

DB2 for z/OS

Teil 3 – Anwendungsentwicklung und DB2

cps4it

consulting, projektmanagement und seminare für die informationstechnologie

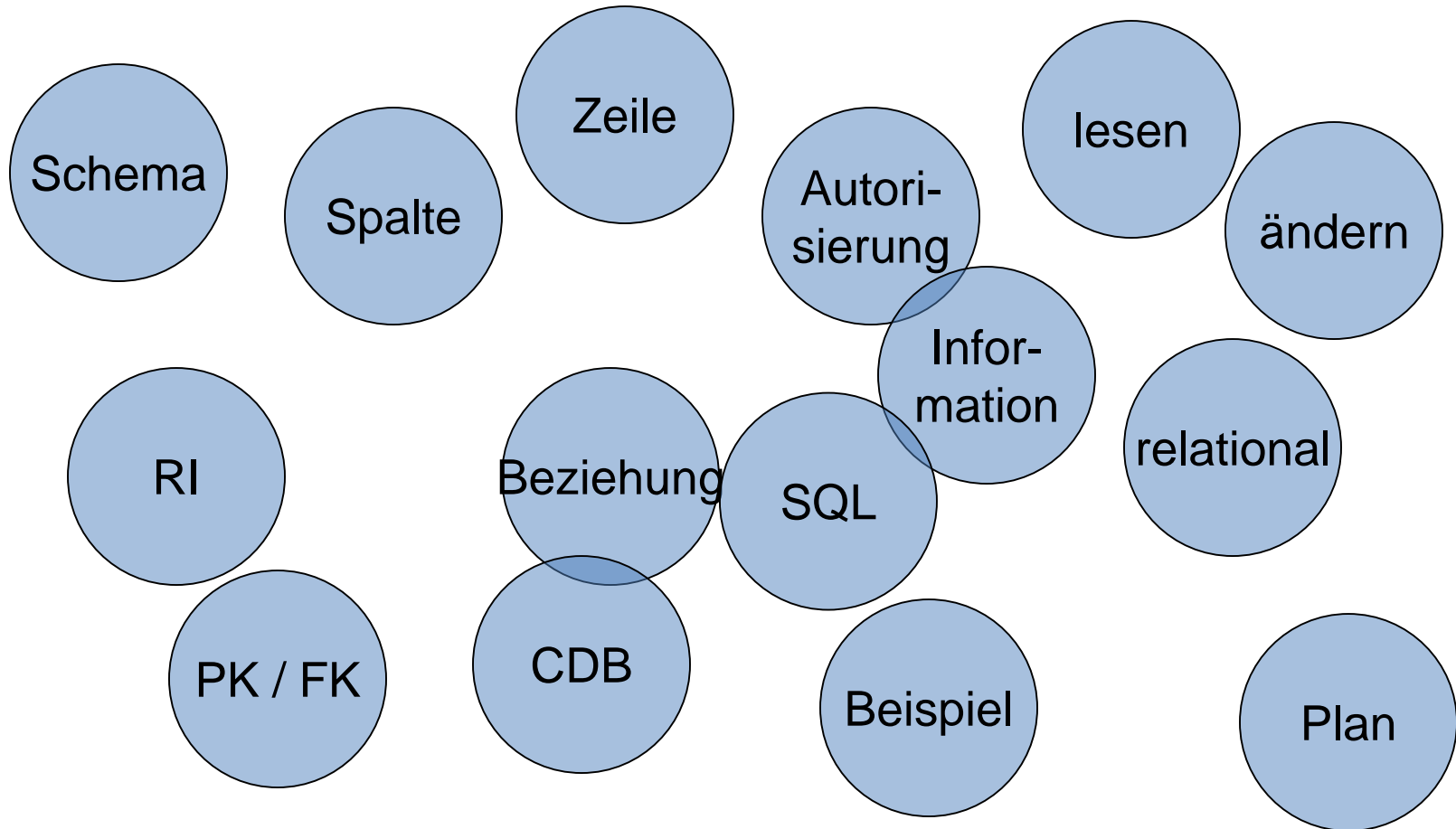
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen

Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de

Internet: <http://www.cps4it.de>

-
- ➔ • DB2-Systemkatalog
 - DB2-Utilities
 - SQL im Anwendungsprogramm
 - Cursor-Verarbeitung
 - Explain und Performance

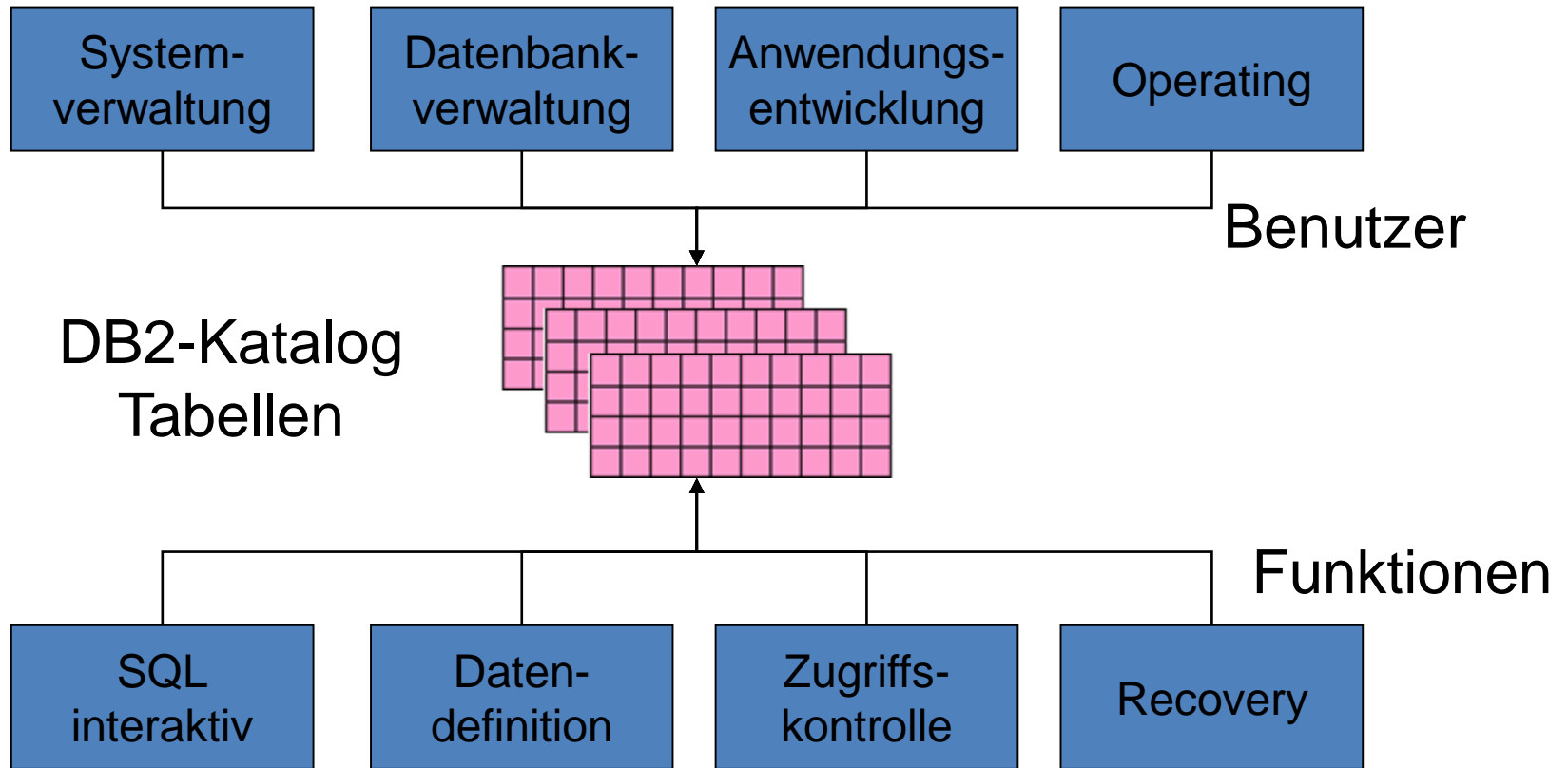
Begriffe








allgemeines

- enthalten alle Informationen zu allen Tabellen innerhalb DB2
- Präfix (d.h. Creator): SYSIBM
- logisch aufgeteilt
- bauen auf eigenem Datenmodell auf

