# Použití SQL v RPG

Vladimír Župka, 2025

# Předkompilátor SQL pro jazyk RPG

*Zdrojový typ* **SQLRPGLE**

*Kompilace* **CRTSQLRPGI** – Create SQL ILE RPG Object

*Source member*

```
CRTSQLRPGI OBJ(PROG) SRCFILE(QRPGLESRC) SRCMBR(*OBJ)
```

*Stream file*

```
CRTSQLRPGI OBJ(PROG) SRCSTMF('/home/vzupka/qrpglesrc/prog.SQLRPGLE')
```

*Parametry překladu*

```
COMMIT(*CHG)                        *NONE *ALL ...
OBJTYPE(*PGM)                       *MODULE *SRVPGM
CLOSQLCSR(*ENDACTGRP)               *ENDMOD   kdy se uzavře kurzor
DFTRDBCOL(*NONE)                    předvolené SQL schema (default collection)
DYNDFTCOL(*NO)                      *YES - DFTRDBCOL také pro dynamické příkazy
OPTION(*SYS|*SQL                    jmenná konvence
       *JOB|*SYSVAL|*PERIOD|*COMMA
       ...)
TOSRCFILE(QTEMP/QSQLTEMP1)          kam je uložen předkompilovaný zdrojový text
DBGVIEW(*NONE)                      *SOURCE, *STMT, *LIST
...
```

# Zápis příkazů v RPG programu

*Ve volném formátu*

```
EXEC SQL příkaz ;
```

*V pevném formátu*

```
C/EXEC SQL příkaz
C/END-EXEC
```

*Alternativní a doplňkové volby pro SQL příkazy. Některé jsou shodné s parametry předkompilátoru.*

```
Exec SQL SET OPTION LANGID = CSY, SRTSEQ = *LANGIDSHR,
                    DATFMT = *ISO, COMMIT = *NONE ;

              COMMIT =     *CHG *NONE *ALL ...
              DATFMT =     *JOB *ISO ...
              DATSEP =     *JOB *PERIOD *COMMA *DASH *BLANK
              DFTRDBCOL =  *NONE
              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
              ...
```

# Soubory – tabulky

### *Soubor CENIKP – Ceník materiálu*

```
A                                          UNIQUE
A           R CENIKPF0
A             MATER          5             COLHDG('Číslo' 'mater.')
A             CENA         10P 2           COLHDG('Cena/j.')
A             NAZEV          30            COLHDG('Název zboží')
A           K MATER
```

### *Soubor STAVYP – Stavy materiálových zásob*

```
A                                          UNIQUE
A           R STAVYPF0
A             ZAVOD          2             COLHDG('Záv')
A             SKLAD          2             COLHDG('Skl')
A             MATER          5             COLHDG('Číslo' 'mater.')
A             MNOZ         10P 2           COLHDG('Množství')
A           K ZAVOD
A           K SKLAD
A           K MATER
```

### *Soubor OBRATP – Obraty materiálu*

```
A           R OBRATPF0
A             ZAVOD          2             COLHDG('Záv')
A             SKLAD          2             COLHDG('Skl')
A             MATER          5             COLHDG('Číslo' 'mater.')
A             MNOBR        10P 2           COLHDG('Množství obratu')
A           K ZAVOD
A           K SKLAD
A           K MATER
```

# Stavy a obraty – statické SQL příkazy SELECT, UPDATE

Program STAOB3_SQL

```
**free
Exec SQL set option COMMIT = *NONE;

Dcl-DS stavy  ExtName('*LIBL/STAVYP') End-DS; // host variables
Dcl-DS obraty ExtName('*LIBL/OBRATP') qualified End-DS; // odstraní duplicitu

Exec SQL declare CS cursor for
         select ZAVOD, SKLAD, MATER, MNOZ from STAVYP;
Exec SQL open CS;

Exec SQL fetch from CS into :stavy;
dow sqlstate < '02000';
    Exec SQL update STAVYP set MNOZ = MNOZ +
                ( select sum(MNOBR) from OBRATP
                  where ZAVOD = :ZAVOD
                    and SKLAD = :SKLAD
                    and MATER = :MATER
                  group by ZAVOD, SKLAD, MATER
                )
              where ZAVOD = :ZAVOD and SKLAD = :SKLAD and MATER = :MATER;
    Exec SQL fetch from CS into :stavy;
enddo;

Exec SQL close CS;
return;
```

