declare CUR cursor for
      SELECT * FROM TABLE ( OBJDAT_F(:datum1 , :datum2 ) ) AS DAT_F;
```

Program tiskne čísla objednávek v rozmezí zadaných datumů.

```
000001 2020-09-20
000002 2014-02-27
000003 2014-01-29
000005 2014-01-10
000006 2020-09-20
444444 2015-04-25
```



## Spouštěč zapsaný v RPG

### Trigger BEFORE INSERT pro tabulku OBJDET\_T

Program TRG\_MNOBJI provádí stejnou činnost jako jeho SQL ekvivalent (viz [výše](#)).

```
**free

//  Vstupní parametry získané z databázového systému
Dcl-PI *n;
    Buf                likeDS(TrgBuffer);
    Len                like(TrgBufLen);
End-PI;

//  Trigger buffer - obsahuje data nového záznamu
//  (1. parametr)
Dcl-DS TrgBuffer                Len(1000) qualified;
    //  Statická oblast
    FileName                    Char(10);
    LibraryName                  Char(10);
    MemberName                    Char(10);
    TrgEvent                      Char(1);
    TrgTime                      Char(1);
    CmtLckLvl                    Char(1);
    //  Dynamická oblast
    NewRecOffset                Uns(10) pos(65);
    NewRecLen                    Uns(10) pos(69);
End-DS;

//  Délka trigger bufferu
//  (2. parametr)
Dcl-S TrgBufLen                Uns(10);

//  Nový záznam (before insertion)
Dcl-DS NewRecord    ExtName('OBJDET_T') Based(NewRecPtr) End-DS;

//  Pointer na Nový záznam
Dcl-S NewRecPtr                Pointer;

NewRecPtr = %Addr(Buf) + Buf.NewRecOffset;

If MNOBJ > 1000;
    MNOBJ = 0;
EndIf;
Return;
```

Program se kompiluje normálně - volbou 14 (CRTBNDRPG) do knihovny VZSQLPGM.

Spouštěč je vázán k souboru (tabulce OBJDET\_T) a registruje se do knihovny VZSQL CL příkazem

```
ADDPFTRG FILE(VZSQL/OBJDET_T) TRGTIME(*BEFORE) TRGEVENT(*INSERT)
          PGM(VZSQLPGM/TRG_MNOBJI) RPLTRG(*YES) TRG(TRG_MNOBJI) TRGLIB(*FILE)
          ALWREPCHG(*YES)
```

Odpojuje se CL příkazem

```
RMVPFTRG FILE(VZSQL/OBJDET_T) TRGTIME(*BEFORE) TRGEVENT(*INSERT)
```

nebo SQL příkazem

```
DROP TRIGGER TRG_MNOBJI
```

CL příkaz CHGPFTRG zneschopní nebo uschopní spouštěč (viz [výše](#)).

Spouštěč můžeme demonstrovat na programu OBJ\_SQLF z knihovny VZSQLPGM.

## **Trigger BEFORE UPDATE pro tabulku OBJDET\_T**

Program TRG\_MNOBJU provádí stejnou činnost jako jeho SQL ekvivalent (viz [výše](#)), až na signalizovanou zprávu.

```
**free

//   Trigger buffer - obsahuje data nového záznamu
//   (1. parametr)
Dcl-DS TrgBuffer          Len(1000);
//   Statická oblast
  FileName                Char(10);
  LibraryName              Char(10);
  MemberName               Char(10);
  TrgEvent                 Char(1);
  TrgTime                  Char(1);
  CmtLckLvl                Char(1);
//   Dynamická oblast
  OldRecOffset             Uns(10) pos(49);
  OldRecLen                 Uns(10);
  NewRecOffset              Uns(10) pos(65);
  NewRecLen                 Uns(10);
End-DS;

//   Délka trigger bufferu
//   (2. parametr)
Dcl-S   TrgBufLen          Uns(10);

//   Starý záznam (before update)
Dcl-DS OldRecord   ExtName('OBJDET_T') Based(OldRecPtr) Qualified End-DS;

//   Nový záznam (before insertion)
Dcl-DS NewRecord   ExtName('OBJDET_T') Based(NewRecPtr) Qualified End-DS;

//   Pointer na Starý záznam
Dcl-S OldRecPtr      Pointer;
//   Pointer na Nový záznam
Dcl-S NewRecPtr      Pointer;

//   Vstupní parametry získané z databázového systému
Dcl-PI *n;
  Buf                likeDS(TrgBuffer);
  Len                like(TrgBufLen);
End-PI;

OldRecPtr = %Addr(Buf) + Buf.OldRecOffset; // Ukazatel na Starý záznam
NewRecPtr = %Addr(Buf) + Buf.NewRecOffset; // Ukazatel na Nový záznam

If (NewRecord.MNOBJ > 100
    and NewRecord.MNOBJ > OldRecord.MNOBJ * 1.10) ;
  NewRecord.MNOBJ = OldRecord.MNOBJ * 1.10 ;
EndIf;
Return;
```

Program se kompiluje normálně - volbou 14 (CRTBNDRPG) do knihovny VZSQLPGM.

Spouštěč je vázán k souboru (tabulce OBJDET\_T) a registruje se do knihovny VZSQL CL příkazem

```
ADDPFTRG FILE(VZSQL/OBJDET_T) TRGTIME(*BEFORE) TRGEVENT(*UPDATE)
          PGM(VZSQLPGM/TRG_MNOBJU) RPLTRG(*YES) TRG(TRG_MNOBJU) TRGLIB(*FILE)
          ALWREPCHG(*YES)
```

Odpojuje se CL příkazem

```
RMVPFTRG FILE(VZSQL/OBJDET_T) TRGTIME(*BEFORE) TRGEVENT(*UPDATE)
```

nebo SQL příkazem

```
DROP TRIGGER TRG_MNOBJU
```

CL příkaz CHGPFTRG zneschopní nebo uschopní spouštěč (viz [výše](#)).

Spouštěč můžeme demonstrovat na programu OBJ\_SQLF z knihovny VZSQLPGM.

# Některé postupy při pořizování vydávaných faktur

Příklady jsou vybrány z programu FAK\_SQLF v knihovně VZSQLPGM.

## *Převod dat ze starých souborů jiné knihovny*

```
CREATE TABLE ZAKAZ_T (  
  CZAK DEC(6) GENERATED ALWAYS AS IDENTITY  
    ( START WITH 1 INCREMENT BY 1 CYCLE ) ,  
  NAZZAK CHAR(30) NOT NULL DEFAULT ' ' ,  
  ADRZAK CHAR(20) NOT NULL DEFAULT ' ' )  
RCDFMT ZAKAZ_T
```

Data (kromě čísla zákazníka) se zkopírují z knihovny VZRPGPOKR, souboru ZAKAZ:

```
insert into ZAKAZ_T (NAZZAK, ADRZAK)  
  select  NAZZAK, ADR1  
  from VZRPGPOKR/ZAKAZ  
  order by CZAK
```

Číslo zákazníka CZAK je v tabulce ZAKAZ\_T definováno jako IDENTITY automaticky se zvyšující od 1 po 1, takže při vkládání nového řádku do tabulky se dosadí nově vytvořené číslo zákazníka. Proto je v příkazech INSERT i SELECT sloupec CZAK vynechán.

Je-li třeba kopírování z nějakého důvodu opakovat, lze příkazem ALTER TABLE číslování obnovit, tj. nastavit na jinou hodnotu:

```
alter table ZAKAZ_T alter column CZAK restart with 1
```

## *Číslování faktur*

Číslo faktury se v programu čerpá z SQL objektu “**sekvence**” CISLO\_FAKTURY, která je definována příkazem

```
CREATE SEQUENCE CISLO_FAKTURY AS DECIMAL (6, 0)  
  START WITH 1 INCREMENT BY 1  
  NO MAXVALUE  
  NO CYCLE  
  CACHE 20
```

Sekvence je ve skutečnosti objekt typu \*DTAARA uložený v knihovně VZSQL a má jiné jméno CISLO00001 (předělané kvůli max. 10 znakům jmen). Uplatní se v příkazu INSERT vytvářejícím novou hlavičku faktury přes pohled FAKHLA.

```
// Nový řádek hlavičky s automaticky zvýšeným číslem faktury  
// ze sekvence CISLO_FAKTURY  
Exec SQL insert into FAKHLA  
  ( CFAK, SFAK, CZAK, COBJZ, DTFAK, DTPLA, CENFAK, UHRADA )  
  values  
  ( NEXT VALUE FOR CISLO_FAKTURY,  
    :SFAK, :CZAK, :COBJZ, :DTFAK, :DTPLA, :CENFAK, :UHRADA );  
// Totéž číslo faktury uložím do proměnné pro zobrazení  
Exec SQL values PREVIOUS VALUE FOR CISLO_FAKTURY into :CFAK;
```

Hodnotu sekvence lze změnit příkazem ALTER SEQUENCE – s opatrností!

```
alter sequence CISLO_FAKTURY restart with 1
```

## ***Získání ceny faktury z detailů do hlavičky***

Při pořizování a údržbě faktury se do hlavičky spočítá celková cena fakturovaného zboží jako součet součinu jednotkové ceny a množství. Ta se pak zapíše do hlavičky faktury.

```
exec SQL update FAKHLA set CENFAK =  
  ( select SUM(C.CENAJ * D.MNFAK)  
    from FAKDET as D  
   join CENYF as C on D.CZBOZIF = C.CZBOZIF  
   where D.CFAK = :CFAK )  
where CFAK = :CFAK;
```

## ***Odstranění osiřelých řádků před přidáním referenčního omezení***

Referenční omezení na číslo faktury v detailech (foreign key) proti číslu faktury v hlavičce (parent key) je přidáno k tabulce FAKDET\_T takto:

```
ALTER TABLE FAKDET_T ADD CONSTRAINT HLAVICKA_FAKTURY_EXISTUJE  
  FOREIGN KEY (CFAK) REFERENCES FAKHLA_T (CFAK)  
  ON DELETE CASCADE ON UPDATE NO ACTION
```

Dokud však existují detailní řádky obsahující čísla neexistujících faktur (hlaviček), nelze kvůli nim referenční omezení přidat. Takové řádky zjistíme příkazem

```
select D.CFAK from FAKDET_T D  
  exception join FAKHLA_T H on D.CFAK = H.CFAK
```

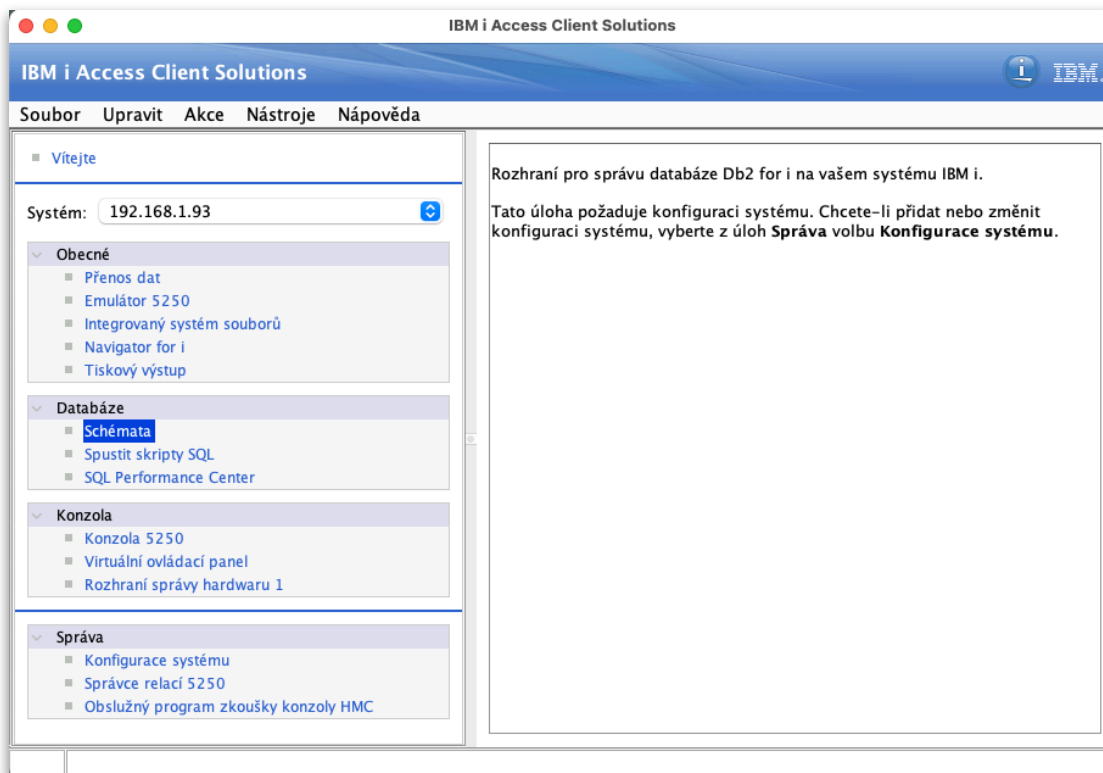
a můžeme je odstranit příkazem