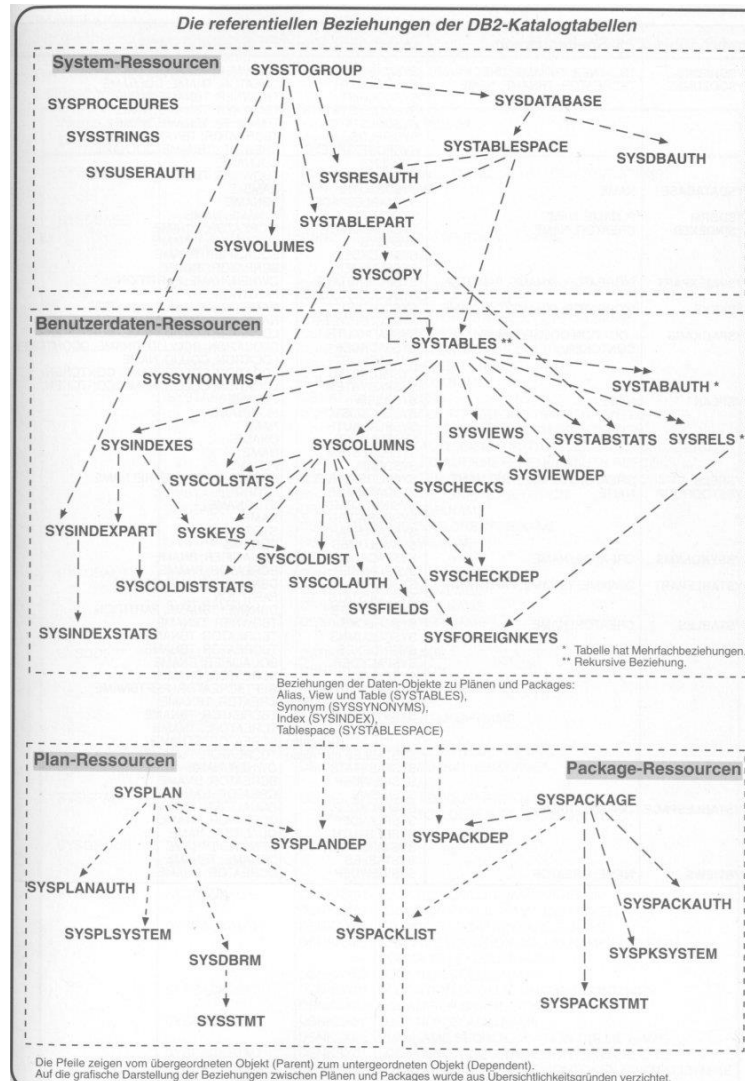
Schema



Liste der Tabellen mit Funktionen

Ressource-Typ	Tabelle	Inhalte
Benutzer-Privilegien Autorisierung 	SYSCOLAUTH SYSDBAUTH SYSPACKAUTH SYSPLANAUTH SYSRESAUTH SYSTABAUTH SYSUSERAUTH	GRANT-Privilegien für Update einzelner Spalten einer Tabelle (TABLE) GRANT-Privilegien für eine bestimmte Database (DATABASE) GRANT-Privilegien für Packages (PACKAGE) GRANT-Privilegien für Application Plans (PLAN) GRANT-Privilegien für Collections, Bufferpools, Storage Groups und Tablespaces (USE) GRANT-Privilegien für Tables und Views (TABLE/VIEW) GRANT-Privilegien für System-Privilegien (SYSTEM)
Benutzerbezogene Objekt-Typen: Tabelle (Base-Table), View, Alias, Synonym. 	SYSCHECKDEP SYSCHECKS SYSCOLUMNS SYSFIELDS SYSFORIGNKEYS SYSRELS SYSSYNONYMS SYSTABLES SYSVIEWDEP SYSVIEWS	Abhängigkeiten eines Check Constraints zu Spalten einer Tabelle Eine Zeile pro Check Constraint einer Tabelle Eine Zeile pro Spalte jeder Tabelle und jedes Views Eine Zeile für jede Spalte mit Feld-Prozedur Eine Zeile für jede Spalte eines Foreign Keys Eine Zeile pro referential constraint Eine Zeile pro Synonym einer Tabelle bzw. eines Views Eine Zeile pro Tabelle, Alias oder View Abhängigkeiten zwischen einem View und Basis-Tabellen oder anderen Views SQL-SELECT-Definition eines Views
Systembezogene Objekt-Typen: Database, Tablespace, Index und Indexspace, Storage Group, Volumes 	SYSCOLDIST SYSDATABASE SYSINDEXES SYSINDEXPART SYSKEYS SYSSTOGROUP SYSTABLEPART SYSTABLESPACE SYSVOLUMES	Für die erste Index-Spalte die inhaltliche Streuung bei ungleicher Verteilung. Eine Zeile pro Database Eine Zeile pro Index Eine Zeile für jeden unpartitioned Index bzw. jede Index-Partition Eine Zeile pro Spalte eines Index-Keys Eine Zeile pro Storage Group Eine Zeile pro Tablespace bzw. pro Partition Eine Zeile pro Tablespace Eine Zeile pro Volume jeder Storage Group
Anwendungen Collection DBRM Package Plan Stored Procedures 	SYSDBRM SYSPACKAGE SYSPACKDEP SYSPACKLIST SYSPACKSTMT SYSPKSYSTEM SYSPLAN SYSPLANDEP SYSPLSYSTEM SYSPROCEDURES SYSSTMT	Eine Zeile pro Data Base Request Modul (DBRM) Eine Zeile pro Package des lokalen Servers Abhängigkeiten der Package zu lokalen Objekten Packages und/oder Collections eines Plans Texte der SQL-Statements einer Package Ausführbare Connections einer Package Eine Zeile pro Plan Abhängigkeiten des Plans zu lokalen Objekten Ausführbare Connections eines Plans Eine Zeile pro Stored Procedure Texte der SQL-Statements eines DBRMs
Partition-Level- Statistik-Tabellen 	SYSCOLDISTSTATS SYSCOLSTATS SYSINDEXSTATS SYSTABSTATS	Partitionbezogene Infos analog SYSCOLDIST Partitionbezogene Infos analog SYSCOLUMNS Partitionbezogene Infos analog SYSINDEXES Partitionbezogene Infos analog SYSTABLES
Sonstiges Recovery-Status Konvertierungsregeln Sonstige Tabellen	SYSCOPY SYSDUMMY1 SYSSTRINGS	Informationen für das Recovery-Utility Dummy-Tabelle für Nutzung von Funktionen, die keine Daten benötigen Konvertierungs-Informationen für Code-Umsetzung

Beziehungen zwischen den Tabellen



weitere Details ...

- PK-FK-Beziehungen zwischen den Tabellen
 - Informationen der Katalog-Tablespaces und Indizes
 - was mit den Tabellen gemacht werden darf
 - Katalog-Strukturen
 - etc.
-
- siehe DB2 Theorie und Praxis (Denne)

Benutzung

- Benutzer kann nur Abfragen
- Änderungen über Befehle
 - CREATE, DROP, ALTER für DB2-Objekte
 - GRANT, REVOKE für Autorisierungen
- Informationen über Kopieren von Datenbanken

- **SYSTABLES**
 - 1 Zeile je Table und View
 - Felder: NAME, CREATOR, COLCOUNT ...
- **SYSCOLUMNS**
 - 1 Zeile für jede Spalte aller Tabellen
 - Felder: NAME, TBNAME, COLTYPE ...
- **SYSINDEXES**
 - 1 Zeile für jeden Index
 - Felder: NAME, TBNAME, CREATOR

Beispiel für Abfragen – 1

- Aufgabe:
 - Welche Tabellen enthalten Spalten mit dem Namen LNR?