# Chybové stavy

| SQLSTATE | | SQLCODE |
|---|---|---|
| 00000 | Operace byla **úspěšná**, bez varování nebo výjimky. | 0 |
| 01503 | Počet výsledných sloupců je větší než poskytnutý počet proměnných. | +000, +030 |
| 02000 | Nastala jedna z výjimek:<br>- Výsledek příkazu SELECT INTO nebo subselektu v příkazu INSERT je **prázdná** tabulka.<br>- Počet řádků určených v hledacím příkazu UPDATE nebo DELETE je **nula**.<br>- Pozice kurzoru v příkazu FETCH je za posledním řádkem výsledné tabulky.<br>- **Orientace** příkazu FETCH je nesprávná. | 100 |
| 07001 | **Počet** proměnných neodpovídá počtu parametrů (markerů) | -313 |
| 09000 | **Trigger** v SQL příkazu selhal. | -723 |
| 42703 | Zjištěn **nedefinovaný** sloupec nebo jméno parametru. | -205, -206,<br>-213, -5001 |

# Aktualizace příkazem MERGE

Program STAOBR_SQL

```
**free

Exec SQL set option COMMIT = *NONE;

Exec SQL MERGE INTO STAVYP S     // statický příkaz
   USING (select O.ZAVOD, O.SKLAD, O.MATER, sum(O.MNOBR) SUMA_OBRATU
          from OBRATP O
          group by O.ZAVOD, O.SKLAD, O.MATER
         ) as OBR
   ON ( S.ZAVOD = OBR.ZAVOD
    and S.SKLAD = OBR.SKLAD
    and S.MATER = OBR.MATER )
   when MATCHED then
        update set S.MNOZ = S.MNOZ + OBR.SUMA_OBRATU
-- when NOT MATCHED then
--      insert (ZAVOD, SKLAD, MATER, MNOZ)
--      values(OBR.ZAVOD, OBR.SKLAD, OBR.MATER, OBR.SUMA_OBRATU)
;

return;
```

# Dynamický SELECT s markery

Program DYNSEL_MK

```
**free
Dcl-S sel_stmt   char(500) inz;
Dcl-S spodni     packed(5: 2) inz( 2.5 );
Dcl-S horni      packed(5: 2) inz( 300 );
Dcl-DS *N  ExtName('CENIKP')  End-DS;

sel_stmt = 'select MATER, CENA, NAZEV from CENIKP +
            where CENA between ? and ? +
            order by CENA asc  +
            for read only' ;

Exec SQL PREPARE PREP from :sel_stmt ;
Exec SQL declare CUR cursor for PREP;
Exec SQL open CUR using :spodni, :horni;

Exec SQL fetch CUR into :MATER, :CENA, :NAZEV;
DoW sqlstate < '02000';
    snd-msg *info MATER + ' ' + %editc(CENA: 'K') + ' ' + NAZEV;
    Exec SQL fetch CUR into :MATER, :CENA, :NAZEV;
EndDo;

Exec SQL close CUR;
return;
```

# Dynamický příkaz – EXECUTE IMMEDIATE

Program DYNEX_IM

```
**free

Dcl-S DYNST       Char(500) Inz;
Dcl-S Limit       Packed(10: 2) Inz(500.00);

Exec SQL set option COMMIT = *NONE;

DYNST = 'update CENIKP set CENA = CENA * 1.10 +
                where CENA < ';
DYNST = %trim(DYNST) + ' ' + %char(Limit) ;

Exec SQL EXECUTE IMMEDIATE :DYNST;

return;
```

# Dynamický příkaz – PREPARE, EXECUTE, marker

Program DYNEX_MK

```
**free

Dcl-S DYNST      Char(500) Inz;

Dcl-PI *n;
   Limit Packed(10: 2);  // parametr z CMD příkazu
End-PI;

Exec SQL set option COMMIT = *NONE;

DYNST = 'update CENIKP set CENA = CENA * 1.10  +
                where CENA < ? ';

Exec SQL PREPARE STMT from :DYNST;
Exec SQL EXECUTE STMT using :Limit;

return;
```

CL příkaz k vyvolání programu

```
CMD         PROMPT('Volání SQL programu DYNEX_MK')
PARM        KWD(LIMIT) TYPE(*DEC) LEN(10 2) +
              DFT(250) PROMPT('Limit ceny:')
```

