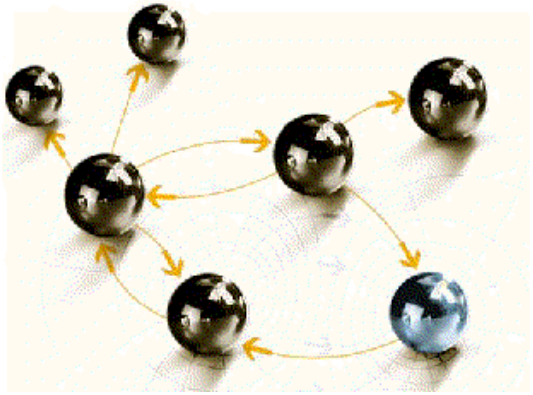


cps4it

2 Highlights aus SQL

ROWSET
FIRST FIRST ROW

cps4it
consulting, projektmanagement und seminare für die informationstechnologie
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen
Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de
Internet: http://www.cps4it.de

Eine Vorstellung im Hause

im

März 2009

Inhalt

- SQL-Code – ROWSET
- SQL-Code – SELECT ... FETCH FIRST ROW

SQL-Code – ROWSET

cps4it

Was ist ROWSET-Verarbeitung?

- statt 1 Zeile werden n Zeilen an DB2 übergeben
- unterschiedliche Begriffe
  - ROWSET-Verarbeitung
  - multi-row-Verarbeitung / multiple-row
- benutzbar bei
  - Cursorverarbeitung (multi-row FETCH)
  - insert (multi-row INSERT)
  - merge (multi-row MERGE)
- gibt es übrigens nicht erst seit DB2-V9

SQL-Code – ROWSET

cps4it

Warum ROWSET-Verarbeitung?

- klarer Performancevorteil
- jeder SQL heißt: Adressraumwechsel

SQL-Code – ROWSET

cps4it

Testkonstellation – nur mit FETCH

- DB2-Tabelle aus MFF (Flotte)
  - SYSIBM klappte nicht ☹
- Testprogramme in PL1 und COBOL
- Anzahl zu lesender Zeilen via JCL übergeben
- Anzahl Loops via JCL übergeben

SQL-Code – ROWSET

Cursor-Declare – ohne ROWSET

```
/* Declare Cursor "normal"          Index-Only                    */
     EXEC SQL
     DECLARE C_N_I CURSOR                          FOR
     SELECT
            FK_POL_NR
          , FK_OFFERT_VAR
          , DN_ENTITY_ID_ATTTY
          , GUELTIG_AB
          , VERSION
          , GUELTIG_BIS
          , TYP_WERT_DATUM
     FROM  UFLO_UTEXAZU
     FOR FETCH ONLY
     WITH UR
     ;
```

SQL-Code – ROWSET

Cursor-Declare – mit ROWSET

```
/* Declare Cursor "multi-row-fetch" Index-Only                    */
     EXEC SQL
     DECLARE C_M_I CURSOR WITH ROWSET POSITIONING FOR
     SELECT
            FK_POL_NR
          , FK_OFFERT_VAR
          , DN_ENTITY_ID_ATTTY
          , GUELTIG_AB
          , VERSION
          , GUELTIG_BIS
          , TYP_WERT_DATUM
     FROM  UFLO_UTEXAZU
     FOR FETCH ONLY
     WITH UR
     ;
```

SQL-Code – ROWSET

Cursor-Fetch – ohne ROWSET

```
EXEC SQL
FETCH                C_N_I

INTO
      :DB-FELDER-N.FK-POL-NR
    , :DB-FELDER-N.FK-OFFERT-VAR
    , :DB-FELDER-N.DN-ENTITY-ID-ATTTY
    , :DB-FELDER-N.GUELTIG-AB
    , :DB-FELDER-N.VERSION
    , :DB-FELDER-N.GUELTIG-BIS
    , :DB-FELDER-N.TYP-WERT-DATUM
      :DB-FELDER-N.TYP-WERT-DATUM-IND
;
```