- Befehl

```
SELECT TBNAME
      FROM SYSIBM.SYSCOLUMNS
      WHERE NAME = 'LNR'
```

- Ergebnis

```
TBNAME
-----
LIEFERANT
AUFTRAG
```

Beispiel für Abfragen – 2

- Aufgabe:
 - Welche Spalten hat die Tabelle LIEFERANT?

- Befehl

```
SELECT NAME
      FROM SYSIBM.SYSCOLUMNS
      WHERE TBNAME = 'LIEFERANT'
```

- Ergebnis

```
NAME
-----
LNR
LNAME
LSTATUS
ORT
```

Übung(en)

- Kapitel 1.4.5.31 Beispiel 31
- Kapitel 1.4.5.32 Beispiel 32



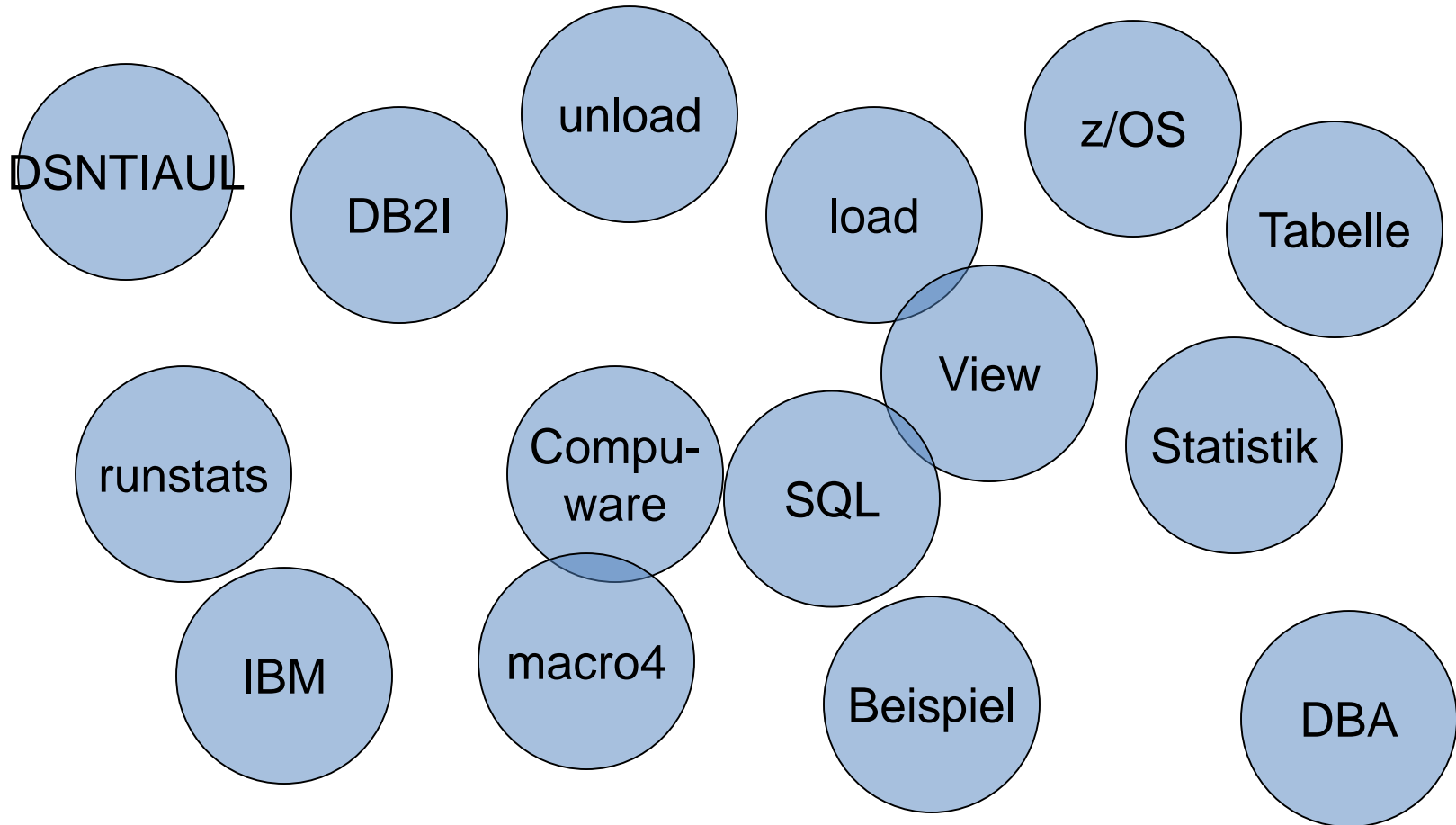
Übung(en)

- Kapitel 8.1 versch. Infos auslesen



-
- DB2-Systemkatalog
 - • DB2-Utilities
 - SQL im Anwendungsprogramm
 - Cursor-Verarbeitung
 - Explain und Performance

Begriffe



allgemeines

- (fast) alle Funktionen sind im Dialog aufrufbar
 - DSNTIAUL unload
 - DSNTIB71 unload
 - DSNU (Clist) load
 - DSNUTILB runstats
- Benötigt werden diese im Allgemeinen nicht, da jede Firma eigene Verfahren zu Verfügung stellt
- Trotzdem ist es interessant zu wissen, dass es so etwas gibt.

- ASM-Programm
- bis 100 Tabellen selektieren und entladen auf PS-Datei
 - kompletter Inhalt
 - Auswahl von Daten wie
 - alle Spalten
 - bestimmte Zeilen über Tabellename
 - Spaltenauswahl mit beliebiger Selektion über View
 - parametrisiert mit einem SQL
- generieren von Lade-Statements

DSNTIAUL – Beispiel 1

```
//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(DSN)
  RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB81) -
    LIB('DSN810.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSREC00 DD DSN=DSN8UNLD.SYSREC00,
//
  UNIT=SYSDA,SPACE=(32760,(1000,500)),DISP=(,CATLG),
//
  VOL=SER=SCR03
//SYSPUNCH DD DSN=DSN8UNLD.SYSPUNCH,
//
  UNIT=SYSDA,SPACE=(800,(15,15)),DISP=(,CATLG),
//
  VOL=SER=SCR03,RECFM=FB,LRECL=120,BLKSIZE=1200
//SYSIN DD *
DSN8810.PROJ WHERE DEPTNO='D01'
```

DSNTIAUL – Beispiel 2

```
//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(DSN)
  RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB81) -
    LIB('DSN810.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSREC00 DD DUMMY
//SYSPUNCH DD DSN=DSN8UNLD.SYSPUNCH,
//          UNIT=SYSDA,SPACE=(800,(15,15)),DISP=(,CATLG),
//          VOL=SER=SCR03,RECFM=FB,LRECL=120,BLKSIZE=1200
//SYSIN DD *
DSN8810.DEPT
```

DSNTIAUL – Beispiel 3

```
//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(DSN)
  RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB81) PARS('SQL,250') -
      LIB('DSN810.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
...
...
//SYSIN DD *
LOCK TABLE DSN8810.EMP IN SHARE MODE;
LOCK TABLE DSN8810.PROJ IN SHARE MODE;
SELECT * FROM DSN8810.PROJ;
SELECT * FROM DSN8810.EMP
  WHERE WORKDEPT LIKE 'D%'
  ORDER BY EMPNO;
```

250 rows pro fetch



- **DSNU**
 - Clist
 - generiert JCL zum Laden
 - JCL editierbar
 - Aufruf über TSO DSNU
- **DSNUTILB**
 - ASM-Programm
 - Aktualisieren der Runstats
 - Aufruf über DB2I

DSNTEP2

- Batch-Programm
- Ausführung von dynamischen SQLs
- Beispiel:

```
//RUNTEP2 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(DSN)
  RUN PROGRAM(DSNTEP2) PLAN(DSNTEP81) -
    LIB('SYS1.DB2.LINKLIB')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
  SELECT * FROM DSN8810.PROJ;
```

und Produkte von verschiedenen Firmen ...