# Lokální SQL tabulka v podproceduře

Program LOC_SQL

```
**free
 // hlavní procedura volá podproceduru
ctl-OPT dftactgrp(*no);  // kvůli podproceduře v modulu

dsply %editc(VRATIT_CENU('00001'): 'P'); // volání podprocedury
return;
```

```
 // podprocedura vrací cenu pro číslo materiálu
dcl-PROC VRATIT_CENU export;

    dcl-DS cenik_ds extname('CENIKP') qualified end-DS;
    dcl-PI *N  packed(10: 2);    // rozhraní podprocedury
        material like(cenik_ds.MATER) CONST; // nebo VALUE
    end-PI;

    Exec SQL select CENA into :cenik_ds.CENA from CENIKP
             where MATER = :material;
    if sqlstate >= '02000';
        cenik_ds.CENA = -1;
    endif;
    return cenik_ds.CENA;

end-PROC;
```

# Dlouhá jména

***Vytvoření SQL tabulky a naplnění záznamy***

Program DL_JM1

```
**free

Exec SQL set option COMMIT = *NONE ;

Exec SQL CREATE OR REPLACE TABLE CENY_ZBOZI
         ( CISLO_ZBOZI      CHAR(5)      UNIQUE,
           CENA_ZA_JEDNOTKU DEC(12, 2),
           NAZEV_ZBOZI      CHAR(50) CCSID 870
         ) ;

Exec SQL INSERT INTO CENY_ZBOZI values ('00003', 1.25, 'Prádelní šňůra');
Exec SQL INSERT INTO CENY_ZBOZI values ('00004', 10.50, 'Ponožky pánské tmavé');
Exec SQL INSERT INTO CENY_ZBOZI values ('00005', 120.00, 'Tričko bílé');
Exec SQL INSERT INTO CENY_ZBOZI values ('00006', 10.55, 'Ponožky pánské bílé');
...

return;
```

***V protokolu o kompilaci najdeme odpovídající systémová jména***

První 4 znaky zůstávají, přidá se pořadové číslo.

```
...

CENA_ZA_JEDNOTKU                  ****      COLUMN
                                            8
CENA_ZA_JEDNOTKU                  6         COLUMN FOR CENA_00001 IN CENY_ZBOZI

CENA_00001                        6         DECIMAL(12,2) COLUMN IN CENY_ZBOZI

...
```

***Alternativně si můžeme volit vlastní systémová jména***

```
Exec SQL CREATE OR REPLACE TABLE CENY_ZBOZI FOR SYSTEM NAME  kratší-jméno
         ( CISLO_ZBOZI FOR COLUMN SYSTEM NAME  kratší-jméno  CHAR(5),
           CENA_ZA_JEDNOTKU FOR COLUMN SYSTEM NAME  kratší-jméno  DEC(12, 2),
           NAZEV_ZBOZI FOR COLUMN SYSTEM NAME  kratší-jméno  CHAR(50)
         );
```

***Výpis cen zboží***
Program DL_JM2

```
**free

Dcl-DS *N ExtName('CENY_ZBOZI') End-DS;  // proměnné – host variables

Exec SQL declare CS cursor for
         select CISLO_ZBOZI, CENA_ZA_JEDNOTKU, NAZEV_ZBOZI
         from   CENY_ZBOZI
         order by CISLO_ZBOZI ;

Exec SQL open CS;

Exec SQL fetch from CS into :CISLO00001, :CENA_00001, :NAZEV00001 ;

dow sqlstate < '02000';
    snd-msg CISLO00001 + ' ' + %char(CENA_00001) + NAZEV00001;
    Exec SQL fetch from CS into :CISLO00001, :CENA_00001, :NAZEV00001 ;
enddo;

Exec SQL close CS;

return;
```

# Datové typy RPG a SQL

| RPG | SQL |
|---|---|
| CHAR(n) | CHAR(n) |
| VARCHAR(n:2) | VARCHAR(n) |
| UCS2(n) | GRAPHIC(n) |
| VARUCS2(n:2) | VARGRAPHIC(n) |
| PACKED(n:m) | DECIMAL(n, m) |
| ZONED(n:m) | NUMERIC(n, m) |
| INT(5) | SMALLINT |
| INT(10) | INTEGER |
| INT(20) | BIGINT |
| BINDEC(1-4:0) | SMALLINT |
| BINDEC(5-9:0) | INTEGER |
| FLOAT(4) | FLOAT(24) \| REAL |
| FLOAT(8) | FLOAT(53) \| FLOAT |
| DATE(*DMY-) | DATE |
| TIME(*HMS.) | TIME |
| TIMESTAMP(n) | TIMESTAMP(n) |