```
delete from FAKDET_T where CFAK in  
( select D.CFAK from FAKDET_T D  
  exception join FAKHLA_T H on D.CFAK = H.CFAK )
```

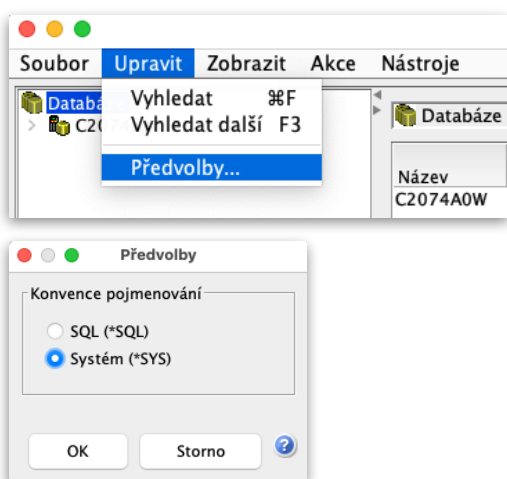
# Volba prostředí SQL v ACS a ve skriptu

ACS je zkratka pro *IBM i Access - Client Solutions*. Jde o nástroj k běžným úlohám spojeným se systémem IBM i, viz stránku <https://www.ibm.com/support/pages/ibm-i-access-client-solutions>.

Po spuštění aplikace se ukáže základní okno, z něhož vybereme položku *Schémata* a pokračujeme naznačeným postupem.



## Nastavení jmenné konvence pro databázi



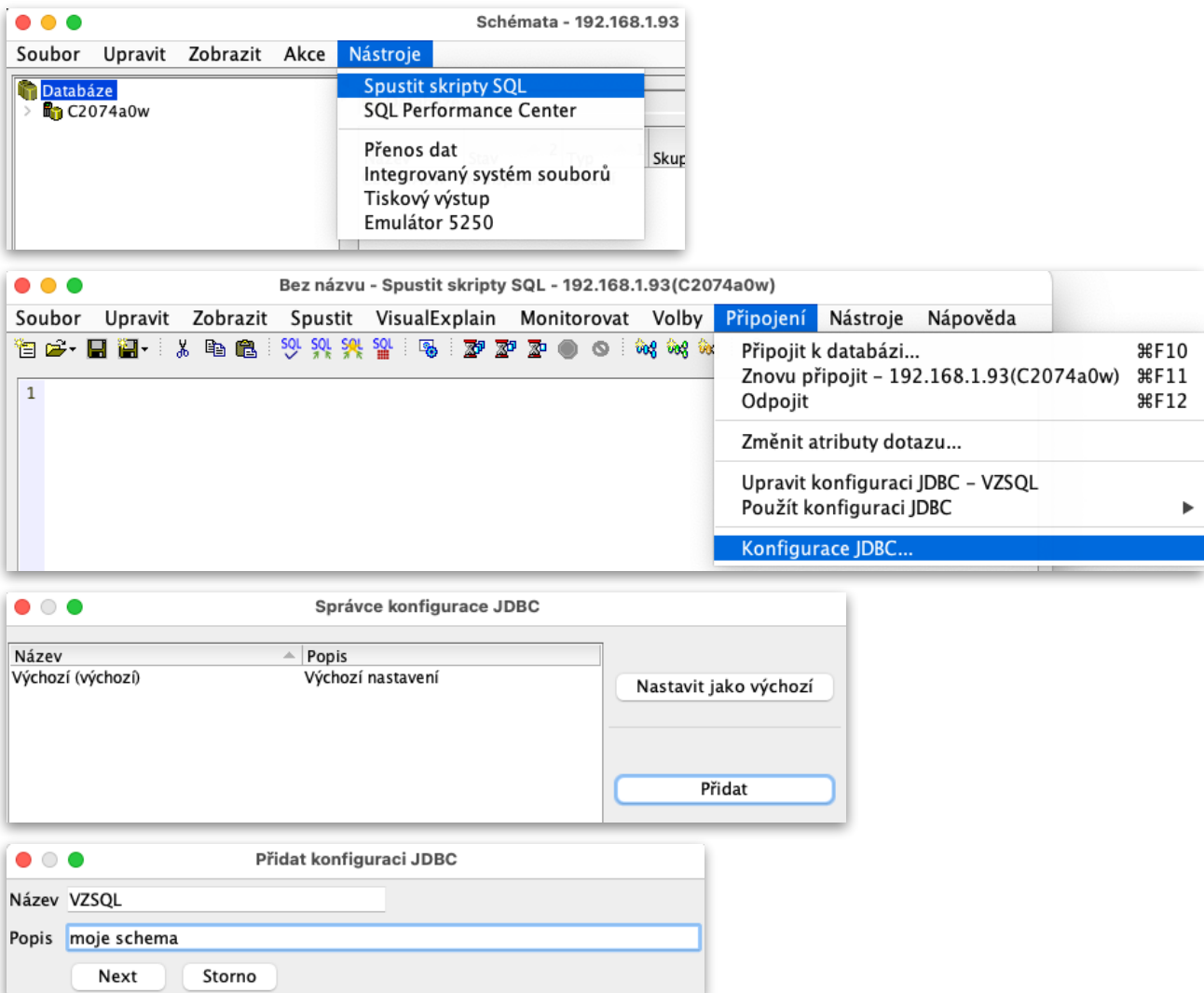
V CL příkazu RUNSQLSTM je k tomu účelu parametr NAMING s hodnotou \*SYS nebo \*SQL. \*SYS je předvolená hodnota.

Poznámka: V předkompilátoru programovacího jazyka k tomu slouží příkaz

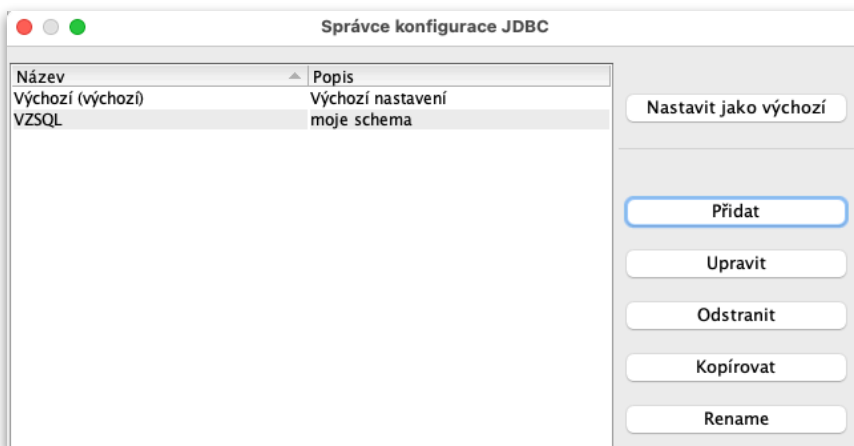
**SET OPTION NAMING = \*SYS nebo \*SQL;**

který ve skriptu nelze použít.

## Nastavení předvoleného schematu (knihovny) pro SQL



Zvolíme *Next* a v dalším okně *Uložit*. Objeví se následující okno, kde stisknu *Nastavit jako výchozí*.



Tento postup je rovnocenný s příkazem

**SET SCHEMA = 'VZSQL';**

ve skriptu ACS nebo v CL skriptu RUNSQLSTM.

V CL příkazu RUNSQLSTM je pro tento účel k dispozici také parametr DFTRDBCOL(VZSQL).

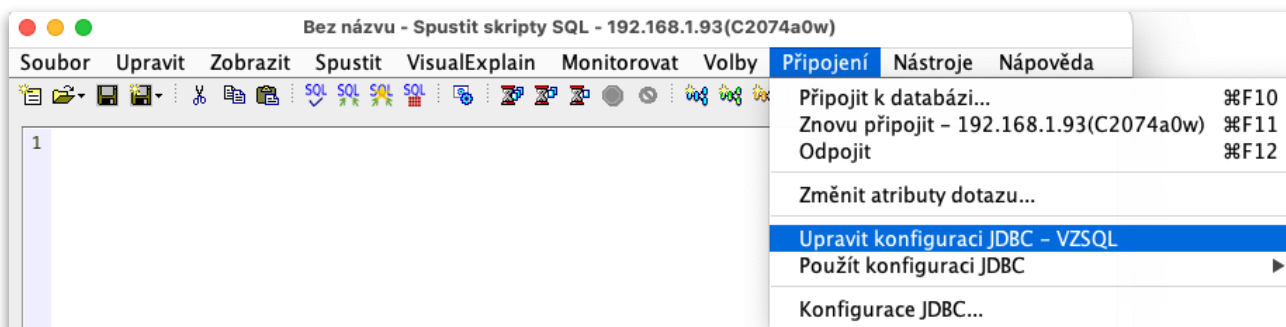


Poznámka: V předkompilátoru programovacího jazyka k tomu slouží příkaz

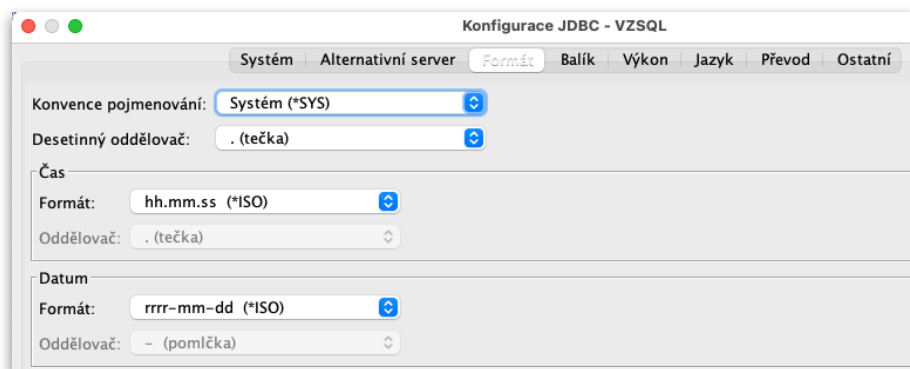
```
SET OPTION DFTRDBCOL = 'VZSQL';
```

který ve skriptu nelze použít.

## Nastavení českého prostředí pro SQL



Volí se v záložce *Jazyk* okna *Konfigurace JDBC - VZSQL*.



V CL příkazu RUNSQLSTM jsou k tomu účelu parametry SRTSEQ(\*LANGIDSHR) a LANGID(CSY).

Poznámka: V předkompilátoru programovacího jazyka k tomu slouží příkaz

```
SET OPTION SRTSEQ = *LANGIDSHR, LANGID = CSY;
```

který ve skriptu nelze použít.

# Opisy programů z příkladů

## Obrazovkový soubor OBJW

```
A*%TS SD 20020806 182338 QPGMR REL-V4R4M0 5769-PW1
A*****
A* Obrazovkový soubor pro pořizování a opravy objednávek
A* (pro program OBJ_SQLF)
A*****
A*%EC
A DSPSIZ(24 80 *DS3 -
A 27 132 *DS4)
A REF(*LIBL/REF)
A CF03(03 'Konec')
A CF12(12 'Návrat')
A*****
A* Okno - Seznam objednávek
A*****
A R OBJWF0
A*%TS SD 19980616 115321 QPGMR REL-V3R7M0 5716-PW1
A WINDOW(*DFT 18 58)
A WDWBORDER((*DSPATR RI) (*CHAR ' -
A '))
A WDWTITLE((*TEXT ' Objednávky '))
A 16 1'F3=Konec'
A COLOR(BLU)
A 16 11'F6=Nová objednávka'
A COLOR(BLU)
A 16 32'F12=Návrat'
A COLOR(BLU)
A 17 1'F23=Zrušit objednávku, na niž ukaz-
A uje kurzor'
A COLOR(YLW)
A SFL
A R OBJWS0
A*%TS SD 19980511 122706 QPGMR REL-V3R7M0 5716-PW1
A CDOD R H TEXT('Číslo dodavatele')
A COBJ R O 5 2TEXT('Číslo objednávky')
A DTOBJ R O 5 10TEXT('Datum objednávky')
A NAZDOD R O 5 22TEXT('Název dodavatele')
A*
A R OBJWC0
A*%TS SD 19980616 115658 QPGMR REL-V3R7M0 5716-PW1
A CF06(06 'Nová objednávka')
A CF23(23 'Zrušit objedn.')
A OVERLAY
A SFLCSRNRN(&SFKUR0)
A 47 SFLDSP
A SFLDSPCTL
A 48 SFLCLR
A 47 SFLEND
A SFLSIZ(0500)
A SFLPAG(0010)
A 80 SFLMSG('Ukažte na řádek!' 80)
A 60 SFLMSG('Transakce se nedokončila' 6-
A 0)
A WINDOW(OBJWF0)
A SFKUR0 5S 0H
A KUR0 4S 0H SFLRCDNBR(CURS0R)
A 1 1'OBJWF0'
A 1 12'Seznam objednávek'
A DSPATR(HI)
A DNES L 1 42
A 3 2'Č.obj.'
A DSPATR(HI)
A 3 10'Datum obj.'
A DSPATR(HI)
A 3 22'Název dodavatele'
A DSPATR(HI)
A*****
```