SQL-Code – ROWSET

Cursor-Fetch – mit ROWSET

```
EXEC SQL
FETCH NEXT ROWSET C_M_I
FOR :I-FETCH-MAX ROWS
INTO
      :DB-FELDER-M.FK-POL-NR
    , :DB-FELDER-M.FK-OFFERT-VAR
    , :DB-FELDER-M.DN-ENTITY-ID-ATTTY
    , :DB-FELDER-M.GUELTIG-AB
    , :DB-FELDER-M.VERSION
    , :DB-FELDER-M.GUELTIG-BIS
    , :DB-FELDER-M.TYP-WERT-DATUM
      :DB-FELDER-M.TYP-WERT-DATUM-IND
;
```

SQL-Code – ROWSET

cps4it

Felddefinitionen – ohne ROWSET

```
DCL 01 DB_FELDER_N,
        05 EX_ATTRIBUT_ZUT_ID          FIXED BIN(31),
        05 FK_POL_NR                   FIXED BIN(31),
        05 FK_OFFERT_VAR               FIXED BIN(15),
        05 DN_ENTITY_ID_ATTTY          FIXED BIN(31),
        05 GUELTIG_AB                  CHAR(10),
        05 VERSION                     FIXED BIN(15),
        05 GUELTIG_BIS                 CHAR(10),
        05 TYP_WERT_DATUM              CHAR(10),
        05 TYP_WERT_DATUM_IND          FIXED BIN(15);
```

SQL-Code – ROWSET

Felddefinitionen – mit ROWSET

```
DCL 01 DB_FELDER_M,
        05 EX_ATTRIBUT_ZUT_ID   (1000) FIXED BIN(31),
        05 FK_POL_NR            (1000) FIXED BIN(31),
        05 FK_OFFERT_VAR        (1000) FIXED BIN(15),
        05 DN_ENTITY_ID_ATTTY   (1000) FIXED BIN(31),
        05 GUELTIG_AB           (1000) CHAR(10)     ,
        05 VERSION              (1000) FIXED BIN(15),
        05 GUELTIG_BIS          (1000) CHAR(10)     ,
        05 TYP_WERT_DATUM       (1000) CHAR(10)     ,
        05 TYP_WERT_DATUM_IND   (1000) FIXED BIN(15);
```