***Neodpovídající typy***

```
dcl-S bin_1    SQLTYPE(BINARY: 50);       // CHAR(50) CCSID(*HEX);
dcl-S varbin_1 SQLTYPE(VARBINARY: 100);   // VARCHAR(100) CCSID(*HEX);


dcl-S FIELD_1  SQLTYPE (CLOB: 1000 );     // DCL-DS FIELD_1;
                                          //    FIELD_1_LEN  UNS(10);
                                          //    FIELD_1_DATA CHAR(1000);
                                          // END-DS FIELD_1;


Exec SQL set :FIELD_1 = 'ABCD';   // příkaz SQL v RPG
FIELD_1_DATA = 'ABCD';            // příkaz v RPG
```

# Trigger – zvýšení ceny nejvýše o 10% při aktualizaci záznamu

***Trigger externí – RPG program***

Program TG_CENA_U

```
**free

Dcl-PI *n;     // vstupní parametry z databázového systému
   Buf         LikeDS(TrgBuffer);
   Len         Uns(10);  // nepotřebuji
End-PI;

   // Trigger buffer (1. parametr) - obsahuje data nového záznamu
Dcl-DS TrgBuffer           Len(1000);
   ...                     ...
   OldRecOffset            Uns(10)  pos(49);
   OldRecLen               Uns(10);
   NewRecOffset            Uns(10)  pos(65);
   NewRecLen               Uns(10);
End-DS;

//   Starý záznam (before update)
Dcl-DS OldRecord  ExtName('CENIKP') Based(OldRecPtr) Qualified End-DS;
//   Nový záznam (before insertion)
Dcl-DS NewRecord  ExtName('CENIKP') Based(NewRecPtr) Qualified End-DS;

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

```
OldRecPtr = %Addr(Buf) + Buf.OldRecOffset;  // Adresa Starého záznamu
NewRecPtr = %Addr(Buf) + Buf.NewRecOffset;  // Adresa Nového záznamu

If (NewRecord.CENA > 100 and NewRecord.CENA > OldRecord.CENA * 1.10) ;
    NewRecord.CENA = OldRecord.CENA * 1.10 ;
EndIf;

Return;
```

***Trigger externí – registrace***

```
AddPfTrg FILE(CENIKP) TRGTIME(*BEFORE) TRGEVENT(*UPDATE) PGM(TG_CENA_U)
         RPLTRG(*YES) TRG(TG_CENA_U) TRGLIB(*FILE) ALWREPCHG(*YES)