```

A*   Okno - Detaily objednávky
A*****
A           R OBJWF1
A**%TS SD 19980616 115658 QPGMR REL-V3R7M0 5716-PW1
A           WINDOW(*DFT 18 58)
A           WDWBORDER((*DSPATR RI) (*CHAR ' -
A           '))
A           WDWTTITLE((*TEXT ' Detaily ') (*CO-
A           LOR RED))
A           16 1'F3=Konec'
A           COLOR(BLU)
A           16 11'F12=Návrat'
A           COLOR(BLU)
A           16 23'F5=Obnova'
A           COLOR(BLU)
A           16 34'F6=Nové zboží'
A           COLOR(BLU)
A           17 1'F23=Zrušit zboží, na něž ukazuje k-
A           urzor'
A           COLOR(YLW)
A           SFL
A           R OBJWS1
A**%TS SD 19980419 094425 QPGMR REL-V3R7M0 5716-PW1
A           COBJ R H
A           CDOD R H
A           CZBOZID R O 5 6
A           MNOBJ R B 5 12EDTCDE(P)
A           NAZZBO R O 5 24
A*
A           R OBJWC1
A**%TS SD 19980616 115658 QPGMR REL-V3R7M0 5716-PW1
A           CF05(05 'Obnova')
A           CF06(06 'Nový detail')
A           CF23(23 'Zrušit detail')
A           OVERLAY
A           SFLCSRRRN(&SFKUR1)
A           27 SFLDSP
A           SFLDSPCTL
A           28 SFLCLR
A           27 SFLEND
A           SFLSIZ(0500)
A           SFLPAG(0010)
A           81 SFLMSG('Ukažte na řádek!')
A           WINDOW(OBJWF1)
A           SFKUR1 5S 0H
A           1 1'OBJWF1'
A           1 10'Objednávka č.'
A           DSPATR(HI)
A           COBJ R O 1 24TEXT('Číslo objednávky')
A           DNES L 1 44
A           3 6'Číslo'
A           DSPATR(HI)
A           3 12'Objednané'
A           DSPATR(HI)
A           3 23'Název zboží'
A           DSPATR(HI)
A           4 6'zboží'
A           DSPATR(HI)
A           4 12'množství'
A           DSPATR(HI)
A*****
A*   Okno - Ceny a názvy zboží
A*****
A           R OBJWF2
A**%TS SD 19980616 115658 QPGMR REL-V3R7M0 5716-PW1
A           WINDOW(*DFT 14 52)
A           WDWBORDER((*DSPATR RI) (*CHAR ' -
A           '))
A           WDWTTITLE((*TEXT ' Zboží ') (*COLO-
A           R RED))
A           13 1'F3=Konec'
A           COLOR(BLU)

```

```

A                                     13 11'F12=Návrat'
A                                     COLOR(BLU)
A                                     SFL
A      R OBJWS2
A      CZBOZID      R      O 5 1TEXT('Číslo zboží')
A      CENAJ      R      O 5 7TEXT('Cena za jednotku')
A                                     EDTCDE(4)
A      NAZZBO      R      O 5 18TEXT('Název zboží')
A      R OBJWC2
A      SFLCTL(OBJWS2)
A**%TS SD 19980616 115658 QPGMR      REL-V3R7M0 5716-PW1
A      PAGEUP(25 'Page up')
A      PAGEDOWN(26 'Page down')
A      OVERLAY
A      SFLCSRNRN(&SFKUR2)
A 37      SFLDSP
A      SFLDSPCTL
A 38      SFLCLR
A 37      SFLEND
A      SFLSIZ(0007)
A      SFLPAG(0007)
A      WINDOW(OBJWF2)
A      SFKUR2      5S 0H
A      1 1'OBJWF2'
A      1 10'Začátek názvu:'
A      DSPATR(HI)
A      DNES      L      1 40
A      ZNAZZB      R      B 2 10REFFLD(NAZZBO)
A      CHECK(LC)
A      3 1'Číslo      Cena/j. Název zboží'
A      DSPATR(HI)
A      4 1'zboží'
A      DSPATR(HI)
A*****
A*      Okno - Dodavatelé
A*****
A      R OBJWF3
A**%TS SD 19980616 115658 QPGMR      REL-V3R7M0 5716-PW1
A      WINDOW(*DFT 14 65)
A      WDWBORDER(( *DSPATR RI) (*CHAR ' -
A      '))
A      WDWTTITLE(( *TEXT ' Dodavatelé ') (-
A      *COLOR RED))
A      13 1'F3=Konec'
A      COLOR(BLU)
A      13 13'F12=Návrat'
A      COLOR(BLU)
A      *
A      R OBJWS3      SFL
A**%TS SD 19980419 093256 QPGMR      REL-V3R7M0 5716-PW1
A      CDOD      R      O 5 1TEXT('Číslo dodavatele')
A      NAZDOD      R      O 5 10TEXT('Název dodavatele')
A      ADRDOD      R      O 5 42TEXT('Adresa dodavatele')
A*
A      R OBJWC3      SFLCTL(OBJWS3)
A**%TS SD 19980616 115658 QPGMR      REL-V3R7M0 5716-PW1
A      PAGEUP(25 'Page up')
A      PAGEDOWN(26 'Page down')
A      OVERLAY
A      SFLCSRNRN(&SFKUR3)
A 47      SFLDSP
A      SFLDSPCTL
A 48      SFLCLR
A 47      SFLEND
A      SFLSIZ(0700)
A      SFLPAG(0007)
A      WINDOW(OBJWF3)
A      SFKUR3      5S 0H
A      KUR3      4S 0H      SFLRCDNBR(CURSOR)
A      1 1'OBJWF3'
A      1 10'Začátek názvu dodavatele:'
A      DSPATR(HI)
A      DNES      L      1 42

```

```

A          ZNAZD          30A B 2 10CHECK(LC)
A          3 1'Číslo'
A          DSPATR(HI)
A          3 10'Název dodavatele'
A          DSPATR(HI)
A          3 42'Adresa dodavatele'
A          DSPATR(HI)
A          4 1'dodav'
A          DSPATR(HI)
A*****
A*   Záhloví objednávky
A*****
A          R OBJWF4
A*%TS SD 19980616 115658 QPGMR REL-V3R7M0 5716-PW1
A 06 CF04(04 'Dodavatelé')
A WINDOW(*DFT 10 54)
A WDWBORDER(( *DSPATR RI) (*CHAR ' -
A '))
A WDWTITLE((*TEXT ' Záhloví objednávk-
A y ')(*COLOR RED))
A 1 1'OBJWF4'
A 2 16'Záhloví objednávky'
A DSPATR(HI)
A DNES L 2 40
A 3 1'Číslo objednávky:'
A DSPATR(HI)
A COBJ R B 3 20TEXT('Číslo objednávky')
A 85 ERRMSG('Objednávka již existuje' 85)
A 4 1'Číslo dodavatele:'
A DSPATR(HI)
A CDOD R B 4 20TEXT('Číslo dodavatele')
A CHGINPDFT
A DSPATR(PR)
A DSPATR(UL)
A CHANGE(31 'Zápis z kláv.')
A BLANKS(32 'Prázdné pole')
A 84 ERRMSG('Chybné číslo dodavatele' 84)
A 5 1'Název dodavatele:'
A DSPATR(HI)
A NAZDOD R O 5 20TEXT('Název dodavatele')
A 6 1'Datum objednávky:'
A DSPATR(HI)
A DTOBJ R B 6 20
A TEXT('Datum objednávky')
A 9 1'F3=Konec'
A COLOR(BLU)
A 9 13'F12=Návrat'
A COLOR(BLU)
A 06 9 28'F4=Dodavatelé'
A COLOR(BLU)
A*****
A*   Detail objednávky
A*****
A          R OBJWF5
A*%TS SD 20020806 182338 QPGMR REL-V4R4M0 5769-PW1
A *DS3 WINDOW(*DFT 11 54)
A *DS4 WINDOW(*DFT 11 54)
A CF04(04 'Zboží')
A WDWBORDER(( *DSPATR RI) (*CHAR ' -
A '))
A WDWTITLE((*TEXT ' Detail objednávk-
A y ')(*COLOR RED))
A 1 1'OBJWF5'
A 1 10'Detail objednávky'
A DSPATR(HI)
A COBJ R O 1 28TEXT('Číslo objednávky')
A DNES L 1 40
A 3 1'Číslo dodavatele:'
A DSPATR(HI)
A CDOD R O 3 20TEXT('Číslo dodavatele')
A 4 1'Název dodavatele:'

```

```

A          DSPATR(HI)
A          NAZDOD      R      O  4 20TEXT('Název dodavatele')
A          5 1'Datum objednávky:'
A          DSPATR(HI)
A          DTOBJ      R      O  5 20
A          TEXT('Datum objednávky')
A          7 1'Číslo zboží      :'
A          DSPATR(HI)
A          CZBOZID    R      B  7 20TEXT('Číslo zboží')
A 82          ERRMSG('Zboží je již v objednávce' -
A          82)
A 83          ERRMSG('Zboží není v ceníku' 83)
A          8 1'Množství v ks      :'
A          DSPATR(HI)
A          MNOBJ      R      B  8 20TEXT('Objednávané množství')
A          EDTCDE(P)
A 04          DSPATR(PC)
A          10 1'F3=Konec'
A          COLOR(BLU)
A          10 13'F12=Návrat'
A          COLOR(BLU)
A          10 28'F4=Zboží'
A          COLOR(BLU)
A*****
A*   Upozornění obsluhy na neukončenou poslední transakci
A*****
A          R OBJWF6
A%%TS SD 19980625 210759 QPGMR REL-V3R7M0 5716-PW1
A          3 23'Poslední transakce se nedokončila!'
A          COLOR(RED)
A          DSPATR(RI)
A          6 11'Poslední dokončená transakce uživa-
A          tele'
A          DSPATR(HI)
A          DSPATR(UL)
A          USERID      R      O  6 50
A          DSPATR(UL)
A          6 61'je tato:'
A          DSPATR(HI)
A          DSPATR(UL)
A          8 11'Druh transakce:'
A          KODTR      R      O  8 32
A          DSPATR(RI)
A          10 11'Číslo objednávky:'
A          COBJ      R      O 10 32
A          12 11'Dodavatel:'
A          CDOD      R      O 12 32
A          NAZDOD    R      O 12 40
A          14 11'Zboží:'
A          CZBOZID    R      O 14 32
A          NAZZBO    R      O 14 40
A          16 11'Objednané množství:'
A          MNOBJ      R      O 16 32
A          EDTCDE(Q)
A          19 11'Stiskněte klávesu ENTER a záznam o-
A          poslední dokončené transakci'
A          COLOR(BLU)
A          20 11'se vymaže.'
A          COLOR(BLU)
A          23 2'F3=Konec (záznam o poslední dokonč-
A          ené transakci se nevymaže).'
A          COLOR(BLU)

```

## Program OBJ\_RPG

```
//*****
//   Program pro pořízení a opravy objednávek zboží od dodavatelů
//   Používá SQL objekty ze schematu (knihovny) VZSQL
//   -----
//*****
Ctl-Opt LangID('CSY') SrtSeq(*LANGIDSHR) DatFmt(*ISO);
//=====
//   P O P I S Y   S O U B O R Ů
//=====
//-----
//   Objednávky - hlavička                klíč: číslo objednávky
Dcl-F OBJHLA_IX                          Usage(*UPDATE:*DELETE:*OUTPUT) Keyed;
//-----
//   Objednávky - detail (zboží)          klíč: číslo objednávky
//                                         číslo zboží
Dcl-F OBJDET_IX                          Usage(*UPDATE:*DELETE:*OUTPUT) Keyed;
//-----
//   Ceny objednávaného zboží            klíč: číslo dodavatele
//                                         číslo zboží
Dcl-F CENYD_IX                          Keyed;
//   Ceny objednávaného zboží            klíč: název zboží
Dcl-F CENYDN_IX                          Keyed
//                                         Rename(CENYD_T: CENYD_T2);
//-----
//   Dodavatelé                          klíč: číslo dodavatele
Dcl-F DODAV_IX                          Keyed;
//   Dodavatelé                          klíč: název dodavatele
Dcl-F DODAVN_IX                          Keyed
//                                         Rename(DODAV_T: DODAV_T2);
//-----
//   Obrazkový soubor
Dcl-F OBJW                                WORKSTN
//                                         Seznam objednávek - hlavičky objednávek
//                                         SFILE(OBJWS0:I0)
//                                         Detaily objednávek
//                                         SFILE(OBJWS1:I1)
//                                         Ceny a názvy zboží
//                                         SFILE(OBJWS2:I2)
//                                         Seznam dodavatelů
//                                         SFILE(OBJWS3:I3);
//=====
//   P R A C O V N Í   Ú D A J E
//=====