SQL-Code – ROWSET

cps4it

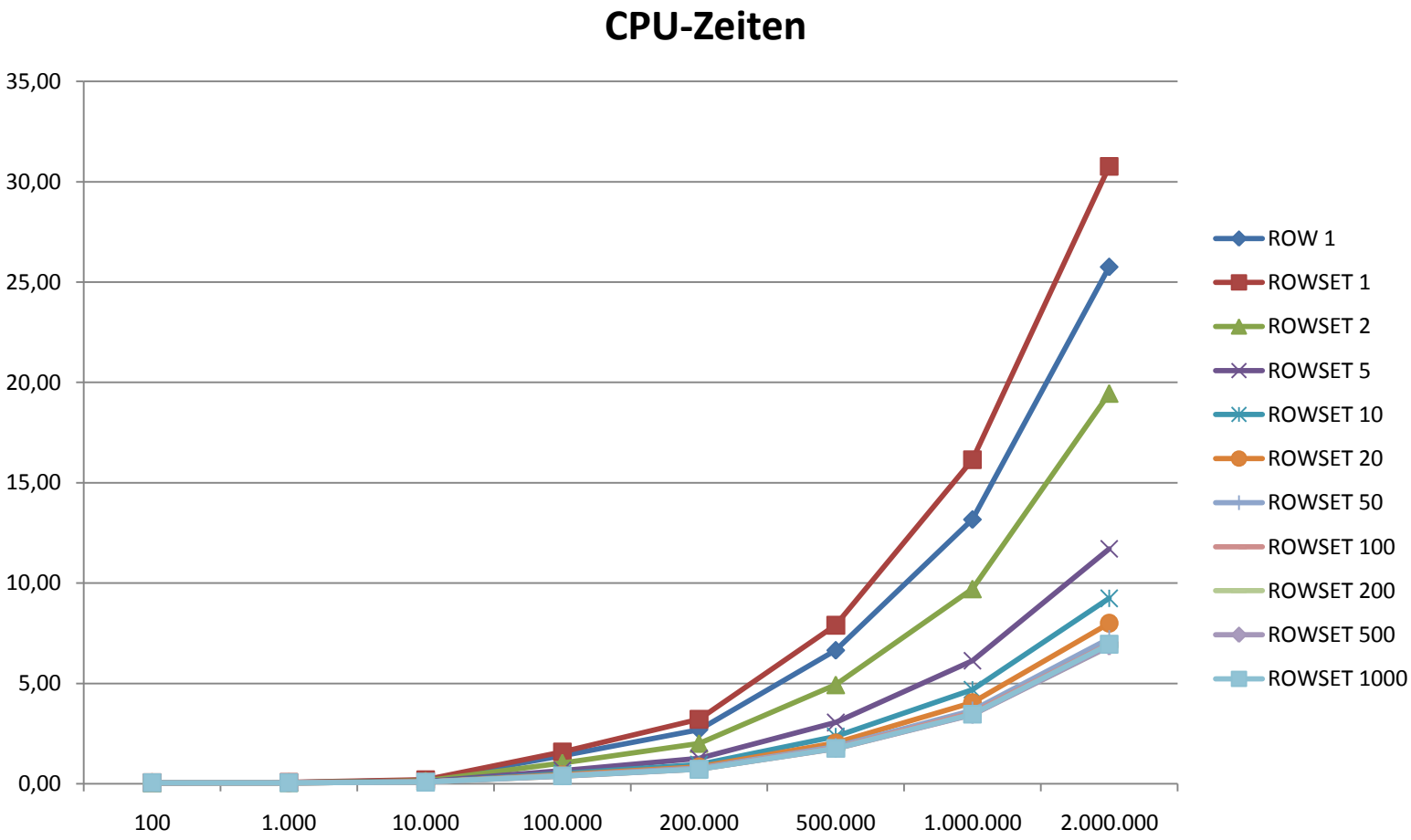
Testvorgehensweise

- Test mit „normalem“ Fetch
- Test mit multiple-row-Fetch 1 Zeile
- Test mit multiple-row-Fetch 2 Zeilen
- Test mit multiple-row-Fetch 5 Zeilen
- ... 10, 20, 50, 100, 200, 500, 1000
- unterschiedliche Anzahl Zugriffe
- Loop über die Zugriffe

Link

SQL-Code – ROWSET

Ergebnisse



SQL-Code – ROWSET

cps4it

Fazit

- ROWSET-Verarbeitung sichtbar effektiver,
- aber nur, wenn mindestens 2 Zeilen gelsen werden.
- Es lohnt sich, darüber nachzudenken, dass und wann mit ROWSET gearbeitet werden sollte.

SQL-Code – ROWSET

cps4it

Programme

- vollständige Testprogramme stehen unter SDE
- Kopien (auch JCL unter X016291.#.DB2V9.SRC)
- PL1-Version: U442RLF
- COBOL-Version: U441RLF

Link PL1-Programm

Link COBOL-Programm

Link JCL

Inhalt

- SQL-Code – ROWSET
- SQL-Code – SELECT ... FETCH FIRST ROW

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Was ist „FETCH FIRST ROW“ und Warum braucht man das?

- „normaler“ SELECT liest immer nach
- Falls weitere Zeile vorhanden: -811 / 21000
- Wenn ich aber genau die erste Zeile brauche?
- Üblich ist, stets mit Cursor zu arbeiten.
- Ergebnis: 1 Addressraumwechsel
- Warum nicht anders? Geht es anders?