RmvPfTrg ...
```

# Uložená procedura – hodnota všeho materiálu ve skladě

***Procedura externí – definiční skript***

Skript PR_CEN_E

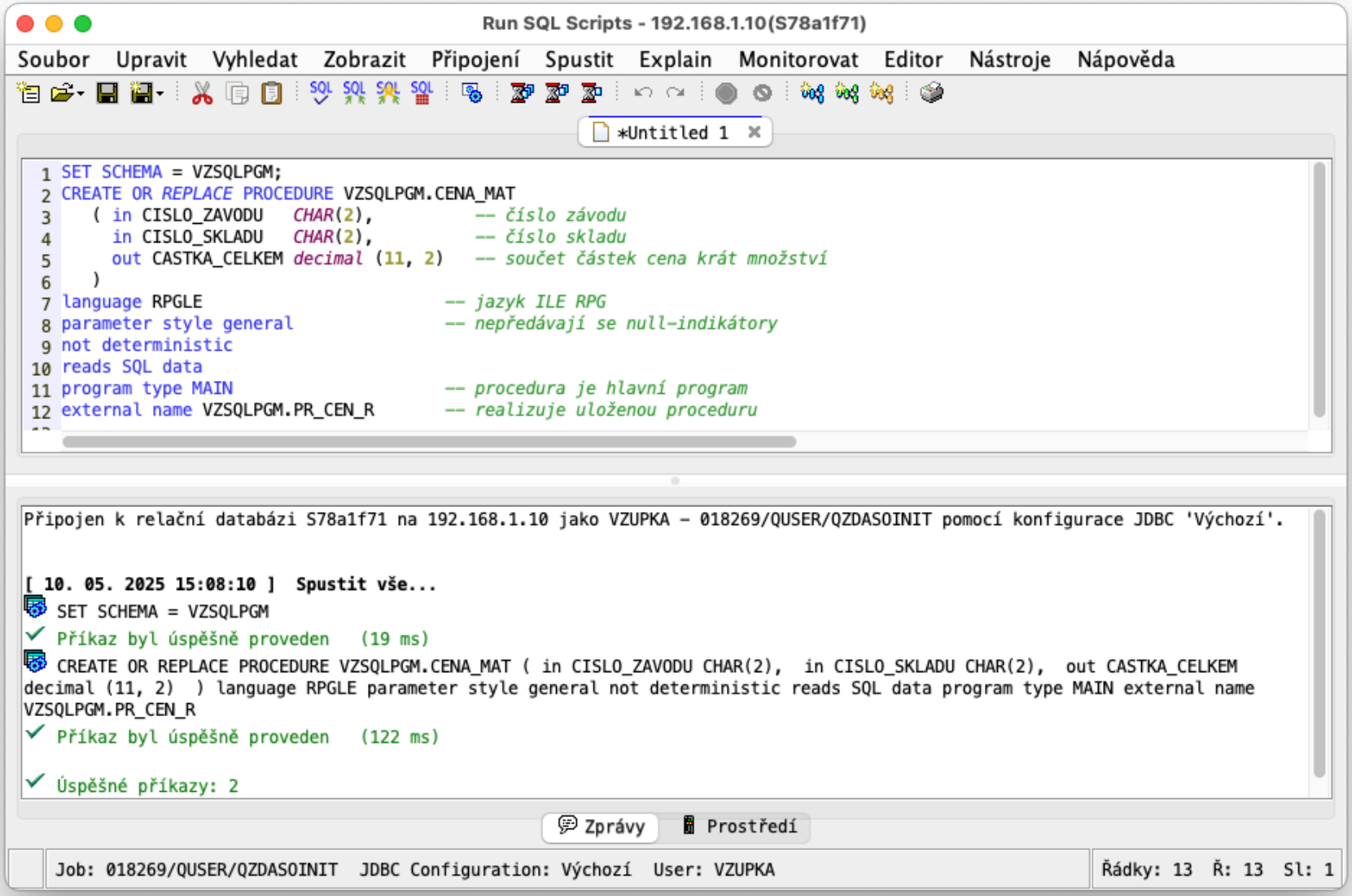
```
SET SCHEMA = VZSQLPGM;
CREATE OR REPLACE PROCEDURE VZSQLPGM.CENA_MAT
   ( in CISLO_ZAVODU   CHAR(2),          -- číslo závodu
     in CISLO_SKLADU   CHAR(2),          -- číslo skladu
     out CASTKA_CELKEM decimal (11, 2)   -- součet částek cena krát množství
   )
language RPGLE                           -- 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.PR_CEN_R          -- realizuje uloženou proceduru
```

Umístíme skript do zdrojového členu PR_CEN_E a spustíme CL příkazem

```
RUNSQLSTM SRCFILE(QSQLSRC) SRCMBR(PR_CEN_E)
```

nebo

V aplikaci ***IBM i Access Client Solutions*** vložíme skript do okna ***Run SQL Scripts*** a spustíme.



## *Procedura externí – RPG program*

Program PR_CEN_R

```
**free

   // Parametry uložené procedury
Dcl-PI *n;
    Zavod                  Char(2);     // in
    Sklad                  Char(2);     // in
    Suma                 Packed(11:2);  // out
End-PI;

Exec SQL select sum(CENA * MNOZ) into :Suma    -- out
      from STAVYP  as S
      join CENIKP  as C on C.MATER = S.MATER
    where ZAVOD = :Zavod and SKLAD = :Sklad ;

Return;
```

## *Volání procedury v programu*

Program PR_CEN_C

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

  Dcl-PI *n; // vstupní parametry uložené procedury
      Zavod                Char(2);
      Sklad                Char(2);
  End-PI;
  Dcl-S Suma               Packed(11:2); // výstupní parametr

  Exec SQL  CALL  CENA_MAT ( :Zavod, :Sklad, :Suma );

  Except DETAIL;   // tisk výsledků
  *inlr = *on;     // uzavře soubor QPRINT

OQPRINT    E            DETAIL         1
O                                          'Celková cena materiálu '
OQPRINT    E            DETAIL         1
O                                          'Závod: '
O                       Zavod               +1
OQPRINT    E            DETAIL         1
O                                          'Sklad: '
O                       Sklad               +1
OQPRINT    E            DETAIL         1
O                                          'Celkem:'
O                       Suma          P     +1
```