//   Pořadové číslo právě zpracovávaného záznamu v podsouborech
Dcl-S I0                                Int(10:0); // hlavičky
Dcl-S I1                                Int(10:0); // detaily
Dcl-S I2                                Int(10:0); // ceny
Dcl-S I3                                Int(10:0); // dodavatelé
//   Počet záznamů v jednotlivých podsouborech
Dcl-S Pocet0                            Int(10:0);
Dcl-S Pocet1                            Int(10:0);
Dcl-S Pocet2                            Int(10:0);
Dcl-S Pocet3                            Int(10:0);

//   Úložné proměnné
Dcl-S CDOD2                             Like(CDOD);
Dcl-S CZBOZ2                             Like(CZBOZID);
Dcl-S MNOBJ2                             Like(MNOBJ);

//   Index pro cyklus
Dcl-S Idx                                Packed(4:0);

//   Maximální počet záznamů v podsouboru obecně
Dcl-C MAXIMUM_SF                        9998;

//   Velikost stránky a podsouboru 2 (zboží) - konst.
Dcl-C PAG2                              7;
```

```

// Velikost stránky a podsouboru 3 (dodavatelé) - konst.
Dcl-C PAG3 7;

// Číslo obrazovkových stránek
Dcl-S PosledniStr Uns(10);
Dcl-S SoucasnaStr Uns(10);

//=====
// H L A V N Í P R O G R A M
//=====

KUR0 = 1; //Nastavit kurzor pro seznam objednávek (podsoubor 0)

// Cyklus 1 -Zpracování objednávek
DoW 0 = 0;
  Exsr PlnitSF0; // Plnit podsoubor 0 seznamem objednávek

// Cyklus 2 - Zpracovat seznam objednávek
DoU 0 = 0;
  Write OBJWF0; // Zapsat návod na funkční klávesy na obrazovku bez čekání na vstup
  DNES = %date(); // dnešní datum
  Exfmt OBJWC0; // Zapsat a číst podsoubor 0 (seznam objednávek) na obrazovce

  If *In03 Or *In12; // F3 nebo F12 - Končit program
    *InLR = *On;
    Return;
  EndIf;

  If *In06; // F6 - Založit novou objednávku (pak pokračujeme jako u Enter)
    Exsr NovaHlaObj; // Nová hlavička objednávky
    If *In12; // F12 - Návrat na zobrazení seznamu objednávek bez zápisu
      Exsr PlnitSF0; // do databáze
      Iter;
    EndIf;
    Exsr ZprDetObj; // Zpracovat detaily nové objednávky
    Exsr ZprHlaObj; // Opakovat zpracování hlavičky objednávky
    // (po F12 v podprogramu ZprDetObj)
    Exsr PlnitSF0; // Opakovat zobrazení seznamu objednávek
    Iter;
  EndIf; // konec F6

  If *In23; // F23 - Zrušit objednávku včetně detailů
    // Vyhledám záznam podsouboru podle pořadového čísla
    // (dostanu klíč - číslo objednávky)
    Chain(E) SFKUR0 OBJWS0;
    *In80 = *Off;
    If Not %Found Or %Error; // Nenalezen/Chyba
      *In80 = *On;
      Iter; // Opakovat cyklus 2
    EndIf;

    Delete COBJ OBJHLA_IX; // Zruším hlavičku, na niž ukazuje kurzor

    // Zruším odpovídající detaily, jestliže nějaké existují
    // Najdu první záznam v databázi OBJDET_T s daným číslem objednávky
    SetLL COBJ OBJDET_IX;
    // Existuje-li, zruším všechny detaily s tímto číslem objednávky
    If %Equal;
      ReadE COBJ OBJDET_IX; // Čtu první záznam se stejným číslem objednávky
      DoW Not %Eof; // Cyklus, dokud není konec skupiny záznamů se stejným
        // číslem objednávky
        Delete OBJDET_T; // Zruším přečtený záznam
        ReadE COBJ OBJDET_IX; // Čtu další záznam se stejným číslem objednávky
      EndDo;
    EndIf; // konec %equal

    // Nastavím kurzor na předchozí záznam podsouboru pro příští
    // zobrazení. Je-li současná poloha kurzoru (SFKUR0)
    // větší než 1, dosadím ji zmenšenou o 1 do KUR0 pro
    // příští zobrazení. Jinak nechám KUR0 (rovno 1).

```



```

If SFKUR0 > 1;
    KUR0 = SFKUR0 - 1;
EndIf;

// Opakovat zpracování od začátku cyklu (s plněním podsouboru)
Exsr PlnitSF0;
Iter;
EndIf; // F23

// Enter:

// Nastavit kurzor na posledně zpracovávanou objednávku
// pro příští zobrazení (podle aktuální pozice kurzoru
// v podsouboru SFKUR0). Je-li chybná, nechat dosavadní.
If SFKUR0 >= 1 And SFKUR0 <= Pocet0;
    KUR0 = SFKUR0;
EndIf;

Chain(E) SFKUR0 OBJWS0; // Vyhledat záznam podsouboru podle poř. čísla
*In80 = *Off;           // (dostaneme klíč - číslo objednávky)
If Not %Found Or %Error;
    *In80 = *On;        // Nenalezen/Chyba - opak. cyklus 2
    Iter;
EndIf;

// Cyklus 3 - Zpracovat objednávku (hlavičku i detaily)
DoW 0 = 0;
    Exsr ZprHlaObj; // Zpracování hlavičky objednávky

    If *In12; // F12 po zpracování hlavičky - Zobrazit seznam objednávek
        Exsr PlnitSF0;
        Leave;
    EndIf;

    Exsr ZprDetObj; // Zpracovat detaily objednávky

// Znovu zpracovat objednávku
EndDo; // (Cyklus 3 - konec)

// Znovu zpracovat seznam objednávek (bez nového plnění SF0)
EndDo; // (Cyklus 2 - konec)

EndDo; // (Cyklus 1 - konec)

//=====
// P O D P R O G R A M Y
//=====
//-----
// ZprHlaObj - Zpracovat hlavičku objednávky
//-----
BegSr ZprHlaObj;

// Cyklus - Zobrazit hlavičku objednávky
DoW 0 = 0;
    Exfmt OBJWF4;
    If *In03; // F3 - Konec programu
        *InLR = *On;
        Return;
    EndIf;

// F4 - Zpracovat dodavatele a návrat k zobrazení hlavičky obj.
If *In04;
    Exsr ZprDodav;
    Iter;
EndIf;

// F12 - Končit podprogram ZprHlaObj bez zápisu hlavičky objednávky
If *In12;
    Leave;
EndIf;

```

```

// Enter:

CDOD2 = CDOD; // Schovat číslo dodavatele pro aktualizaci
Chain COBJ OBJHLA_IX; // Číst hlavičku objednávky z databáze
If Not %Found;
    Iter;
EndIf;

// Číst dodavatele z databáze podle čísla z hlavičky objedn.
// v databázi DODAV_T
Chain CDOD2 DODAV_IX;

// Změněné nebo prázdné č. dodavatele nebo nenalezeno - chyba
*In84 = *Off;
If *In31 Or *In32 Or Not %Found;
    *In84 = *On;
    Iter;
EndIf;

// Přepsat hlavičku (i když se nezměnil dodavatel)
CDOD = CDOD2;
Update OBJHLA_T;

// Když byl vstup z klávesnice do pole CDOD
// nebo bylo pole prázdné - Znovu zobrazit
// If *In31 Or *In32;
//     Iter;
// EndIf;
// Jinak končit podprogram
Leave;
EndDo;

EndSr; // konec ZprHlaObj

//-----
// ZprDetObj - Zpracovat detaily objednávky
//-----

BegSr ZprDetObj;

Exsr PlnitSF1; // Plnit podsoubor 1 - detaily objednávek

// Cyklus 1 - Zobrazit okno se seznamem detailů objednávek
DoW 0 = 0;
Write OBJWF1;
Exfmt OBJWC1;

If *In03; // F3 - Konec programu
    *InLR = *On;
    Return;
EndIf;

If *In12; // F12 - Návrat z podprogramu ZprDetObj
    Leave;
EndIf;

If *In05; // F5 - Znovu zobrazit nově naplněný seznam detailů objednávek
    Exsr PlnitSF1;
    Iter;
EndIf;

// F6 - Založit nový detail objednávky a znovu naplnit seznam
//     detailů a zobrazit jej
If *In06;
    Exsr NovyDetObj;
    Exsr PlnitSF1;
    Iter;
EndIf;

// F23 - Zrušit detail objednávky a znovu naplnit seznam

```

```

//      (kurzor nastavit na záznam předcházející zrušenému)
If *In23;
// Vyhledat záznam podsouboru podle poř. čísla
// (dostaneme klíč detailu objednávky)
Chain SFKUR1 OBJWS1;
If Not %Found; // Nenalezen - chyba
    *In81 = *On;
    Iter;
Else;
    *In81 = *Off;
EndIf;
// Zrušit detail objednávky, na nějž ukazuje kurzor
// (podle klíče získaného z podsouboru)
Delete (COBJ: CZBOZID) OBJDET_IX;
Exsr PlnitSF1;
Iter;
EndIf;

// Enter - Zpracovat změněné detaily, není-li podsoubor prázdný
//      (aktualizovat změněné množství zboží v databázi)
If Pocet1 > 0;
    ReadC OBJWS1;

    // Cyklus 2
    DoW Not %EOF;
        Exsr Uloz1;
        Chain (COBJ: CZBOZID) OBJDET_IX;
        If %Found(OBJDET_IX);
            Exsr Obnov1;
            Update OBJDET_T;
        EndIf;
        ReadC OBJWS1;
    EndDo; // (Cyklus 2 - konec)

EndIf; // (Cyklus 1 - konec)

EndDo; // konec hlavního cyklu
EndSr; // konec ZprDetObj

//-----
// NovaHlaObj - Pořídít novou hlavičku objednávky
//-----

BegSr NovaHlaObj;

// Vyčistit data a dosadit dnešní datum
Clear OBJHLA_T;
Clear NAZDOD;
DTOBJ = %date(); // Nové datum

// Cyklus - Zobrazit prázdnou hlavičku objednávky
DoW 0 = 0;
    *In33 = *On;
    Exfmt OBJWF4;

    If *In03; // F3 - Konec programu
        *InLR = *On;
        Return;
    EndIf;

    If *In12; // F12 - Návrat
        Leave;
    EndIf;

    // F4 - Zpracovat dodavatele a znovu zobrazit hlavičku objednávky
    If *In04;
        Exsr ZprDodav;
        Iter;
    EndIf;

    // Enter:

```

```

// Číst dodavatele z databáze DODAV_T podle čísla z obrazovky
Chain CDOD DODAV_IX;
*In84 = *Off;
If Not %Found;
    *In84 = *On;
    Iter;
EndIf;

// Zapsat novou hlavičku, není-li již v databázi
SetLL COBJ OBJHLA_IX;
If %Equal;
    Eval *In85 = *On; // Už tam je - chyba
    Iter; // Opakovat
Else;
    Eval *In85 = *Off;
EndIf;
Write OBJHLA_T; // Zapsat
Leave;

EndDo; // (Cyklus - konec)

EndSr;

//-----
// NovyDetObj - Pořídít nový detail objednávky (zboží)
//-----

BegSr NovyDetObj;

// Vyčistit údaje pro obrazovku
Clear CZBOZID;
Clear MNOBJ;

// Cyklus - Zobrazit okno pro pořízení nového detailu objednávky
DoW 0 = 0;
Exfmt OBJWF5;

If *In03; // F3 - Konec programu
    *InLR = *On;
    Return;
EndIf;

If *In12; // F12 - Návrat
    Leave;
EndIf;

If *In04; // F4 - Zobrazit a zpracovat seznam zboží
    Exsr ZprCeny; // Zobr. zboží
    Iter; // a znovu detaily
EndIf;

// Enter:

// Číst zboží v ceníku
Chain (CDOD: CZBOZID) CENYD_IX;
If Not %Found;
    *In83 = *On;
    Iter; // Není tam - chyba
Else;
    *In83 = *Off;
EndIf;

// Zapsat detail zboží do objednávky
SetLL (COBJ: CZBOZID) OBJDET_IX; // Najít detail (zboží) v objednávce
If %Equal;
    *In82 = *On;
    Iter; // Je tam - chyba
Else;
    *In82 = *Off;
EndIf;

```

```

        // Není-li detail (zboží) v objednávce - Zapsat ho tam
        Write OBJDET_T;
        Leave;

        EndDo; // (Cyklus - konec)

    EndSr;

//-----
// PlnitSF0 - Plnit podsoubor 0 - hlavičky objednávek
//-----

BegSr PlnitSF0;