- FileAid for DB2 (Compuware)
- File Manager for z/OS (IBM)
- InSync (macro4)
- BMC Mastermind / DB2 Catalog Manager
- Plan Analyzer (Platinum)
- QuickStart

Produkt FileAid for DB2

- Tabellen ansehen
- Tabellen ändern
- DB2-Objekte anlegen, löschen, ändern
- Tabellen laden / entladen
- SQL im Batch laufen lassen
- etc.

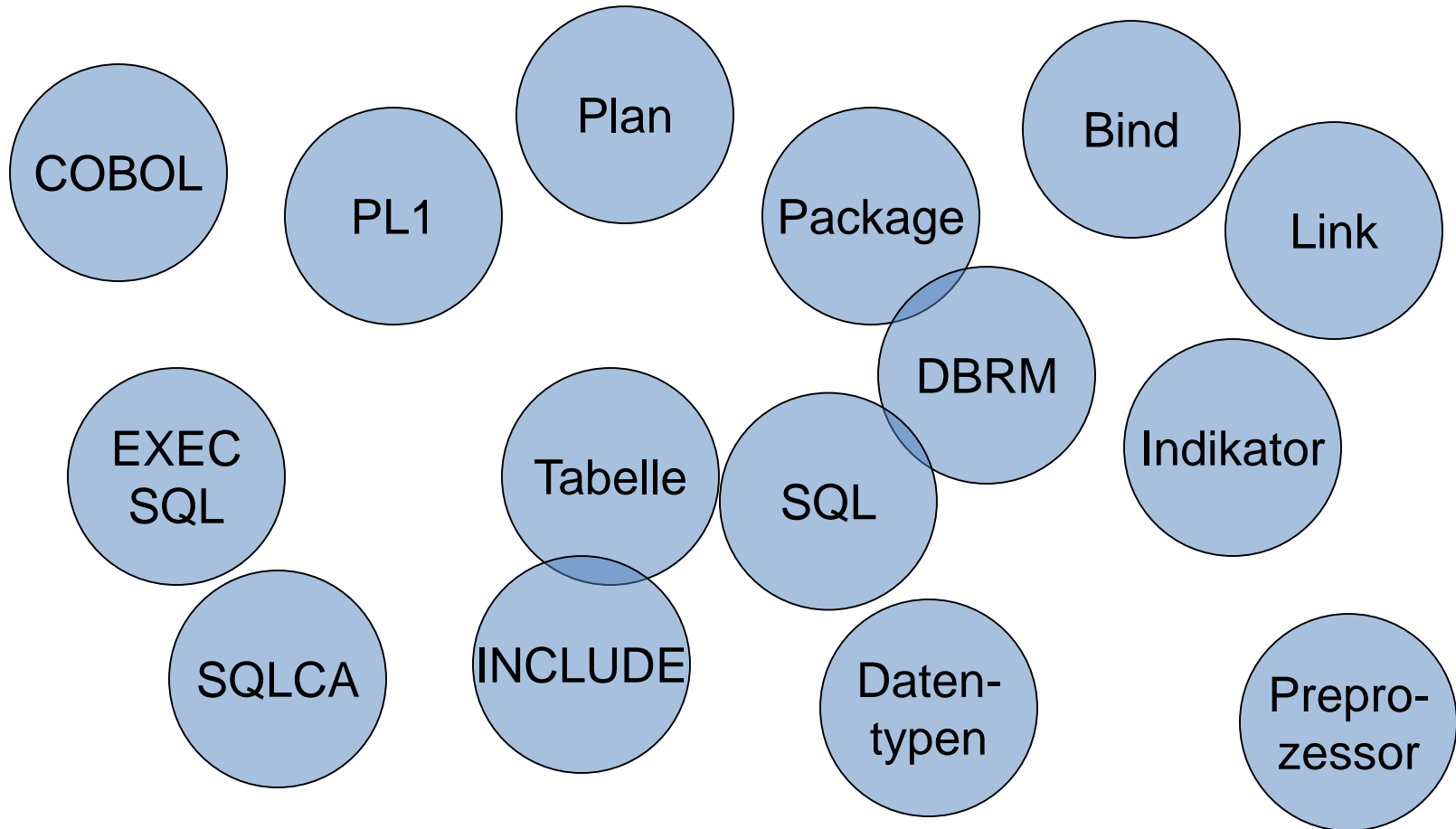
Übung(en)

- Kapitel 9.1 Tabelle entladen
- Kapitel 9.2 Tabelle teilweise entladen
- Kapitel 9.3 Runstats aktualisieren
- Kapitel 9.4 BMC Catalog Manager nutzen
- Kapitel 9.5 Aufruf Quickstart
- Kapitel 9.6 spielen mit FileAid DB2



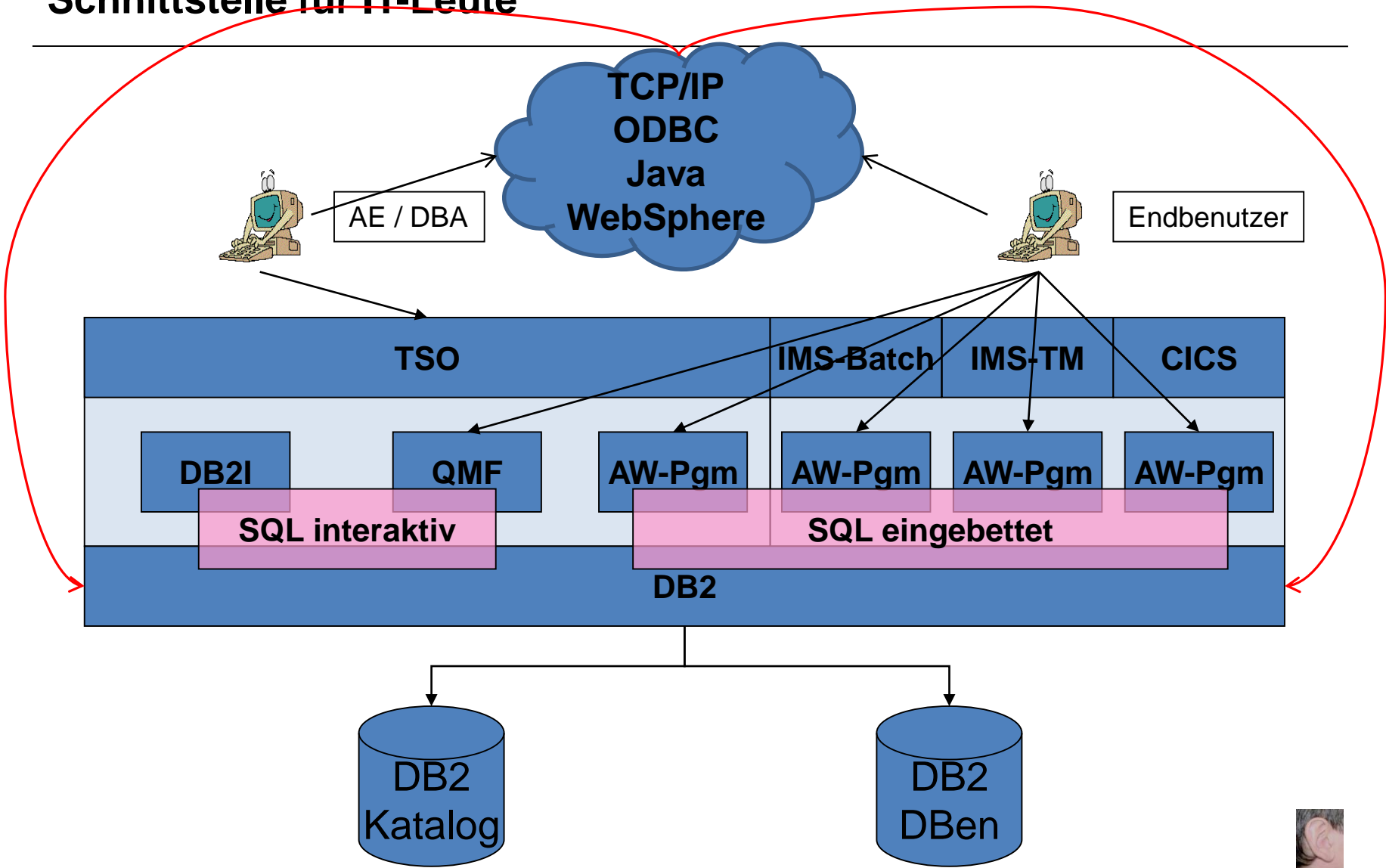
-
- DB2-Systemkatalog
 - DB2-Utilities
 - • SQL im Anwendungsprogramm
 - Cursor-Verarbeitung
 - Explain und Performance