CL příkaz k vyvolání programu PR_CEN_C

```
CMD         PROMPT('Volání SQL programu PR_CEN_C')
PARM        KWD(ZAVOD) TYPE(*CHAR) LEN(2) +
              DFT('01') +
              PROMPT('Číslo závodu:')
PARM        KWD(SKLAD) TYPE(*CHAR) LEN(2) +
              DFT('01') +
              PROMPT('Číslo skladu:')
```

Výsledný tisk

```
Celková cena materiálu
Závod:  01
Sklad:  01
Celkem:      16966.62
```

# User defined function (UDF) – vrací jednotkovou cenu

***Funkce externí – definiční skript***

Skript FU_CEN_E

```
SET SCHEMA = VZSQLPGM;
CREATE OR REPLACE FUNCTION VZSQLPGM.VRAT_CENU_MATERIALU (MATERIAL CHAR(5))
   RETURNS  DECIMAL (10, 2)
   LANGUAGE  RPGLE
   PARAMETER STYLE  GENERAL
   NOT DETERMINISTIC  NO SQL
   PROGRAM TYPE  SUB              -- funkci realizuje podprocedura
   NOT FENCED                     -- není vázána na stejnou úlohu
   NO FINAL CALL                  -- není první a poslední volání
   EXTERNAL NAME VZSQLPGM.FU_CEN_R(FU_CEN_R)  -- sevisní program a podprocedura
```

Umístíme skript do zdrojového členu FU_CEN_E a spustíme CL příkazem

```
RUNSQLSTM SRCFILE(QSQLSRC) SRCMBR(FU_CEN_E)
```

## *Funkce externí – RPG podprocedura*

Program FU_CEN_R

```
**free
ctl-opt nomain;
dcl-PROC FU_CEN_R  export;
    dcl-F CENIKP keyed;  // příp. STATIC - nechá soubor otevřený
    dcl-DS cenik_ds  likerec(CENIKPF0); // je nutná datová struktura

    dcl-PI *N  packed(10: 2);    // rozhraní podprocedury
        material like(cenik_ds.MATER) CONST; // nebo VALUE
    end-PI;

    chain(e)  material  CENIKP  cenik_ds;  // čte do datové struktury
    if not %found();
        cenik_ds.CENA = -1;
    endif;
    return cenik_ds.CENA;
end-PROC FU_CEN_R;
```

- Lokální soubor negeneruje popisy I a O s proměnnými.
- Pro datová pole je nutné použít datovou strukturu nebo funkci %fields u UPDATE.
- Soubor se otevře vždy při vstupu do podprocedury. Uzavírá se při výstupu z podprocedury a proměnné zanikají.
- Klíčové slovo STATIC u souboru nechá soubor otevřený a zachová jeho data pro příští volání.

## *Program s voláním funkce*

Program FU_CEN_C

```
**free

Dcl-DS cenik_ds extname('CENIKP') qualified End-DS;
Dcl-S  cena_materialu Like(cenik_ds.CENA) Inz;

Dcl-PI *N;
    material Like(cenik_ds.MATER);  // vstupní parametr
End-PI;

// volám SQL funkci
Exec SQL  set :cena_materialu = VRAT_CENU_MATERIALU ( :material );

Dsply cena_materialu;

Return;
```

Vytvořím **modul** pro servisní program

```
CRTRPGMOD OBJ(FU_CEN_R) SRCFILE(QRPGLESRC)
```

Vytvořím **spojovací text** (binder source) v souboru QSRVSRC pro servisní program

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

Vytvořím **servisní program**

```
CRTSRVPGM SRVPGM(FU_CEN_R) MODULE(*SRVPGM) SRCFILE(QSRVSRC) SRCMBR(*SRVPGM)
```

---

Vytvořím **modul** volajícího programu

```
CRTSQLRPGI OBJ(FU_CEN_C) SRCFILE(QRPGLESRC) SRCMBR(*OBJ) OBJTYPE(*MODULE)
```

Vytvořím **volající program** připojením servisního programu k modulu

```
CRTPGM PGM(FU_CEN_C) MODULE(*PGM) BNDSRVPGM((FU_CEN_R))
```