    // Vymazat podsoubor 0 - hlavičky objednávky
    *In48 = *On;
    Write OBJWF0;
    Write OBJWC0;
    *In48 = *Off;
    // Vynulovat čítač záznamů podsouboru
    I0 = 0;

    // Přecházet první záznam z databáze hlaviček objednávek
    SetLL *LoVal OBJHLA_IX;
    Read(N) OBJHLA_IX;

    // Cyklus, dokud není konec databázového souboru
    DoW Not %EoF And I0 < MAXIMUM_SF;

        Chain CDOD DODAV_IX; // Číst dodavatele podle čísla z obrazovky
        I0 += 1; // Zvýšit čítač záznamů podsouboru

        Write OBJWS0; // Zapsat záznam do podsouboru podle čítače

        Read(N) OBJHLA_IX; // Číst další záznam z databáze hlaviček objednávek
    EndDo;

    // Pamatovat počet záznamů podsouboru. Zapnout ind. 47,
    // je-li počet větší než 0. Umožní to zobrazit stránku
    // na obrazovce.
    Pocet0 = I0;
    If Pocet0 > 0;
        *In47 = *On;
    Else;
        *In47 = *Off;
    EndIf;

EndSr;

//-----
// PlnitSF1 - Plnit podsoubor 1 s detaily objednávky
//-----

BegSr PlnitSF1;

    // Vymazat podsoubor 1 - detaily objednávek
    *In28 = *On;
    Write OBJWF1;
    Write OBJWC1;
    *In28 = *Off;
    I1 = 0 ; // Vynulovat čítač záznamů podsouboru

    // Najít v databázi první záznam skupiny detailů
    // s daným číslem objednávky
    SetLL COBJ OBJDET_IX;
    // Když se našel, zapsat skupinu detailů do podsouboru
    If %Equal;
        // Číst první záznam skupiny s daným číslem objednávky
        // z databáze detailů
        ReadE(N) COBJ OBJDET_IX;

```

```

// Cyklus, dokud se nezmění číslo objednávky
DoW Not %EOF And I1 < MAXIMUM_SF;
Chain (CDOD: CZBOZID) CENYD_IX; // Číst záznam o zboží z databáze podle čísla z obr
I1 += 1; // Zvýšit čítač záznamů podsouboru

Write OBJWS1; // Zapsat záznam do podsouboru podle čítače

ReadE(N) COBJ OBJDET_IX; // Číst další záznam detailů ze stejné objednávky
EndDo;
EndIf;

// Pamatovat počet záznamů podsouboru. Zapnout ind. 27,
// je-li počet větší než 0. Umožní to zobrazit stránku
// na obrazovce.
Pocet1 = I1;
If Pocet1 > 0;
    *In27 = *On;
Else;
    *In27 = *Off;
EndIf;

EndSr;

//-----
// ZprDodav - Získat číslo dodavatele z přehledu dodavatelů
//-----
BegSr ZprDodav;
    // Uschovat číslo dodavatele pro příp. F12
    CDOD2 = CDOD;
    // Vymazat pole obrazovky pro zadání začátku názvu dodavatele
    Clear ZNAZD;
    // Nastavit kurzor na 1. záznam podsouboru
    KUR3 = 1;
    // Plnit podsoubor 3 - dodavatelé
    Exsr PlnitSF3;

    // Cyklus - Zpracovat číslo dodavatele
    DoW 0 = 0;
        // Zobrazit seznam dodavatelů
        Write OBJWF3;
        Exfmt OBJWC3;

        If *In03; // F3 - Konec programu
            *InLR = *On;
            Return;
        EndIf;

        If *In12; // F12 - Vrátit uschované č. dodavatele a končit podprog. ZprDodav
            Eval CDOD = CDOD2;
            Leave;
        EndIf;

        If *In25; // Page up - Vrátit se o stránku zpět (nastavením kurzoru)
            Exsr PgUp3; // a zobrazit ji
            Iter;
        EndIf;

        If *In26; // Page down - Přečíst další záznamy z databáze pro další
            Exsr PlnitSF3D; // stránku a zobrazit ji
            Iter;
        EndIf;

        // Enter:
        // Je-li kurzor mimo stránku ( > 0 a < Pocet3 ) - Znovu plnit
        // podsoubor podle zadaného začátku názvu (ZNAZD)
        If SFKUR3 <= 0 Or SFKUR3 > Pocet3;
            Exsr PlnitSF3;
            Iter;
        EndIf;

        // Vyhledat záznam podsouboru podle polohy kurzoru

```

```

        // (dostaneme číslo dodavatele)
Chain SFKUR3 OBJWS3;
Leave;

EndDo; // (Cyklus - konec)

EndSr;

//-----
// PlnitSF3 - Plnit podsoubor 3 s dodavatelem
//-----

BegSr PlnitSF3;

    // Vymazat podsoubor 3 - dodavatelé
    *In48 = *On;
    Write OBJWF3;
    Write OBJWC3;
    *In48 = *Off;
    // Vynulovat čítač záznamů podsouboru 3
    I3 = 0;

    // Najít v databázi prvního dodavatele podle začátečních písmen
    // nebo nejbližší vyššího podle abecedy
    SetLL ZNAZD DODAVN_IX;

    // Když se našel, zapsat do podsouboru další stránku dodavatelů.
    // (není-li jich v databázi tolik, tak se zapíše méně, popř. nic).
    If %Found;
        Read DODAVN_IX;
        // Cyklus, dokud není konec databáze nebo není vyčerpán
        // počet řádků obrazovkové stránky (Pag3)
        DoW Not %Eof And I3 < Pag3;
            I3 += 1; // Zvýšit čítač záznamů podsouboru
            Write OBJWS3; // Zapsat záznam do podsouboru podle čítače
            Read DODAVN_IX; // Číst další záznam z databáze dodavatelů
        EndDo;
    EndIf; // konec %found

    // Pamatovat počet záznamů podsouboru. Zapnout ind. 47,
    // je-li počet větší než 0. Umožní to zobrazit stránku
    // na obrazovce.
    Pocet3 = I3;
    If Pocet3 > 0;
        *In47 = *On;
    Else;
        *In47 = *Off;
    EndIf;

    KUR3 = 1;

EndSr;

//-----
// PgUp3 - Nastavit předchozí stránku po Page up u dodavatelů
//-----

BegSr PgUp3;

    // Je-li současná poloha kurzoru větší než velikost stránky,
    // odečíst velikost stránky od nastavení kurzoru pro
    // příští zobrazení (Kurzor3). Jinak nastavit číslo 1.
    If SFKUR3 > Pag3;
        Eval KUR3 -= Pag3;
    Else;
        Eval KUR3 = 1;
    EndIf;

EndSr;

//-----

```

```

// PlnitSF3D - Plnit podsoubor 3 s dodavateli po Page down
//-----

BegSr PlnitSF3D;

    // Spočítat číslo poslední stránky podsouboru
    PosledniStr = %Div(I3: Pag3);
    If %Rem(I3: Pag3) = 0;
        Eval PosledniStr -= 1;
    EndIf;

    // Spočítat číslo právě zobrazené stránky
    SoucasnaStr = %Div(SFKUR3: Pag3);
    If %Rem(SFKUR3: Pag3) = 0;
        Eval SoucasnaStr -= 1;
    EndIf;

    // Když zobrazená stránka není poslední stránka podsouboru,
    // (číslo zobrazené stránky je menší než číslo poslední str.)
    // nastavit kurzor o jednu stránku dál
    If SoucasnaStr < PosledniStr;

        KUR3 = SFKUR3 + Pag3;

    Else; // Když to je poslední stránka - Přečíst další záznamy z DB

        Chain I3 OBJWS3; // Přečíst poslední záznam podsouboru podle čítače
        SetGT NAZDOD DODAVN_IX; // Nastavit databázi na vyšší název dodavatele
        Read DODAVN_IX; // Přečíst nastavený záznam z databáze

        Idx = I3 + Pag3; // Hranice cyklu - Čítač plus velikost stránky

        // Cyklus, dokud není konec souboru dodavatelů nebo
        // není dosažena hranice cyklu Idx
        DoW Not %Eof And I3 < Idx;
            I3 += 1; // Zvýšit čítač záznamů podsouboru
            Write OBJWS3; // Zapsat záznam do podsouboru podle čítače
            Read DODAVN_IX; // Číst další záznam z databáze dodavatelů
        EndDo;

        // Pamatovat počet záznamů podsouboru. Zapnout ind. 47,
        // je-li počet větší než 0. Umožní to zobrazit stránku
        // na obrazovce.
        Pocet3 = I3;
        If Pocet3 > 0;
            *In47 = *On;
        Else;
            *In47 = *Off;
        EndIf;

        KUR3 = I3; // Nastavit kurzor na poslední záznam podsouboru

    EndIf; // konec testu na poslední stránku

EndSr;

//-----
// Uloz1 - Uložit změněná data z podsouboru 1
//-----
BegSr Uloz1;
    MNOBJ2 = MNOBJ;
EndSr;

//-----
// Obnov1 - Obnovit změněná data z podsouboru 1
//-----
BegSr Obnov1;
    MNOBJ = MNOBJ2;
EndSr;

//-----

```



```

// ZprCeny - Získat číslo a název zboží z tabulky CENYD_T
//-----

BegSr ZprCeny;

    // Uschovat hodnoty z obrazovky (pro příp. F12)
    CZBOZ2 = CZBOZID;
    MNOBJ2 = MNOBJ ;
    // Vymazat pole pro zadání začátku názvu zboží
    Clear ZNAZZB ;

    // Plnit podsoubor 2 - zboží
    Exsr PlnitSF2;

    // Cyklus 1:
    DoW 0 = 0;
    // Zobrazit seznam zboží
    Write OBJWF2;
    Exfmt OBJWC2;

    If *In03; // F3 - Konec programu
        *InLR = *On;
        Return;
    EndIf;

    // F12 - Vrátit uschované hodnoty a končit podprog. ZprCeny
    If *In12;
        CZBOZID = CZBOZ2;
        MNOBJ = MNOBJ2;
        Leave;
    EndIf;

    // Page up - Naplnit předchozí stránku předchozími záznamy
    //          databáze a zobrazit tuto stránku
    If *In25;
        Exsr PlnitSF2U;
        Iter;
    EndIf;

    // Page down - Naplnit další stránku dalšími záznamy
    //          databáze a zobrazit tuto stránku
    If *In26;
        Exsr PlnitSF2D;
        Iter;
    EndIf;

    // Enter:
    // Je-li kurzor mimo stránku ( > 0 a < Pocet2 ) - Znovu plnit
    // podsoubor podle zadaného začátku názvu (ZNAZZB)
    If SFKUR2 <= 0 Or SFKUR2 > Pocet2;
        Exsr PlnitSF2;
        Iter;
    EndIf;

    // Vyhledat záznam podsouboru podle polohy kurzoru
    // (dostaneme číslo zboží)
    Chain SFKUR2 OBJWS2;
    Leave;
EndDo; // konec Cyklu 1

EndSr;

//-----
// PlnitSF2 - Plnit podsoubor 2 se zbožím od zadaného začátku
//          názvu zboží (ZNAZZB) - jen jednu stránku
//-----
BegSr PlnitSF2;
    // Vymazat podsoubor 2 - zboží
    *in38 = *on;
    Write OBJWF2;
    Write OBJWC2;

```

```

*in38 = *off;
// Vynulovat čítač záznamů podsouboru 2
I2 = 0;

CDOD2 = CDOD;

// Najít v databázi první název zboží podle začátečních písmen
// nebo nejbližší vyšší podle abecedy
SetLL ZNAZZB CENYDN_IX;

// Když se našel, zapsat do podsouboru další stránku zboží
If %Found;
  Read CENYDN_IX;
  // Cyklus, dokud není konec databáze nebo není vyčerpán
  // počet řádků obrazovkové stránky (Pag2)
  DoW Not %EOF And I2 < Pag2;
    If CDOD = CDOD2; // Jen zboží daného dodavatele
      I2 += 1; // Zvýšit čítač záznamů podsouboru
      Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
    EndIf;
    Read CENYDN_IX; // Číst další záznam z databáze zboží
  EndDo;
EndIf; // %Found

// Pamatovat počet záznamů podsouboru. Zapnout ind. 37,
// je-li počet větší než 0. Umožní to zobrazit stránku
// na obrazovce.
Pocet2 = I2;
If Pocet2 > 0;
  *In37 = *On;
Else;
  *In37 = *Off;
EndIf;

EndSr;

//-----
// PlnitSF2U - Plnit podsoubor 2 se zbožím pozpátku od dosavadního
// prvního názvu zboží v podsouboru
//-----
BegSr PlnitSF2U;
// Vynulovat čítač záznamů podsouboru 2
I2 = 0;
// Najít první dosavadní název zboží v podsouboru
// (podsoubor má jen jednu stránku!)
// Když neexistuje, NAZZBO se nezmění.
Chain(E) 1 OBJWS2;
//.....
// ALTERNATIVA 1 (s opakováním řádku)
// Nastavit ukazatel do databáze ZA tento název (nebo na konec)
// (na příští stránce se bude opakovat řádek z minulé stránky)
SetGT NAZZBO CENYDN_IX;
//.....
// ALTERNATIVA 2 (bez opakování řádku)
// Nastavit ukazatel do databáze PŘED tento název
// (na příští stránce se neopakuje žádný řádek z minulé str.)
//SetLL NAZZBO CENYDN_IX;
//.....