Begriffe



SQL im Anwendungsprogramm

Schnittstelle für IT-Leute



allgemeines – 1

- alle SQL-Befehle können eingebettet werden
 - DML (SELECT, UPDATE, ...)
 - DDL (CREATE TABLE, ...)
 - DCL (GRANT, REVOKE, ...)
- erlaubte Programmiersprachen
 - PL1, COBOL, Fortran
 - Assembler, C, C++, Ada
 - Basic, APL2
 - JAVA, PROLOG, LE

allgemeines – 2

- Es gibt Detailunterschiede zwischen interaktiven und eingebetteten SQL-Befehlen.
- Es sind zusätzliche SQL-Befehle im Anwendungsprogramm nötig.
- Die Logik ist unterschiedlich:
 - Programmiersprachen sind “prozedural”
Ein Satz nach dem anderen.
 - SQL-Sprache ist “nicht-prozedural”
SET-Verarbeitung

Kennzeichnung

- COBOL: **EXEC SQL** SELECT ... **END-EXEC.**
- Fortran: **EXEC SQL** SELECT ...
- PL1: **EXEC SQL** SELECT ... ;
- ASM: **EXEC SQL** SELECT ...

Beispiel – PL1

- Tabelle:
 - LIEFERANT (LNR, LNAME, LSTATUS, ORT)
- Befehl:

```
EXEC SQL
```

```
    SELECT LSTATUS, ORT  
        INTO :LSTATUS, :STADT  
        FROM LIEFERANT  
        WHERE LNR = :LNR
```

```
;
```

Beispiel – COBOL

- Tabelle:
 - LIEFERANT (LNR, LNAME, LSTATUS, ORT)
- Befehl:

```
EXEC SQL
    SELECT LSTATUS, ORT
        INTO :LSTATUS, :STADT
        FROM LIEFERANT
        WHERE LNR = :LNR
END-EXEC.
```

Hostvariable

- Felder im SQL, die im Programm definiert sind, nennt man Hostvariablen. Diese sind durch einen **:** vor dem Namen gekennzeichnet.
- Hostvariablen und Datenbankfelder können den gleichen Namen tragen.

Definition von Hostvariablen

- PL1:

```
DCL LSTATUS FIXED BIN(15) ;  
DCL STADT CHAR(10) ;  
DCL LNR CHAR(06) ;
```

- COBOL:

```
01 LSTATUS PIC S9(04) BINARY .  
01 STADT PIC X(10) .  
01 LNR PIC X(06) .
```

Definition von Hostvariablen – Tabelle

DB2	COBOL	PL1
INTEGER	PIC S9(9) COMP	BIN FIXED(31)
SMALLINT	PIC S9(4) COMP	BIN FIXED(15)
DECIMAL(m,n)	PIC S9(m-n)V9(n) COMP-3	FIXED DEC(m,n)
FLOAT	USAGE COMP-2	BIN FLOAT(53)
CHAR(n)	PIC X(n)	CHAR(n)
VARCHAR(n) LONG VARCHAR	PIC S9(4) COMP PIC X(n)	CHAR(n) VARYING
DATE	PIC X(10)	CHAR(10)
TIME	PIC X(8)	CHAR(8)
TIMESTAMP	PIC X(26)	CHAR(26)



Fehlerbehandlung

- Nach Ausführung eines SQL-Befehls stellt DB2 in einem Bereich Statusinformationen bereit.
- SQLCA
 - Kommunikationsbereich zwischen DB2 und Anwendungsprogramm
 - “Communication Area”
 - muss im Anwendungsprogramm definiert sein
 - zentrales Copybook; wird per SQL INCLUDE bereitgestellt

Fehlerbehandlung – SQLCODE

- SQLCODE ist ein Feld in SQLCA
- SQLCODE = 0 alles ok
- SQLCODE = 100 keine Daten gefunden
- SQLCODE < 0 Fehlersituation eingetreten

SQLCA Beschreibung

SQLCA Struktur

Fehlerbehandlung – Definition der SQLCA in COBOL

01 SQLCA.

```
05 SQLCAID      PIC X(8) .
05 SQLCABC      PIC S9(9) COMP-4 .
05 SQLCODE      PIC S9(9) COMP-4 .
05 SQLERRM.
    49 SQLERRML  PIC S9(4) COMP-4 .
    49 SQLERRMC  PIC X(70) .
05 SQLERRP      PIC X(8) .
05 SQLERRD      OCCURS 6 TIMES
                PIC S9(9) COMP-4 .

05 SQLWARN.
    10 SQLWARN0  PIC X.
    10 SQLWARN1  PIC X.
    10 SQLWARN2  PIC X.
    10 SQLWARN3  PIC X.
    10 SQLWARN4  PIC X.
    10 SQLWARN5  PIC X.
    10 SQLWARN6  PIC X.
    10 SQLWARN7  PIC X.

05 SQLEXT.
    10 SQLWARN8  PIC X.
    10 SQLWARN9  PIC X.
    10 SQLWARNA  PIC X.
    10 SQLSTATE  PIC X(5) .
```


Fehlerbehandlung – Definition der SQLCA in PL1

```
DECLARE
```

```
  1 SQLCA,  
    2 SQLCAID CHAR(8),  
    2 SQLCABC FIXED(31) BINARY,  
    2 SQLCODE FIXED(31) BINARY,  
    2 SQLERRM CHAR(70) VAR,  
    2 SQLERRP CHAR(8),  
    2 SQLERRD(6) FIXED(31) BINARY,  
    2 SQLWARN,  
      3 SQLWARN0 CHAR(1),  
      3 SQLWARN1 CHAR(1),  
      3 SQLWARN2 CHAR(1),  
      3 SQLWARN3 CHAR(1),  
      3 SQLWARN4 CHAR(1),  
      3 SQLWARN5 CHAR(1),  
      3 SQLWARN6 CHAR(1),  
      3 SQLWARN7 CHAR(1),  
    2 SQLEXT,  
      3 SQLWARN8 CHAR(1),  
      3 SQLWARN9 CHAR(1),  
      3 SQLWARNA CHAR(1),  
      3 SQLSTATE CHAR(5);
```