---

**Spustím** volající program testující UDTF funkcl

```
CALL PGM(FU_CEN_C) PARM('00001')
```

# User defined table function (UDTF) – vrací tabulku

### *UDTF externí – RPG podprocedura*

Vytvoříme definiční **skript** pro externí uživatelskou funkci. Funkce OBJDAT_F přijímá dva parametry určující rozmezí datumů. Vybere objednávky z tabulky OBJHLA_T v daném rozmezí a ***vytvoří tabulku*** s čísly objednávek a s datumy. 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 RPGLE
   PARAMETER STYLE DB2SQL        -- umožňuje open, fetch, close
   NOT DETERMINISTIC
   READS SQL DATA
   SCRATCHPAD                    -- průběžná paměť
   PROGRAM TYPE SUB              -- funkce je ILE podprocedura
   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) -- serv. program a podprocedura
```

Skript umístíme do zdrojového členu, např. OBJDAT_FE a spustíme CL příkazem

```
RUNSQLSTM SRCFILE(QSQLSRC) SRCMBR(OBJDAT_FE)
```

## *UDTF externí – vytvoření funkce*

Vytvořím zdrojový **modul** OBJDAT_F s procedurou (funkcí) OBJDAT_F_1. (Podprocedura nesmí mít stejné jméno jako servisní program.)

```
**free
Ctl-Opt nomain;
Dcl-Proc OBJDAT_F   Export;
   Dcl-DS OBJHLA_T  Ext   Template ;  // host variables z tabulky OBJHLA_T
   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;

   //  Procedure interface
   Dcl-PI *N;                                  // parametry
      Datum1                     Date;         // in
      Datum2                     Date;         // in
      COBJ_PAR                Like(COBJ);         // out
      DTOBJ_PAR               Like(DTOBJ);        // out
      Datum1_ind                 Int(5:0);     // in
      Datum2_ind                 Int(5:0);     // in
      COBJ_PAR_ind               Int(5:0);        // out
      DTOBJ_PAR_ind              Int(5:0);        // out
      SQLSTATE_PAR              Char(5);          // out
      Func_name              VarChar(517);     // in
      Spec_name              VarChar(128);     // in
      Message_text           VarChar(1000);       // out
      Scratchpad             LikeDS(ScratchDS); // inout
      CallType                   Int(10:0);    // in
   End-PI;
```

```
   If Calltype = -1;       // 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;
   ElseIf Calltype = 0;    // 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 ;
   ElseIf Calltype = 1;    // CLOSE
      Exec SQL close CUR;
   EndIf;

End-Proc OBJDAT_F;
```

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

Přijímá dva parametry a vrací tabulku obsahující čísla a data objednávek v rozmezí parametrů.

Vytvořím **spojovací text** (binder source) v souboru QSRVSRC **pro servisní program**

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

Vytvořím **servisní program** OBJDAT_F s procedurou (funkcí) OBJDAT_F_1

```
CRTSQLRPGI OBJ(VZSQLPGM/OBJDAT_F) SRCFILE(VZSQLPGM/QRPGLESRC) SRCMBR(OBJDAT_F)
           OBJTYPE(*SRVPGM) CLOSQLCSR(*ENDACTGRP)
```

Vytvořím ***program*** OBJDAT_P s připojeným servisním programem OBJDAT_F

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

## *Program s voláním funkce*

Program OBJDAT_P

```
**free

Dcl-PI *n;   // vstupní parametry
   Dcl-S Datum1 Date;
   Dcl-S Datum2 Date;
End-PI;

Dcl-S COBJ   Char(6);  // host variables pro sloupce tabulky
Dcl-S DTOBJ  Date;

Exec SQL declare CUR cursor for
    select * from  TABLE ( OBJDAT_F(:Datum1 , :Datum2 ) ) as DAT_F;

Exec SQL open CUR ;

Exec SQL fetch CUR into :COBJ, :DTOBJ ;
DoW sqlstate = '00000';
   SND-MSG COBJ + ' ' + %char(DTOBJ);  // výpis do joblogu
   Exec SQL fetch CUR into :COBJ, :DTOBJ ;
EndDo;

Exec SQL close CUR;

Return;
```

# Statické a dynamické příkazy

Statické příkazy jsou zapsány přímo v příkazu EXEC SQL.
Dynamické příkazy jsou zapsány v textové proměnné.

### *Postup příkazů pro SELECT*