// Přecházet Pag2 (nebo méně) databázových vět pozpátku
ReadP CENYDN_IX;
DoW Not %EOF And I2 < Pag2;
  I2 += 1;
  ReadP CENYDN_IX;
EndDo;

// Vymazat podsoubor 2 - zboží (má velikost jedné stránky)
*in38 = *on;
Write OBJWF2;
Write OBJWC2;
*in38 = *off;

```

```

// Vynulovat čítač záznamů podsouboru 2
I2 = 0;

// Nastavit ukazatel do databáze PŘED naposledy přečtený název
SetLL NAZZBO CENYDN_IX;

// Naplnit podsoubor záznamy z databáze od tohoto názvu dále
// Číst nastavený záznam z databáze (dopředu)
Read CENYDN_IX;
// Cyklus, dokud není konec databáze nebo není vyčerpán
// počet řádků obrazovkové stránky (Pag2)
DoW Not %Eof And I2 < Pag2;
    I2 += 1; // Zvýšit čítač záznamů podsouboru
    Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
    Read CENYDN_IX; // Číst další záznam z databáze zboží
EndDo;

// Pamatovat počet záznamů podsouboru. Zapnout ind. 37,
// je-li počet větší než 0. Umožní to zobrazit stránku
// na obrazovce.
Pocet2 = I2;
If Pocet2 > 0;
    *In37 = *On;
Else;
    *In37 = *Off;
EndIf;

EndSr;

//-----
// PlnitSF2D - Plnit podsoubor 2 se zbožím dopředu od dosavadního
// posledního názvu zboží v podsouboru
//-----
BegSr PlnitSF2D;

    I2 = 0; // Vynulovat čítač záznamů podsouboru 2

    // Najít poslední dosavadní název zboží v podsouboru
    // (podsoubor má jen jednu stránku!)
    // Když neexistuje, zapne se ind. 86 a NAZZBO se nezmění.
    Chain(E) Pag2 OBJWS2;

    // Vymazat podsoubor 2 - zboží
    *in38 = *on;
    Write OBJWF2;
    Write OBJWC2;
    *in38 = *off;

    I2 = 0; // Vynulovat čítač záznamů podsouboru 2
    //.....
    // ALTERNATIVA 1 (s opakováním řádku)
    // Nastavit ukazatel do databáze PŘED tento název
    // (na příští stránce se opakuje poslední řádek z minulé str.)
    SetLL NAZZBO CENYDN_IX;
    //.....
    // ALTERNATIVA 2 (bez opakování řádku)
    // Nastavit ukazatel do databáze ZA tento název
    // (na příští stránce se neopakuje žádný řádek z minulé str.)
    //SetGT NAZZBO CENYDN_IX;
    //.....

    // Číst nastavený záznam z databáze
    Read CENYDN_IX;

    // Cyklus, dokud není konec databáze nebo není vyčerpán
    // počet řádků obrazovkové stránky (Pag2)
    DoW Not %Eof And I2 < Pag2;
        I2 += 1; // Zvýšit čítač záznamů podsouboru
        Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
        Read CENYDN_IX; // Číst další záznam z databáze zboží
    EndDo;

```

```

// Pamatovat počet záznamů podsouboru. Zapnout ind. 37,
//   je-li počet větší než 0. Umožní to zobrazit stránku
//   na obrazovce.
Pocet2 = I2;
If Pocet2 > 0;
    *In37 = *On;
Else;
    *In37 = *Off;
EndIf;

EndSr;

```

## Program OBJ\_SQLF

```
//*****
//  OBJ_SQLF
//  Program pro pořízení a opravy objednávek zboží od dodavatelů
//  - pro databázové soubory používá vestavěné příkazy SQL
//  Spouští se rovnou
//*****
Ctl-Opt DECEDIT('0,');

//  Obrazovkový soubor
Dcl-F OBJW          WORKSTN
//              Seznam objednávek - hlavičky objednávek
//              SFILE(OBJWS0:I0)
//              Detaily objednávek
//              SFILE(OBJWS1:I1)
//              Ceny a názvy zboží
//              SFILE(OBJWS2:I2)
//              Seznam dodavatelů
//              SFILE(OBJWS3:I3);

//  Objednávky - hlavička - přejmenování proti souboru DODAV
//              inicializace je nutná kvůli datumu
Dcl-DS OBJHLA          extname('OBJHLA_T') INZ;
      CDODHLA          EXTFLD ('CDOD');
End-DS;

//  Objednávky - detail (zboží) - přejmenování proti souborům DODAV, OBJHLA, CENYD
Dcl-DS OBJDET          extname('OBJDET_T');
      CDODDET          extfld ('CDOD');
      COBJDET          extfld ('COBJ');
      CZBOZIDET        extfld ('CZBOZID');
End-DS;

//  Ceny objednávaného zboží - přejmenování proti souboru DODAV
Dcl-DS CENYD          extname('CENYD');
      CDODCEN          extfld ('CDOD');
End-DS;

//  Dodavatelé
Dcl-DS DODAV          extname('DODAV');
End-DS;

//  Pořadové číslo právě zpracovávaného záznamu v podsouborech
Dcl-S I0              Int(10:0);           // hlavičky
Dcl-S I1              Int(10:0);           // detaily
Dcl-S I2              Int(10:0);           // ceny
Dcl-S I3              Int(10:0);           // dodavatelé
//  Počet záznamů v jednotlivých podsouborech
Dcl-S Pocet0          Int(10:0);
Dcl-S Pocet1          Int(10:0);
Dcl-S Pocet2          Int(10:0);
Dcl-S Pocet3          Int(10:0);

//  Úložné proměnné
Dcl-S CZBOZ2          Like(CZBOZID);
Dcl-S MNOBJ2          Like(MNOBJ);
Dcl-S CDOD2          Like(CDOD);

//  Index pro cyklus
Dcl-S Idx              Packed(4:0);

//  Dnešní datum
Dcl-S Dnes              Date(*ISO);

Dcl-S DodavZmenen      Ind;
Dcl-S MnozZmeneno      Ind;

//  Maximální počet záznamů v podsouboru obecně
Dcl-C MAXIMUM_SF      9998;
```

```

// Velikost stránky a podsouboru 2 (zboží) - konst.
Dcl-S Pag2                      Int(10:0) INZ(7);

// Velikost stránky a podsouboru 3 (dodavatelé) - konst.
Dcl-S Pag3                      Int(10:0) inz(7);

// Číslo obrazovkových stránek
Dcl-S PosledniStr               Int(10:0);
Dcl-S SoucasnaStr              Int(10:0);

//=====
//   H L A V N Í   P R O G R A M
//=====

// Řazení znaků podle českých pravidel (pro názvy zboží a dodavatelů)
Exec SQL set option LANGID = CSY, SRTSEQ = *LANGIDSHR,
// Určit typ pro datum: ISO
//          DATFMT = *ISO,
// Zrušit sledování transakcí (aby byl umožněn UPDATE bez žurnálování)
//          COMMIT = *NONE ;

Dnes = %Date(); // dnešní datum
KUR0 = 1; // Nastavit kurzor pro seznam objednávek (podsoubor 0)

// Cyklus 1 - Zpracování objednávek
DoW 0 = 0;
  Exsr PlnitSF0; // Plnit podsoubor 0 seznamem objednávek

// Cyklus 2 - Zpracovat seznam objednávek
DoU 0 = 0;
  Write OBJWF0; // Zapsat návod na funkční klávesy na obrazovku bez čekání na vstup
  Exfmt OBJWC0; // Zapsat a číst podsoubor 0 (seznam objednávek) na obrazovce

  If *In03 Or *In12; // F3 nebo F12 - Končit program
    Exsr EndPgm;
  EndIf;

  If *In06; // F6 - Založit novou objednávku (pak pokračujeme jako u Enter)
    Exsr NovaHlaObj; // Nová hlavička objednávky
    If *In12; // F12 - Návrat na zobrazení seznamu objednávek bez zápisu
      Exsr PlnitSF0; // do databáze
      Iter;
    EndIf;
    Exsr ZprDetObj; // Zpracovat detaily nové objednávky
    Exsr ZprHlaObj; // Opakovat zpracování hlavičky objednávky
    // (po F12 v podprogramu ZprDetObj)
    Exsr PlnitSF0; // Opakovat zobrazení seznamu objednávek
    Iter;
  EndIf; // konec F6

  If *In23; // F23 - Zrušit objednávku včetně detailů
    // Vyhledám záznam podsouboru podle pořadového čísla
    // (dostanu klíč - číslo objednávky)
    Chain(E) SFKUR0 OBJWS0;
    *In80 = *Off;
    If Not %Found Or %Error; // Nenalezen/Chyba
      *In80 = *On;
      Iter; // Opakovat cyklus 2
    EndIf;

    // Zruším hlavičku, na niž ukazuje kurzor
    Exec SQL delete from OBJHLA where COBJ = :COBJ;

    // Zruším odpovídající detaily, jestliže nějaké existují
    Exec SQL delete from OBJDET where COBJ = :COBJ;

    // Nastavím kurzor na předchozí záznam podsouboru pro příští
    // zobrazení. Je-li současná poloha kurzoru (SFKUR0)
    // větší než 1, dosadím ji zmenšenou o 1 do KUR0 pro

```

```

// příští zobrazení. Jinak nechám KUR0 (rovno 1).
If SFKUR0 > 1;
    KUR0 = SFKUR0 - 1;
EndIf;

// Opakovat zpracování od začátku cyklu (s plněním podsouboru)
Exsr PlnitSF0;
Iter;
EndIf; // F23

// Enter:

// Nastavit kurzor na posledně zpracovávanou objednávku
// pro příští zobrazení (podle aktuální pozice kurzoru
// v podsouboru SFKUR0). Je-li chybná, nechat dosavadní.
If SFKUR0 >= 1 And SFKUR0 <= Pocet0;
    KUR0 = SFKUR0;
EndIf;

Chain(E) SFKUR0 OBJWS0; // Vyhledat záznam podsouboru podle poř. čísla
*In80 = *Off; // (dostaneme klíč - číslo objednávky)
If Not %Found Or %Error;
    *In80 = *On; // Nenalezen/Chyba - opak. cyklus 2
    Iter;
EndIf;

// Cyklus 3 - Zpracovat objednávku (hlavičku i detaily)
DoW 0 = 0;
    Exsr ZprHlaObj; // Zpracování hlavičky objednávky
    If *In12; // F12 po zpracování hlavičky - Zobrazit seznam objednávek
        Exsr PlnitSF0;
        Leave;
    EndIf;
    Exsr ZprDetObj; // Zpracovat detaily objednávky
EndDo; // (Cyklus 3 - konec)

// Znovu zpracovat seznam objednávek (bez nového plnění SF0)
EndDo; // (Cyklus 2 - konec)

EndDo; // (Cyklus 1 - konec)

//=====
// P O D P R O G R A M Y
//=====
//-----
// ZprHlaObj - Zpracovat hlavičku objednávky
//-----
BegSr ZprHlaObj;

// Cyklus - Zobrazit hlavičku objednávky
DoW 0 = 0;
    Exfmt OBJWF4;
    If *In03; // F3 - Konec programu
        Exsr EndPgm;
    EndIf;

    // F4 - Zpracovat dodavatele a návrat k zobrazení hlavičky obj.
    //     Jen pro novou objednávku (F6)
    If *In04 and *In06;
        // Zpracovat dodavatele (vyberu jednoho ze seznamu)
        Exsr ZprDodav;

        Iter; // znovu zobrazit
    EndIf;

    // F12 - Končit podprogram ZprHlaObj bez zápisu hlavičky objednávky
    If *In12;
        If MnozZmeneno or DodavZmenen;
            DTOBJ = %date(); // změním datum objednávky
            // Přepsat hlavičku (i když se nezměnil dodavatel)
            Exec SQL update OBJHLA

```

```

        set OBJHLA.CDOD = :CDOD, OBJHLA.DTOBJ = :DTOBJ
        where COBJ = :COBJ;

    EndIf;
    Leave;
EndIf;

// Enter:

// Prázdné č. dodavatele nebo nenalezeno - chyba
Exec SQL select NAZDOD into :NAZDOD
        from DODAV
        where CDOD = :CDOD;
*In84 = *Off;
If *In32 Or sqlstate = '02000';
    *In84 = *On;
    Iter;
EndIf;

// Jinak končit podprogram
Leave;
EndDo;

EndSr; // konec ZprHlaObj

//-----
// ZprDetObj - Zpracovat detaily objednávky
//-----

BegSr ZprDetObj;