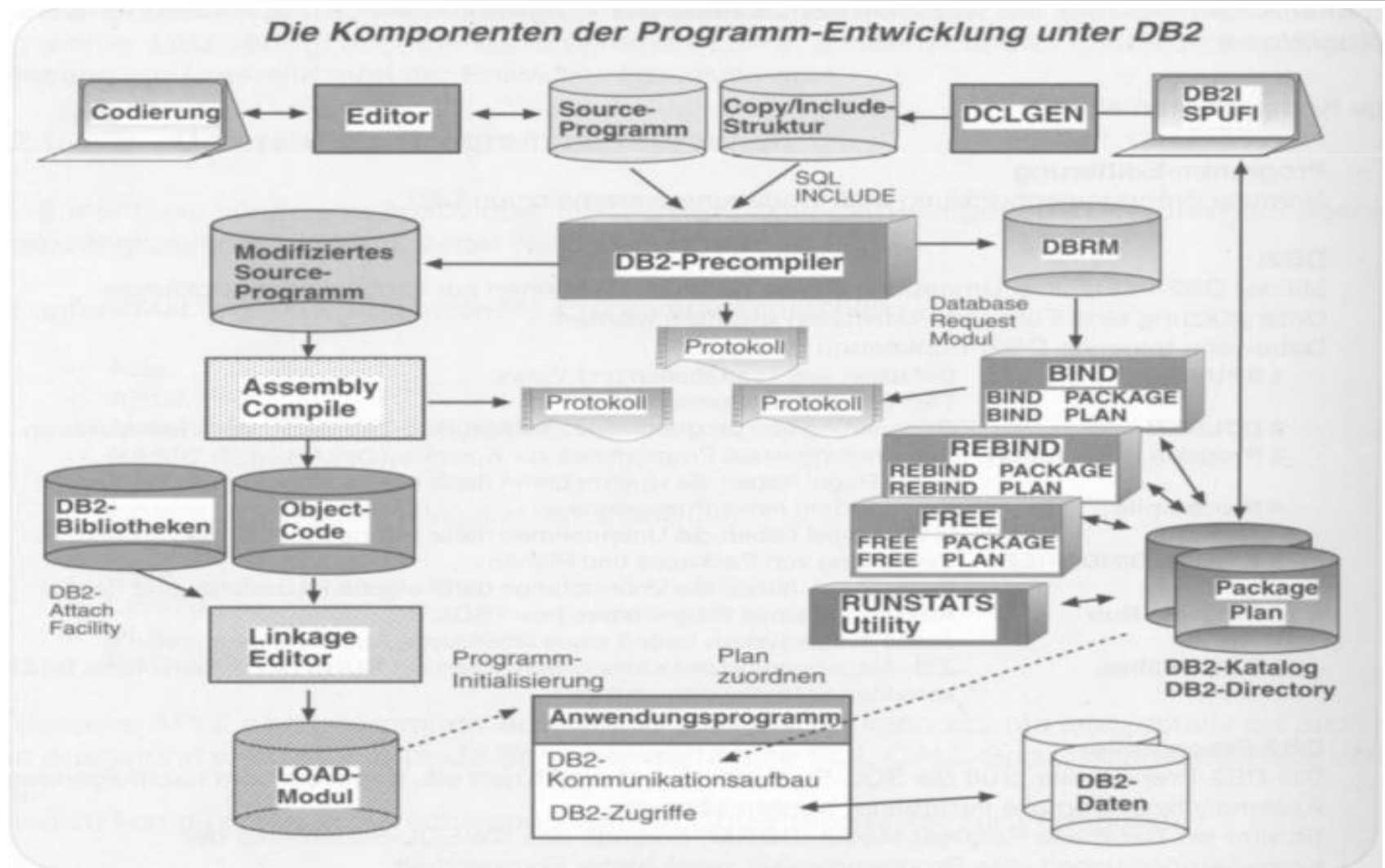
Fehlerbehandlung – Felder in der SQLCA (Auswahl)

- SQLCODE Returncode
- SQLERRM Fehlernachricht
- SQLERRP DB2-Modulname
- SQLERRD(3) Anzahl veränderter Zeilen
- SQLWARN0 blank: alles ok / 'W': teste 1-7
- SQLWARN1 Truncation bei char-Feldern
- SQLWARN2 NULL-Werte bei Funktionen
wurden eliminiert
- SQLWARN3 mehr Spalten als Hostvar.



SQL im Anwendungsprogramm

Programmentwicklung im DB2



- Programm schreiben
- Precompile
 - Syntaxprüfung
 - Prüfung auf Existenz der Hostvariablen
 - umsetzen SQL-Code in Programm-Code
 - generieren DBRM
 - bei Anwesenheit einer Declare-Struktur wird Kompatibilität geprüft
- Compile

- Bind Package
 - Syntaxprüfung der SQLs
 - prüfen, ob DB2-Objekte existieren
 - Berechtigung prüfen
 - Auswahl optimaler Zugriffspfad
 - berücksichtigen Indizes
 - berücksichtigen Runstats
 - generieren Package bzw. Plan
- Bind / Link