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

SQL normal

```
 EXEC SQL SELECT
            FK_POL_NR
          , FK_OFFERT_VAR
          , DN_ENTITY_ID_ATTTY
          , GUELTIG_AB
          , VERSION
. . .
     INTO
          :DB_FELDER_N.FK_POL_NR
        , :DB_FELDER_N.FK_OFFERT_VAR
        , :DB_FELDER_N.DN_ENTITY_ID_ATTTY
        , :DB_FELDER_N.GUELTIG_AB
        , :DB_FELDER_N.VERSION
. . .
     FROM  UFLO_UTEXAZU
     WHERE FK_POL_NR = 2000101
     WITH UR ;
```

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Cursor-Verarbeitung – 1

```
   EXEC SQL
   DECLARE C_N_I       CURSOR FOR
   SELECT
          FK_POL_NR
        , FK_OFFERT_VAR
        , DN_ENTITY_ID_ATTTY
        , GUELTIG_AB
        , VERSION
. . .
   FROM  UFLO_UTEXAZU
   WHERE FK_POL_NR = 2000101
   FOR FETCH ONLY
   WITH UR
   ;
```

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Cursor-Verarbeitung – 2

```
      . . .
      EXEC SQL OPEN  C_N_I;
      . . .
      EXEC SQL FETCH C_N_I
      INTO
           :DB_FELDER_N.FK_POL_NR
         , :DB_FELDER_N.FK_OFFERT_VAR
         , :DB_FELDER_N.DN_ENTITY_ID_ATTTY
         , :DB_FELDER_N.GUELTIG_AB
         , :DB_FELDER_N.VERSION
. . .
      ;
      . . .
      EXEC SQL CLOSE C_N_I ;
      . . .
```

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

SQL normal (Wdh.)

```
 EXEC SQL SELECT
            FK_POL_NR
          , FK_OFFERT_VAR
          , DN_ENTITY_ID_ATTTY
          , GUELTIG_AB
          , VERSION
. . .
      INTO
           :DB_FELDER_N.FK_POL_NR
         , :DB_FELDER_N.FK_OFFERT_VAR
         , :DB_FELDER_N.DN_ENTITY_ID_ATTTY
         , :DB_FELDER_N.GUELTIG_AB
         , :DB_FELDER_N.VERSION
. . .
      FROM  UFLO_UTEXAZU
      WHERE FK_POL_NR = 2000101
      WITH UR

      ;
```

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

SQL mit FETCH FIRST ROW

```
 EXEC SQL SELECT
            FK_POL_NR
          , FK_OFFERT_VAR
          , DN_ENTITY_ID_ATTTY
          , GUELTIG_AB
          , VERSION
. . .
      INTO
           :DB_FELDER_N.FK_POL_NR
         , :DB_FELDER_N.FK_OFFERT_VAR
         , :DB_FELDER_N.DN_ENTITY_ID_ATTTY
         , :DB_FELDER_N.GUELTIG_AB
         , :DB_FELDER_N.VERSION
. . .
      FROM  UFLO_UTEXAZU
      WHERE FK_POL_NR = 2000101
      WITH UR
      FETCH FIRST ROW ONLY
      ;
```