    Exsr PlnitSF1; // Plnit podsoubor 1 - detaily objednávek

    // Cyklus 1 - Zobrazit okno se seznamem detailů objednávek
    DoW 0 = 0;
        Write OBJWF1;
        Exfmt OBJWC1;

        If *In03; // F3 - Konec programu
            Exsr EndPgm;
            EndIf;

        If *In12; // F12 - Návrat z podprogramu ZprDetObj
            Leave;
            EndIf;

        If *In05; // F5 - Znovu zobrazit nově naplněný seznam detailů objednávek
            Exsr PlnitSF1;
            Iter;
            EndIf;

        // F6 - Založit nový detail objednávky a znovu naplnit seznam
        //      detailů a zobrazit jej
        If *In06;
            Exsr NovyDetObj;
            Exsr PlnitSF1;
            Iter;
            EndIf;

        // F23 - Zrušit detail objednávky a znovu naplnit seznam
        //      (kurzor nastavit na záznam předcházející zrušenému)
        If *In23;
            // Vyhledat záznam podsouboru podle poř. čísla
            // (dostaneme klíč detailu objednávky)
            Chain SFKUR1 OBJWS1;
            If Not %Found; // Nenalezen - chyba
                *In81 = *On;
                Iter;
            Else;
                *In81 = *Off;
            EndIf;
            // Zrušit detail objednávky, na nějž ukazuje kurzor

```



```

// (podle klíče získaného z podsouboru)
Exec SQL delete from OBJDET
        where COBJ = :COBJ
        and CDOD = :CDOD
        and CZBOZID = :CZBOZID;
// a přepsat datum objednávky v hlavičce
Dnes = %date();
Exec SQL update OBJHLA set DTOBJ = :Dnes
        where COBJ = :COBJ;
Exsr PlnitSF1;
Iter;
EndIf;

// Enter - Zpracovat změněné detaily, není-li podsoubor prázdný
//      (aktualizovat změněné množství zboží v databázi)

If Pocet1 > 0;
    MnozZmeneno = *Off;    // množství zatím nezměněno
    MNOBJ2 = MNOBJ;        // uložím dosavadní množství

    ReadC OBJWS1;

    // Cyklus 2
    DoW Not %EOF;
        If MNOBJ <> MNOBJ2;    // změnilo se množství proti starému
            Exec SQL update OBJDET set MNOBJ = :MNOBJ
                    where COBJ = :COBJ and CZBOZID = :CZBOZID;
            MnozZmeneno = *on;  // označím změnu
            DTOBJ = %date();    // a dosadím dnešní datum pro objednávku
        EndIf;
        ReadC OBJWS1;
    EndDo; // (Cyklus 2 - konec)
    Exsr PlnitSF1; // Znovu naplnit podsoubor 1 - detaily objednávek

EndIf; // (Cyklus 1 - konec)

EndDo; // konec hlavního cyklu
EndSr; // konec ZprDetObj

//-----
// NovaHlaObj - Pořídít novou hlavičku objednávky
//-----

BegSr NovaHlaObj;

// Vyčistit data a dosadit dnešní datum
Clear OBJHLA; // vyčistit datovou strukturu hlavičky
Clear CDOD;   // a ještě původní pole dodavatele pro obrazovku
Clear NAZDOD; // vyčistit název dodavatele

// Cyklus - Zobrazit prázdnou hlavičku objednávky
DoW 0 = 0;
    *In33 = *On;
    Exfmt OBJWF4;

    If *In03; // F3 - Konec programu
        Exsr EndPgm;
    EndIf;

    If *In12; // F12 - Návrat
        Leave;
    EndIf;

    // F4 - Zpracovat dodavatele a znovu zobrazit hlavičku objednávky
    If *In04;
        Exsr ZprDodav;
        Iter;
    EndIf;

    // Enter:

```

```

// Přečíst dodavatele podle čísla z obrazovky
Exec SQL select CDOD into :CDOD
      from DODAV
      where CDOD = :CDOD;

*In84 = *Off;
If sqlstate = '02000'; // dodavatel nenalezen
  *In84 = *On;          // indikátor chyby
  Iter;                // znovu zobrazit okno
EndIf;

// Zkusit, zda záznam hlavičky existuje
Exec SQL select COBJ into :COBJ from OBJHLA
      where COBJ = :COBJ;

If sqlstate = '00000'; // když se našel
  *In85 = *on;          // indikace chyby
  Iter;                // znovu zobrazit okno
EndIf;
*In85 = *off;

// Zapsat nový záznam do hlavičky
DTOBJ = %date();
Exec SQL insert into OBJHLA ( COBJ, CDOD, DTOBJ )
      values ( :COBJ, :CDOD, :DTOBJ );
Leave;

EndDo; // (Cyklus - konec)

EndSr;

//-----
// NovyDetObj - Pořídít nový detail objednávky (zboží)
//-----

BegSr NovyDetObj;

// Vyčistit údaje pro obrazovku
Clear CZBOZID;
Clear MNOBJ;

// Cyklus - Zobrazit okno pro pořízení nového detailu objednávky
DoW 0 = 0;
  Exfmt OBJWF5;

  If *In03; // F3 - Konec programu
    Exsr EndPgm;
  EndIf;

  If *In12; // F12 - Návrat
    Leave;
  EndIf;

  If *In04; // F4 - Zobrazit a zpracovat seznam zboží
    Exsr ZprCeny; // Zobr. zboží
    Iter;        // a znovu detaily
  EndIf;

  // Enter:

  // Zkusím, zda zboží je v ceníku
  Exec SQL select CDOD, CZBOZID into :CDOD, :CZBOZID
        from CENYD
        where CDOD = :CDOD
              and CZBOZID = :CZBOZID;
  If sqlstate = '02000'; // nenalezeno v ceníku
    *In83 = *On;          // indikátor chyby
    Iter;                // znovu zobrazit okno
  EndIf;
  *In83 = *Off;

```

```

// Zkusím, zda je zboží už v detailech
Exec SQL select COBJ into :COBJ from OBJDET
        where COBJ = :COBJ and CZBOZID = :CZBOZID;
If sqlstate = '00000'; // nalezeno v detailech
    *In82 = *on;        // indikátor chyby
    Iter;              // znovu zobrazit okno
EndIf;
*In82 = *off;

// Zapsat zboží do detailů objednávky
Exec SQL insert into OBJDET ( COBJ, CDOD, CZBOZID, MNOBJ )
        values ( :COBJ, :CDOD, :CZBOZID, :MNOBJ );
// a přepsat datum objednávky v hlavičce
Dnes = %date();
Exec SQL update OBJJHLA set DTOBJ = :Dnes
        where COBJ = :COBJ;
Leave;

EndDo; // (Cyklus - konec)

EndSr;

//-----
// PlnitSF0 - Plnit podsoubor 0 - hlavičky objednávek
//-----

BegSr PlnitSF0;

// Vymazat podsoubor 0 - hlavičky objednávky
*In48 = *On;
Write OBJWF0;
Write OBJWC0;
*In48 = *Off;
// Vynulovat čítač záznamů podsouboru
I0 = 0;

// Přechíst celý soubor hlaviček se jmény dodavatelů
Exec SQL declare CS0 cursor for
        select COBJ, H.CDOD, DTOBJ, NAZDOD
        from OBJJHLA as H
        join DODAV as D on H.CDOD = D.CDOD
        order by COBJ asc
        fetch first 9998 rows only // MAXIMUM_SF je 9998
        for read only;
Exec SQL open CS0;

// Přechíst první záznam z databáze hlaviček objednávek
Exec SQL fetch from CS0 into :COBJ, :CDOD, :DTOBJ, :NAZDOD;

// Cyklus, dokud není konec databázového souboru
DoW sqlstate < '02000' And I0 < MAXIMUM_SF;

    I0 += 1; // Zvýšit čítač záznamů podsouboru

    Write OBJWS0; // Zapsat záznam do podsouboru podle čítače

    // Číst další záznam z databáze hlaviček objednávek
    Exec SQL fetch from CS0 into :COBJ, :CDOD, :DTOBJ, :NAZDOD;
EndDo;
Exec SQL close CS0;

// Pamatovat počet záznamů podsouboru. Zapnout ind. 47,
// je-li počet větší než 0. Umožní to zobrazit stránku
// na obrazovce.
Pocet0 = I0;
If Pocet0 > 0;
    *In47 = *On;
Else;
    *In47 = *Off;
EndIf;

```

```

EndSr;

//-----
// PlnitSF1 - Plnit podsoubor 1 s detaily objednávky
//-----

BegSr PlnitSF1;

    // Vymazat podsoubor 1 - detaily objednávek
    *In28 = *On;
    Write OBJWF1;
    Write OBJWC1;
    *In28 = *Off;
    I1 = 0 ; // Vynulovat čítač záznamů podsouboru

    // Přecházet záznamy detailů stejného čísla objednávky
    Exec SQL declare CS1 cursor for
        select C.CZBOZID, MNOBJ, NAZZBO
        from OBJDET D
        join CENYD C on D.CDOD = C.CDOD
        and D.CZBOZID = C.CZBOZID
        where D.COBJ = :COBJ
        order by D.COBJ, D.CZBOZID asc
        fetch first 9998 rows only; // MAXIMUM_SF je 9998
    Exec SQL open CS1;

    // Číst první záznam detailů ze stejné objednávky
    Exec SQL fetch from CS1 into :CZBOZID, :MNOBJ, :NAZZBO;

    // Cyklus, dokud jsou data a není vyčerpáno maximum řádků
    DoW sqlstate < '02000' And I1 < MAXIMUM_SF;

        I1 += 1; // Zvýšit čítač záznamů podsouboru

        Write OBJWS1; // Zapsat záznam do podsouboru podle čítače

        // Číst další záznam detailů ze stejné objednávky
        Exec SQL fetch from CS1 into :CZBOZID, :MNOBJ, :NAZZBO;
    EndDo;
    Exec SQL close CS1;

    // Pamatovat počet záznamů podsouboru. Zapnout ind. 27,
    // je-li počet větší než 0. Umožní to zobrazit stránku
    // na obrazovce.
    Pocet1 = I1;
    If Pocet1 > 0;
        *In27 = *On;
    Else;
        *In27 = *Off;
    EndIf;

EndSr;

//-----
// ZprDodav - Získat číslo dodavatele z přehledu dodavatelů
//-----

BegSr ZprDodav;

    // Uschovat číslo dodavatele pro příp. F12
    // Vymazat pole obrazovky pro zadání začátku názvu dodavatele
    Clear ZNAZD;
    // Nastavit kurzor na 1. záznam podsouboru
    KUR3 = 1;
    // Plnit podsoubor 3 - dodavatelé
    Exsr PlnitSF3;

    // Cyklus - Zpracovat číslo dodavatele
    DoW 0 = 0;
        // Zobrazit seznam dodavatelů
        Write OBJWF3;
        Exfmt OBJWC3;

```

```

DodavZmenen = *Off;

If *In03; // F3 - Konec programu
    Exsr EndPgm;
EndIf;

If *In12; // F12 - končit podprog. ZprDodav
    Leave;
EndIf;

If *In25; // Page up - Vrátit se o stránku zpět (nastavením kurzoru)
    Exsr PgUp3; // a zobrazit ji
    Iter;
EndIf;

If *In26; // Page down - Přečíst další záznamy z databáze pro další
    Exsr PlnitSF3D; // stránku a zobrazit ji
    Iter;
EndIf;

// Enter:

// Je-li kurzor mimo stránku ( < 1 a > Pocet3 ) - Znovu plnit
// podsoubor podle zadaného začátku názvu (ZNAZD)
If SFKUR3 < 1 Or SFKUR3 > Pocet3;
    Exsr PlnitSF3;
    Iter;
EndIf;

CDOD2 = CDOD;
// Vyhledat záznam podsouboru podle polohy kurzoru
// (dostaneme číslo dodavatele)
Chain SFKUR3 OBJWS3;
If CDOD <> CDOD2;
    DTOBJ = %date(); // Změním datum na dnešní
    DodavZmenen = *On; // poznamenám změnu dodavatele
EndIf;
Leave;

EndDo; // (Cyklus - konec)

EndSr;

//-----
// PlnitSF3 - Plnit podsoubor 3 s dodavatelem
//-----

BegSr PlnitSF3;

// Vymazat podsoubor 3 - dodavatelé
*In48 = *On;
Write OBJWF3;
Write OBJWC3;
*In48 = *Off;
// Vynulovat čítač záznamů podsouboru 3
I3 = 0;

// Přečíst záznamy dodavatelů podle zadaného textu v poli ZNAZD
// od stejného nebo nejbližší vyššího podle abecedy
Exec SQL declare CS3 scroll cursor for
    select CDOD, NAZDOD, ADRDOD from DODAV
    where NAZDOD >= :ZNAZD
    order by NAZDOD asc
    fetch first 7 rows only; // Pag3 = 7