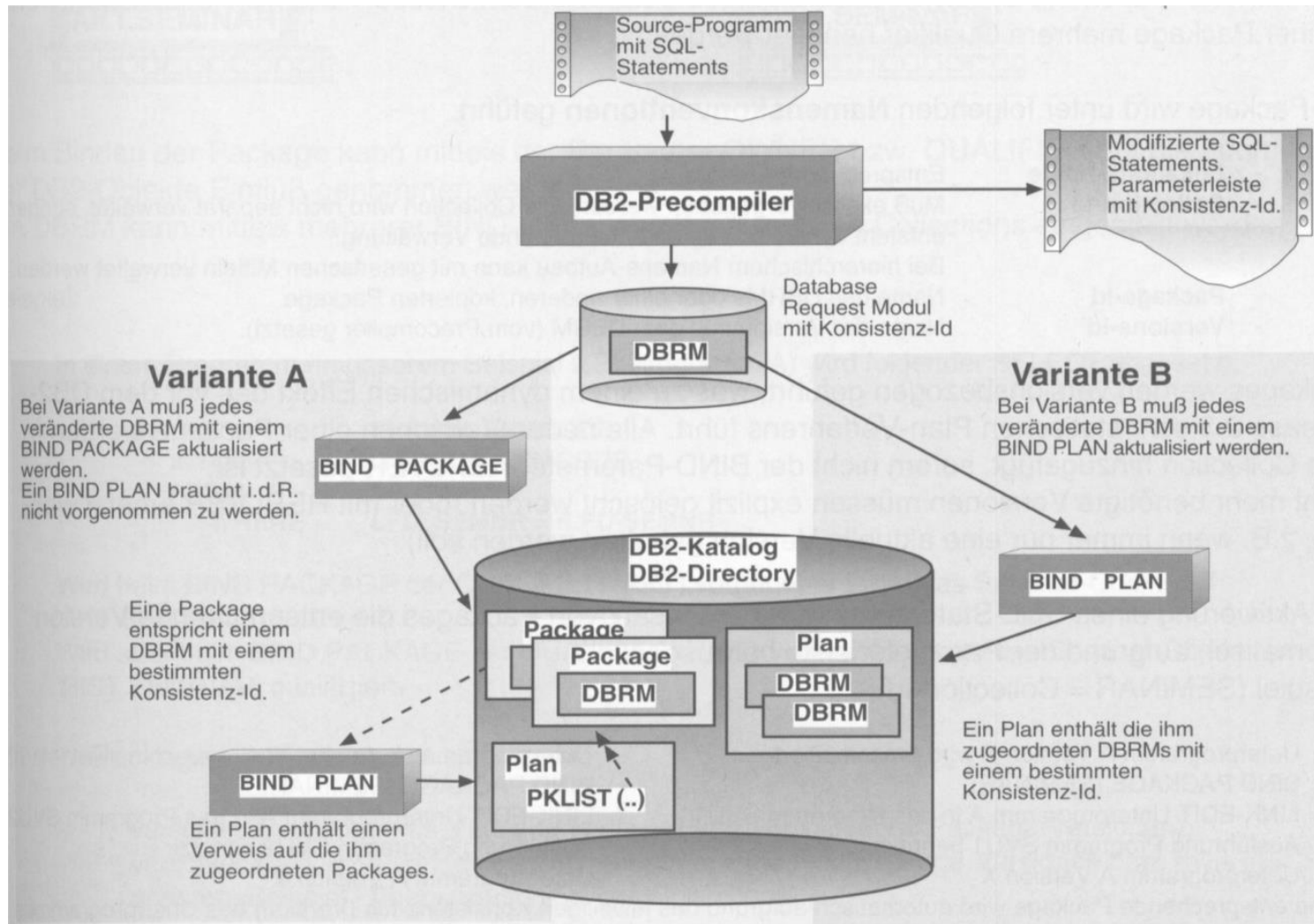
SQL im Anwendungsprogramm

Modifikation durch DB2-Preprozessor

SQL-Statement Vorgabe im Programm:	EXEC SQL SELECT INTO FROM WHERE END-EXEC	SEMCODE, TERMIN :SEMCODE, :TERMIN V004711 LFD_SEMNR = :LFD_SEMNR
SQL-Statement Modifikation durch den DB2-Precompiler	** EXEC SQL ** SELECT ** INTO ** FROM ** WHERE ** END-EXEC ** PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE ** CALL 'DSNHLL' USING SQL-PLIST2	
Durch den DB2-Precompiler eingefügte Routine zur Adressierung der Parameterliste für das Attach-Facility	SQL SKIP. GO TO SQL-INIT-END. SQL-INITIAL MOVE 1 TO SQL-INIT-FLAG. CALL 'DSNHADDR' USING	SQL-VPARAMPTR OF SQL-PLIST2 SQL-PVAR-LIST2 SQL-PVAR-ADDRS OF SQL-PLIST2 LFD_SEMNR OF DCLV004711 SQL-NULL SQL-APARAMPTR OF SQL-PLIST2 SQL-AVAR-LIST2
Durch den DB2-Precompiler eingefügte Parameterliste für das Attach-Facility	CALL 'DSNHADDR' USING	SQL-AVAR-ADDRS OF SQL-PLIST2 SEMCODE OF DCLV004711 SQL-NULL TERMIN OF DCLV004711 SQL-NULL SQL-CODEPTR OF SQL-PLIST2 SQLCA.
	SQL-INIT-END.	

77	SQL-NULL	PIC S9(9)	COMP-4	VALUE +0.	
77	SQL-INIT-FLAG	PIC S9(4)	COMP-4	VALUE +0.	
88	SQL-INIT-DONE	VALUE	COMP-4	+1.	
01	SQL-PLIST2				
05	SQL-PLIST-CON	PIC S9(9)	COMP-4	VALUE +2656256.	Länge fester Teil und Flag.
05	SQL-CALLTYPE	PIC S9(4)	COMP-4	VALUE +30.	Typ des Calls.
05	SQL-PROG-NAME	PIC X(8)	COMP-4	VALUE 'SVU101 '	Programmname, DBRM-Name.
					Die folgenden Felder enthalten den Konsistenz-Id des DBRMs (Timestamp oder Level):
05	SQL-TIMESTAMP-1	PIC S9(9)	COMP-4	VALUE +356396737.	Interner Timestamp 1.
05	SQL-TIMESTAMP-2	PIC S9(9)	COMP-4	VALUE +399142110.	Interner Timestamp 2.
05	SQL-SECTION	PIC S9(4)	COMP-4	VALUE +1.	Sektions-Nr. des Statements im DBRM.
05	SQL-CODEPTR	PIC S9(9)	COMP-4	VALUE +0.	Adresse der SQLCA.
05	SQL-VPARAMPTR	PIC S9(9)	COMP-4	VALUE +0.	Adresse der Variablenliste der Host-Variablen, die in WHERE definiert sind.
05	SQL-APARAMPTR	PIC S9(9)	COMP-4	VALUE +0.	Adresse der Variablenliste der Host-Variablen, die in INTO definiert sind.
05	SQL-STMT-NUM	PIC S9(4)	COMP-4	VALUE +558.	Lfd. Statement-Nr. im Programm.
05	SQL-STMT-TYP	PIC S9(4)	COMP-4	VALUE +231.	SQL-Statement-Typ (hier: SELECT).
05	SQL-PVAR-LIST2.				Es folgt die Liste der Host-Variablen, die in WHERE definiert sind (in unserem Beispiel nur eine).
					Länge dieser Variablenliste in Bytes.
10	SQL-PVAR-SIZE	PIC S9(4)	COMP-4	VALUE +16.	Definition der Host Variable(n), hier nur LFD_SEMNR:
10	SQL-PVAR-DESCS.				Daten-Typ (SQLTYPE), hier: SMALLINT NOT NULL.
15	SQL-PVAR-TYPE1	PIC S9(4)	COMP-4	VALUE +500.	Daten-Länge (SQLLEN), hier: 2 Bytes.
15	SQL-PVAR-LEN1	PIC S9(4)	COMP-4	VALUE +2.	Daten-Länge (SQLLEN), hier: 2 Bytes.
10	SQL-PVAR-ADDRS.				Adressen der Host-Variablen:
15	SQL-PVAR-ADDR1	PIC S9(9)	COMP-4.		Adresse der Host-Variablen mit den Daten.
15	SQL-PVAR-IND1	PIC S9(9)	COMP-4.		Adresse des Null-Indikators.
05	SQL-AVAR-LIST2.				Es folgt die Liste der Host-Variablen, die in INTO definiert sind (in unserem Beispiel zwei).
					Länge dieser Variablenliste in Bytes (inkl. diesem Längenfeld).
10	SQL-AVAR-SIZE	PIC S9(4)	COMP-4	VALUE +28.	Definition der Host Variable(n) - pro Variable 12 Bytes:
10	SQL-AVAR-DESCS.				1. Host-Variable = SEMCODE:
15	SQL-AVAR-TYPE1	PIC S9(4)	COMP-4	VALUE +452.	Daten-Typ (SQLTYPE), hier: CHARACTER NOT NULL.
15	SQL-AVAR-LEN1	PIC S9(4)	COMP-4	VALUE +15.	Daten-Länge (SQLLEN), hier: 15 Bytes.
10	SQL-AVAR-ADDRS.				Adressen der Host-Variablen:
15	SQL-AVAR-ADDR1	PIC S9(9)	COMP-4.		Adresse der Host-Variablen mit den Daten.
15	SQL-AVAR-IND1	PIC S9(9)	COMP-4.		Adresse des Null-Indikators.
15	SQL-AVAR-TYPE2	PIC S9(4)	COMP-4	VALUE +452.	2. Host-Variable = TERMIN (Daten-Typ = Cobol-Format)
15	SQL-AVAR-LEN2	PIC S9(4)	COMP-4	VALUE +10.	Daten-Typ (SQLTYPE), hier: CHARACTER NOT NULL.
15	SQL-AVAR-ADDR2	PIC S9(9)	COMP-4.		Daten-Länge (SQLLEN), hier: 10 Bytes.
15	SQL-AVAR-IND2	PIC S9(9)	COMP-4.		Adresse der Host-Variablen mit den Daten.
					Adresse des Null-Indikators.

DBRM, Plan, Package



Übung(en)

- Kapitel 10.1 Programm mit SQLCA



Kodierregeln

- EXEC SQL INCLUDE SQLCA [END-EXEC.];]
- Sämtliche DB2-Ressourcen sollten im Programm mit DECLARE TABLE definiert werden.
- Hostvariablen oder Hoststrukturen können benutzt werden; nach SQL-Standard liegen diese in einer eigenen SQL-Section

```
EXEC SQL BEGIN DECLARE SECTION
```

```
  . . .
```

```
EXEC SQL END DECLARE SECTION
```



NULL-Werte

- Wenn ein Wert NULL ist, was soll dann als Wert in das Zielfeld übertragen werden?
- Lösung: Indikatoren
- Befehl:

```
EXEC SQL
SELECT TITEL, DAUER
      INTO :TITEL :TITEL-I, :DAUER :DAUER-I
      FROM SEMTYP
      WHERE SEMCODE = :SEMCODE
mit DAUER-I, TITEL-I 2-Byte Binärfeld
```

Indikatoren

- Abfrage:

```
IF DAUER-I = -1 THEN NULL-Wert
```

- Achtung! Precompiler meldet keinen Fehler, wenn Indikatoren fehlen.
- Achtung! Fehler wird genau dann gemeldet, wenn ein NULL-Wert auftritt.
- Achtung! Nachträgliches Eintragen von NULL in der Tabellendefinition ist gefährlich.