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Testvorgehensweise

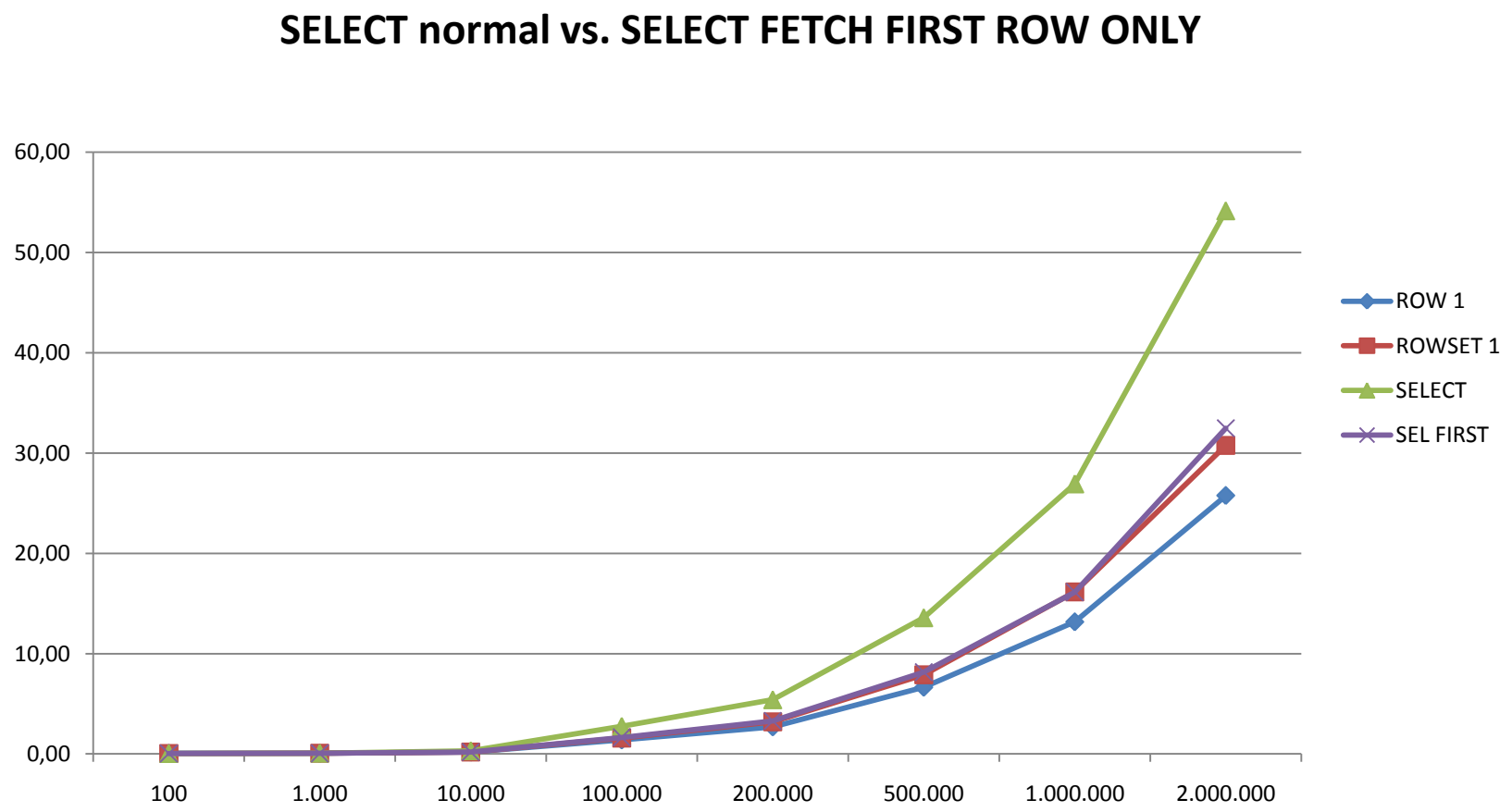
- Test mit „normalem“ Fetch
- Test mit multiple-row-Fetch 1 Zeile
- unterschiedliche Anzahl Zugriffe
- ...100, 1‘000, 10‘000, 100‘000
- Loop über die Zugriffe

Link

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Ergebnisse



SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Fazit

- SELECT mit FETCH FIRST ROW effektiver.
- Es lohnt sich, darüber nachzudenken, dass und wann mit Zusatzangabe gearbeitet werden sollte.

SQL-Code – SELECT ... FETCH FIRST ROW

cps4it

Programme

- vollständige Testprogramme stehen unter SDE
- Kopien (auch JCL unter X016291.#.DB2V9.SRC)
- PL1-Version: U442RLF
- COBOL-Version: U441RLF

Link PL1-Programm

Link COBOL-Programm

Link JCL

c’est tout

~~Fragen?~~ **Danke!!**