```
DECLARE SCROLL kurzor CURSOR FOR SELECT ...
OPEN kurzor
FETCH FIRST FROM kurzor INTO :proměnná, ...   (NEXT, LAST, PRIOR, ...)
CLOSE kurzor
```

### *Postup příkazů pro dynamický SELECT bez parametrů (markerů)*

```
text-příkazu = 'SELECT ... '
PREPARE připravený-příkaz FROM :text-příkazu
DECLARE SCROLL kurzor FOR připravený-příkaz
... jako výše
```

### *Postup příkazů pro dynamický SELECT s parametry (markery)*

```
text-příkazu = 'SELECT ... WHERE xyz BETWEEN ? AND ? ...'
PREPARE připravený-příkaz FROM :text-příkazu
DECLARE SCROLL kurzor FOR připravený-příkaz
OPEN kurzor USING :proměnná1, :proměnná2
... jako výše
```

### *Postup příkazů pro ostatní dynamické příkazy bez parametrů (markerů)*

```
text-příkazu = 'UPDATE ... SET ... '
EXECUTE IMMEDIATE :text-příkazu
```

### *Postup příkazů pro ostatní dynamické příkazy s parametry (markery)*

```
text-příkazu = 'UPDATE ... SET ... WHERE xyz BETWEEN ? AND ?'
PREPARE připravený-příkaz FROM :text-příkazu
EXECUTE připravený-příkaz USING :proměnná1, :proměnná2
```

# Interní SQL rutiny

### *Trigger interní – výkonný skript*

Skript TG_CENA_U

```
-- Trigger BEFORE UPDATE pro tabulku CENIKP
CREATE OR REPLACE TRIGGER TG_CENA_U BEFORE UPDATE ON CENIKP
   REFERENCING OLD ROW AS old_row
               NEW ROW AS new_row
   FOR EACH ROW                        -- spustí se u každého řádku
   MODE DB2ROW                         -- předepsáno pro BEFORE
   WHEN (new_row.CENAJ > 100 and
         new_row.CENAJ > old_row.CENAJ * 1.10 )
   BEGIN
      SET new_row.CENAJ = old_row.CENAJ * 1.10;
      SIGNAL SQLSTATE VALUE '01H01' SET MESSAGE_TEXT =
             'Navýšení množství přesahuje 10 %. Ponechá se navýšení 10 %';
   END ;
```

Umístíme skript do zdrojového členu TG_CENA_U a spustíme CL příkazem

```
RUNSQLSTM SRCFILE(QSQLSRC) SRCMBR(TG_CENA_U)
```

### *Procedura interní – výkonný skript*

Skript PR_CEN (v jazyku PL/SQL)

```
set schema = VZRPG_FREE;
create or replace procedure VZSQLPGM.CENA_MAT
   ( in CISLO_ZAVODU   CHAR(2),            -- číslo závodu
     in CISLO_SKLADU   CHAR(2),            -- číslo skladu
     out CASTKA_CELKEM decimal (11, 2)     -- Součet částek cena krát množství
   )
language SQL
reads sql data
begin
   -- zjištění ceny materiálů v daném závodu a skladu
   select sum(CENA * MNOZ) into CASTKA_CELKEM
      from STAVYP as S
      join CENIKP as C on C.MATER = S.MATER
   where SKLAD = CISLO_SKLADU and ZAVOD = CISLO_ZAVODU ;
end
```

Umístíme skript do zdrojového členu PR_CEN a spustíme CL příkazem

```
RUNSQLSTM SRCFILE(QSQLSRC) SRCMBR(PR_CEN)
```

nebo

v aplikaci ***IBM i Access Client Solutions*** vložíme skript do okna ***Run SQL Scripts*** a spustíme.

## *UDTF interní – výkonný skript*

PL/SQL skript OBJDAT_F

Skript generuje uživatelskou funkci OBJDAT_F, která přijímá dva parametry a ***vrátí tabulku*** se dvěma sloupci, COBJ (číslo objednávky) a DTOBJ (datum objednávky). Objednávky vybere z tabulky OBJHLA_T v daném rozmezí.

```
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
```

Skript umístíme do zdrojového členu, např. OBJDAT_F a spustíme CL příkazem

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

Výsledkem skriptu je **servisní program** OBJDA00001 v zadané knihovně. Ten je nutné spojit s modulem OBJDAT_P, aby vznikl **program OBJDAT_P**:

```
CRTPGM PGM(VZSQLPGM/OBJDAT_P) MODULE(*PGM) BNDSRVPGM((VZSQLPGM/OBJDA00001))
```