// Zapsat do podsouboru další stránku zboží
Exec SQL open CS3;
Exec SQL fetch first from CS3 into :CDOD, :NAZDOD, :ADRDOD;

DoW sqlstate < '02000' and I3 < Pag3;
    I3 += 1; // Zvýšit čítač záznamů podsouboru

```

```

        Write OBJWS3; // Zapsat záznam do podsouboru podle čítače
        Exec SQL fetch next from CS3 into :CDOD, :NAZDOD, :ADRDOD;
    EndDo;
    Exec SQL close CS3;

    // Pamatovat počet záznamů podsouboru. Zapnout ind. 47,
    //   je-li počet větší než 0. Umožní to zobrazit stránku
    //   na obrazovce.
    Pocet3 = I3;
    If Pocet3 > 0;
        *In47 = *On;
    Else;
        *In47 = *Off;
    EndIf;

    KUR3 = 1;

EndSr;

//-----
// PgUp3 - Nastavit předchozí stránku po Page up u dodavatelů
//-----

BegSr PgUp3;

    // Je-li současná poloha kurzoru větší než velikost stránky,
    //   odečíst velikost stránky od nastavení kurzoru pro
    //   příští zobrazení (Kurzor3). Jinak nastavit číslo 1.
    If SFKUR3 > Pag3;
        Eval KUR3 -= Pag3;
    Else;
        Eval KUR3 = 1;
    EndIf;

EndSr;

//-----
// PlnitSF3D - Plnit podsoubor 3 s dodavatelem po Page down
//-----

BegSr PlnitSF3D;

    // Spočítat číslo poslední stránky podsouboru
    PosledniStr = %Div(I3: Pag3);
    If %Rem(I3: Pag3) = 0;
        Eval PosledniStr -= 1;
    EndIf;

    // Spočítat číslo právě zobrazené stránky
    SoucasnaStr = %Div(SFKUR3: Pag3);
    If %Rem(SFKUR3: Pag3) = 0;
        Eval SoucasnaStr -= 1;
    EndIf;

    // Když zobrazená stránka není poslední stránka podsouboru,
    //   (číslo zobrazené stránky je menší než číslo poslední str.)
    //   nastavit kurzor o jednu stránku dál
    If SoucasnaStr < PosledniStr;

        KUR3 = SFKUR3 + Pag3;

    Else; // Když to je poslední stránka - Přečíst další záznamy z DB

        Chain I3 OBJWS3; // Přečíst poslední záznam podsouboru podle čítače

        // Nastavit databázi na vyšší název dodavatele
        Exec SQL declare CS3D scroll cursor for
            select CDOD, NAZDOD, ADRDOD from DODAV
            where NAZDOD > :NAZDOD
            order by NAZDOD asc
            fetch first 7 rows only; // Page3 = 7

```

```

Exec SQL open CS3D;

// Číst první záznam dodavatelů
Exec SQL fetch first from CS3D
      into :CDOD, :NAZDOD, :ADRDOD;
Idx = I3 + Pag3; // Hranice cyklu - Čítač plus velikost stránky

// Cyklus, dokud není konec souboru dodavatelů nebo
// není dosažena hranice cyklu Idx

DoW sqlstate < '02000' And I3 < Idx; // do počtu dat a velikosti stránky
  I3 += 1; // Zvýšit čítač záznamů podsouboru
  Write OBJWS3; // Zapsat záznam do podsouboru podle čítače

  // Číst další záznam dodavatelů
  Exec SQL fetch next from CS3D
        into :CDOD, :NAZDOD, :ADRDOD;

EndDo;
Exec SQL close CS3D;

// Pamatovat počet záznamů podsouboru. Zapnout ind. 47,
// je-li počet větší než 0. Umožní to zobrazit stránku
// na obrazovce.
Pocet3 = I3;
If Pocet3 > 0;
  *In47 = *On;
Else;
  *In47 = *Off;
EndIf;

KUR3 = I3; // Nastavit kurzor na poslední záznam podsouboru

EndIf; // konec testu na poslední stránku

EndSr;

//-----
// ZprCeny - Získat číslo a název zboží ze souboru CENY
//-----

BegSr ZprCeny;

// Uschovat hodnoty z obrazovky (pro příp. F12)
CZBOZ2 = CZBOZID;
MNOBJ2 = MNOBJ ;
// Vymazat pole pro zadání začátku názvu zboží
Clear ZNAZZB ;

// Plnit podsoubor 2 - zboží
Exsr PlnitSF2;

// Cyklus 1:
DoW 0 = 0;
  // Zobrazit seznam zboží
  Write OBJWF2;
  Exfmt OBJWC2;

  If *In03; // F3 - Konec programu
    Exsr EndPgm;
  EndIf;

  // F12 - Vrátit uschované hodnoty a končit podprog. ZprCeny
  If *In12;
    CZBOZID = CZBOZ2;
    MNOBJ = MNOBJ2;
    Leave;
  EndIf;

  // Page up - Naplnit předchozí stránku předchozími záznamy
  //      databáze a zobrazit tuto stránku

```

```

If *In25;
    Exsr PlnitSF2U;
    Iter;
EndIf;

// Page down - Naplnit další stránku dalšími záznamy
//          databáze a zobrazit tuto stránku
If *In26;
    Exsr PlnitSF2D;
    Iter;
EndIf;

// Enter:
// Je-li kurzor mimo stránku ( < 1 a > Pocet2 ) - Znovu plnit
//   podsoubor podle zadaného začátku názvu (ZNAZZB)
If SFKUR2 < 1 Or SFKUR2 > Pocet2;
    Exsr PlnitSF2;
    Iter;
EndIf;

// Vyhledat záznam podsouboru podle polohy kurzoru
//   (dostaneme číslo zboží)
Chain SFKUR2 OBJWS2;
Leave;
EndDo; // konec Cyklu 1

EndSr;

//-----
// PlnitSF2 - Plnit podsoubor 2 se zbožím od zadaného začátku
//          názvu zboží (ZNAZZB) - jen jednu stránku
//-----

BegSr PlnitSF2;

    // Vymazat podsoubor 2 - zboží
    *in38 = *on;
    Write OBJWF2;
    Write OBJWC2;
    *in38 = *off;
    // Vynulovat čítač záznamů podsouboru 2
    I2 = 0;

    // Najít v databázi první název zboží podle začátečních písmen
    //   nebo nejbliže vyšší podle abecedy

    Exec SQL declare CS2 cursor for
        select CZBOZID, CENAJ, NAZZBO from CENYD
        where NAZZBO >= :ZNAZZB and CDOD = :CDOD
        order by NAZZBO asc
        fetch first 7 rows only;    // Page2 = 7

    // Zapsat do podsouboru další stránku zboží
    Exec SQL open CS2;
    Exec SQL fetch from CS2 into :CZBOZID, :CENAJ, :NAZZBO;

    DoW sqlstate < '02000' and I2 < Pag2;    // do počtu dat a velikosti stránky
        I2 += 1; // Zvýšit čítač záznamů podsouboru
        Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
        Exec SQL fetch from CS2 into :CZBOZID, :CENAJ, :NAZZBO;
    EndDo;
    Exec SQL close CS2;

    // Pamatovat počet záznamů podsouboru. Zapnout ind. 37,
    //   je-li počet větší než 0. Umožní to zobrazit stránku
    //   na obrazovce.
    Pocet2 = I2;
    If Pocet2 > 0;
        *In37 = *On;
    Else;
        *In37 = *Off;

```



```

EndIf;

EndSr;

//-----
// PlnitSF2U - Plnit podsoubor 2 se zbožím pozpátku od dosavadního
// prvního názvu zboží v podsouboru
//-----

BegSr PlnitSF2U;

// Najít první dosavadní název zboží v podsouboru
// (podsoubor má jen jednu stránku!)
// Když neexistuje, NAZZBO se nezmění.
Chain(E) 1 OBJWS2;

//.....
// ALTERNATIVA 1 (s opakováním řádku)
// Přečíst nejvýše 7 záznamů menších či rovných naposledy přečtenému názvu
Exec SQL declare CS2U scroll cursor for
      select CZBOZID, CENAJ, NAZZBO from CENYD
      where NAZZBO <= :NAZZBO // menší nebo rovno
      and CDOD = :CDOD
      order by NAZZBO desc // seřadit sestupně pro čtení "pozpátku"
      fetch first 7 rows only; // Pag2 = 7

//.....
// ALTERNATIVA 2 (bez opakování řádku)
// Přečíst nejvýše 7 záznamů menších než je text v poli NAZZBO
//.....

// Exec SQL declare CS2U scroll cursor for
//      select CZBOZID, CENAJ, NAZZBO from CENYD
//      where NAZZBO < :NAZZBO // ostře menší
//      order by NAZZBO desc // seřadit sestupně pro čtení "pozpátku"
//      fetch first 7 rows only; // Pag2 = 7

// Vymazat podsoubor 2 - zboží (má velikost jedné stránky)
*in38 = *on;
Write OBJWF2;
Write OBJWC2;
*in38 = *off;
// Vynulovat čítač záznamů podsouboru 2
I2 = 0;

Exec SQL open CS2U;
// Naplnit podsoubor záznamy z databáze od tohoto názvu dále
// Číst poslední záznam z výsledné tabulky
Exec SQL fetch last from CS2U into :CZBOZID, :CENAJ, :NAZZBO;

// Cyklus, dokud není konec databáze nebo není vyčerpán
// počet řádků obrazovkové stránky (Pag2)
DoW sqlstate < '02000' And I2 < Pag2;
    I2 += 1; // Zvýšit čítač záznamů podsouboru
    Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
    // Číst předchozí záznam výsledné tabulky
    Exec SQL fetch prior from CS2U into :CZBOZID, :CENAJ, :NAZZBO;
EndDo;
Exec SQL close CS2U;

// Pamatovat počet záznamů podsouboru. Zapnout ind. 37,
// je-li počet větší než 0. Umožní to zobrazit stránku
// na obrazovce.
Pocet2 = I2;
If Pocet2 > 0;
    *In37 = *On;
Else;
    *In37 = *Off;
EndIf;

```

```

EndSr;

//-----
// PlnitSF2D - Plnit podsoubor 2 se zbožím dopředu od dosavadního
//                posledního názvu zboží v podsouboru
//-----
BegSr PlnitSF2D;

    I2 = 0; // Vynulovat čítač záznamů podsouboru 2

    // Najít poslední dosavadní název zboží v podsouboru
    // (podsoubor má jen jednu stránku!)
    // Když neexistuje, zapne se ind. 86 a NAZZBO se nezmění.
    Chain(E) Pag2 OBJWS2;

    // Vymazat podsoubor 2 - zboží
    *in38 = *on;
    Write OBJWF2;
    Write OBJWC2;
    *in38 = *off;

    I2 = 0; // Vynulovat čítač záznamů podsouboru 2

    //.....
    // ALTERNATIVA 1 (s opakováním řádku)
    // Nastavit ukazatel do databáze PŘED tento název
    // (na příští stránce se opakuje poslední řádek z minulé str.)

    Exec SQL declare CS2D cursor for
        select CZBOZID, CENAJ, NAZZBO from CENYD
        where NAZZBO >= :NAZZBO and CDOD = :CDOD
        order by NAZZBO asc
        fetch first 7 rows only;    // Page2 = 7

    //.....
    // ALTERNATIVA 2 (bez opakování řádku)
    // Nastavit ukazatel do databáze ZA tento název
    // (na příští stránce se neopakuje žádný řádek z minulé str.)
    //.....

    // Exec SQL declare CS2D cursor for
    //     select CZBOZID, CENAJ, NAZZBO from CENYD
    //     where NAZZBO > :NAZZBO
    //     order by NAZZBO asc
    //     fetch first 7 rows only;    // Page2 = 7

    Exec SQL open CS2D;

    // Číst nastavený záznam z databáze
    Exec SQL fetch from CS2D into :CZBOZID, :CENAJ, :NAZZBO;

    // Cyklus, dokud není konec databáze nebo není vyčerpán
    // počet řádků obrazovkové stránky (Pag2)
    DoW sqlstate < '02000' And I2 < Pag2;    // do počtu data a velikosti stránky
        I2 += 1;    // Zvýšit čítač záznamů podsouboru
        Write OBJWS2; // Zapsat záznam do podsouboru podle čítače
        Exec SQL fetch from CS2D into :CZBOZID, :CENAJ, :NAZZBO;
    EndDo;
    Exec SQL close CS2D;

    // Pamatovat počet záznamů podsouboru. Zapnout ind. 37,
    // je-li počet větší než 0. Umožní to zobrazit stránku
    // na obrazovce.
    Pocet2 = I2;
    If Pocet2 > 0;
        *In37 = *On;
    Else;
        *In37 = *Off;
    EndIf;

EndSr;

```

```
//-----  
// EndPgm - Ukončit program  
//-----  
  
BegSr EndPgm;  
    *InLR = *On;  
    Return;  
EndSr;
```