Beispiele

SELECT

CREATE

UPDATE

DELETE

sperrn oder nicht sperrn, das ist hier die Frage

- Frage: Was passiert mit anderen Programmen, wenn ich Tabellendaten ändern will?
 - Antwort: Fortgeschrittenenkurs besuchen ;-)
- Antwort ist wichtig wegen
 - Deadlock
 - Laufzeit
 - CPU-Zeit

Begriffe wie LUW, UOW, UOR

- LUW: logical unit of work
- UOW: unit of work (=LUW)
- UOR: unit of recovery
- Syncpoint / Checkpoint
- Rollback / Checkout
- Logdatei schreiben (write ahead)
- Transaktion
- 2-phase-commit

Konkurrenzverarbeitung

- Thema: verlorener Update (*ich* will ändern)
- Thema: UOW noch nicht abgeschlossen (es könnte ja noch etwas passieren)
- Thema: erneutes Lesen (ich will die gleichen Daten)
- Thema: Deadlock

Konkurrenzverarbeitung – Isolation Level

- wenn LOCKSIZE PAGE/ANY oder ROW
 - bei Plan – Parameter: ISOLATION
 - bei Package – Parameter: ISOLATION
 - für einzelnes SQL-Statement
-
- RR repeatable read
 - RS read stability
 - CS cursor stability
 - UR uncommitted read

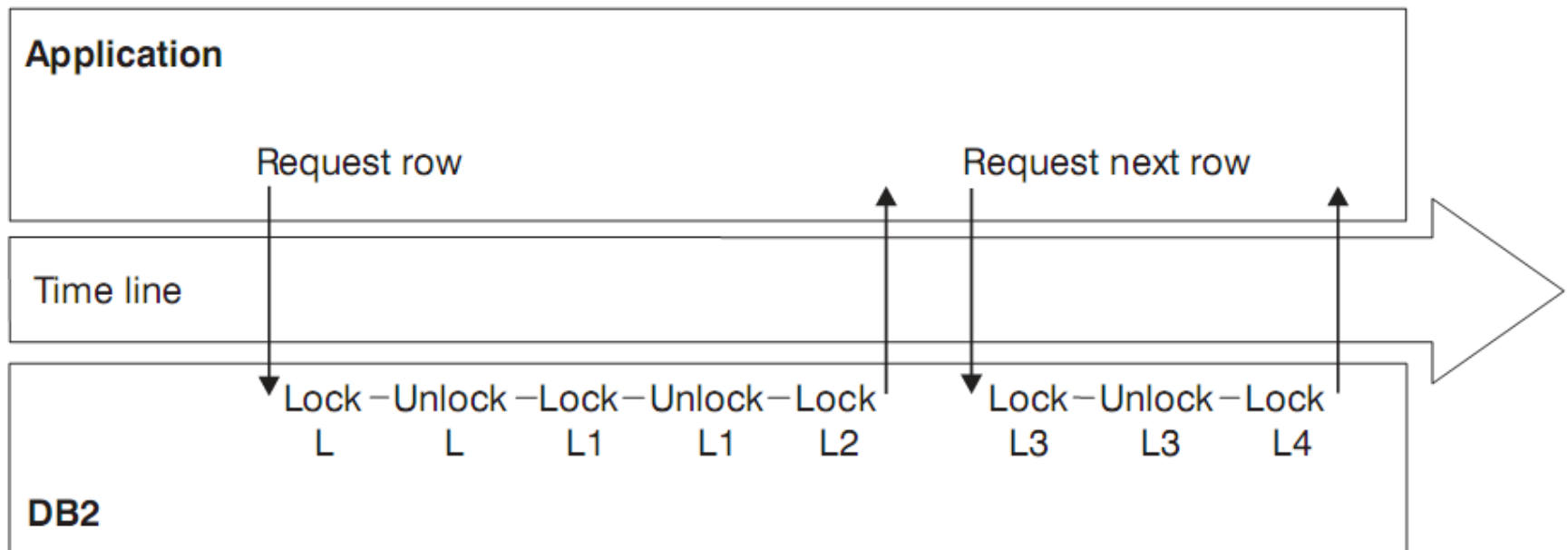
Isolation Level – RR

- RR – Repeatable Read
 - mehrfaches Lesen von Rows oder Pages
 - Jede benutzte Page wird gelockt, selbst wenn sie *nicht* den Predicates genügt.
 - *kein* paralleler Update erlaubt

- RS – Read Stability
 - mehrfaches Lesen von Rows oder Pages
 - Jede benutzte Page wird gelockt, selbst wenn sie *nicht* den Predicates genügt.
 - *paralleler Update teilweise erlaubt
 - Gelockt werden Rows bzw. Pages, die Stage 1 und Stage 2 erfüllen (und keine anderen).

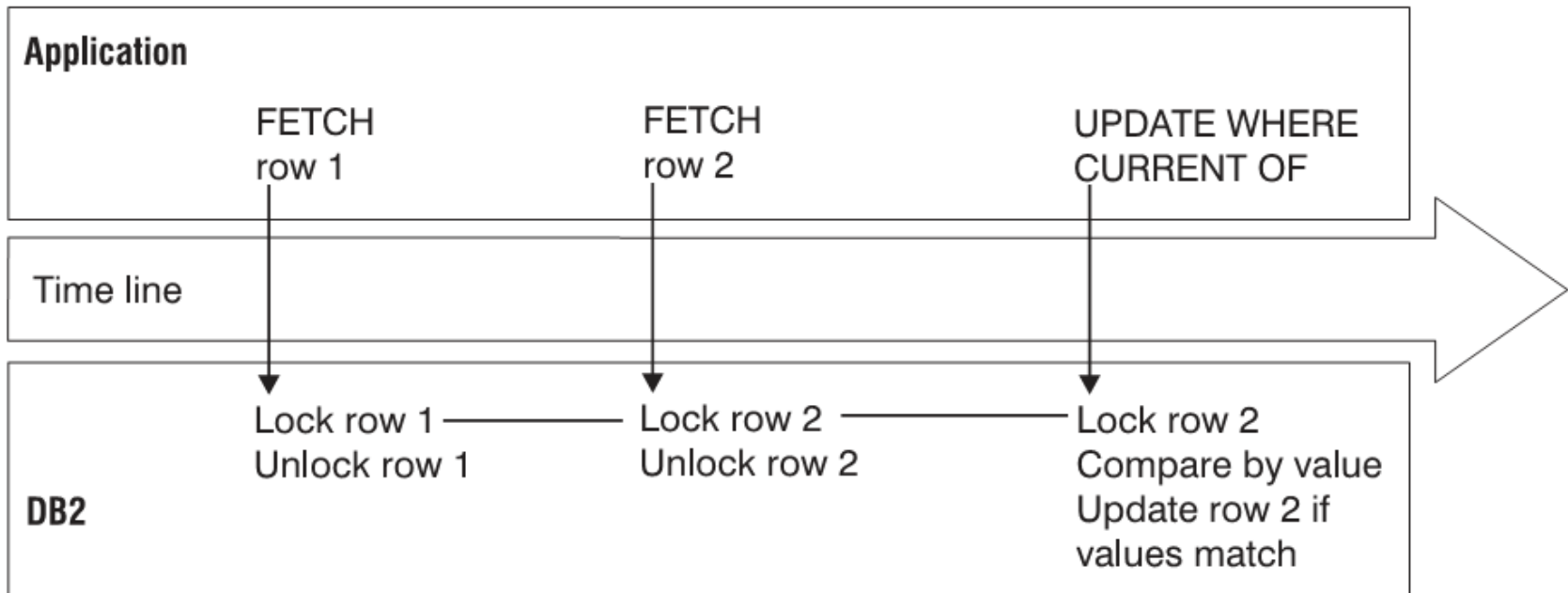
Isolation Level – RS – 2

- RS – Read Stability – Beispiel
 - L2 und L4 erfüllen die Predicates



Isolation Level – CS – 1

- CS – Cursor Stability – höchste Datenintegrität mit „optimistic currency control“



Isolation Level – UR

- UR – Uncommitted Read
 - auch „dirty read“ genannt
 - geht nicht bei DELETE, UPDATE, INSERT, MERGE
 - CURSOR ... FOR UPDATE
 - Sollte immer als Möglichkeit in Betracht gezogen werden. Denn: Kann es denn wirklich sein, dass parallel, also genau zur gleichen Zeit, exakt an diesem einen Objekt etwas getan wird?



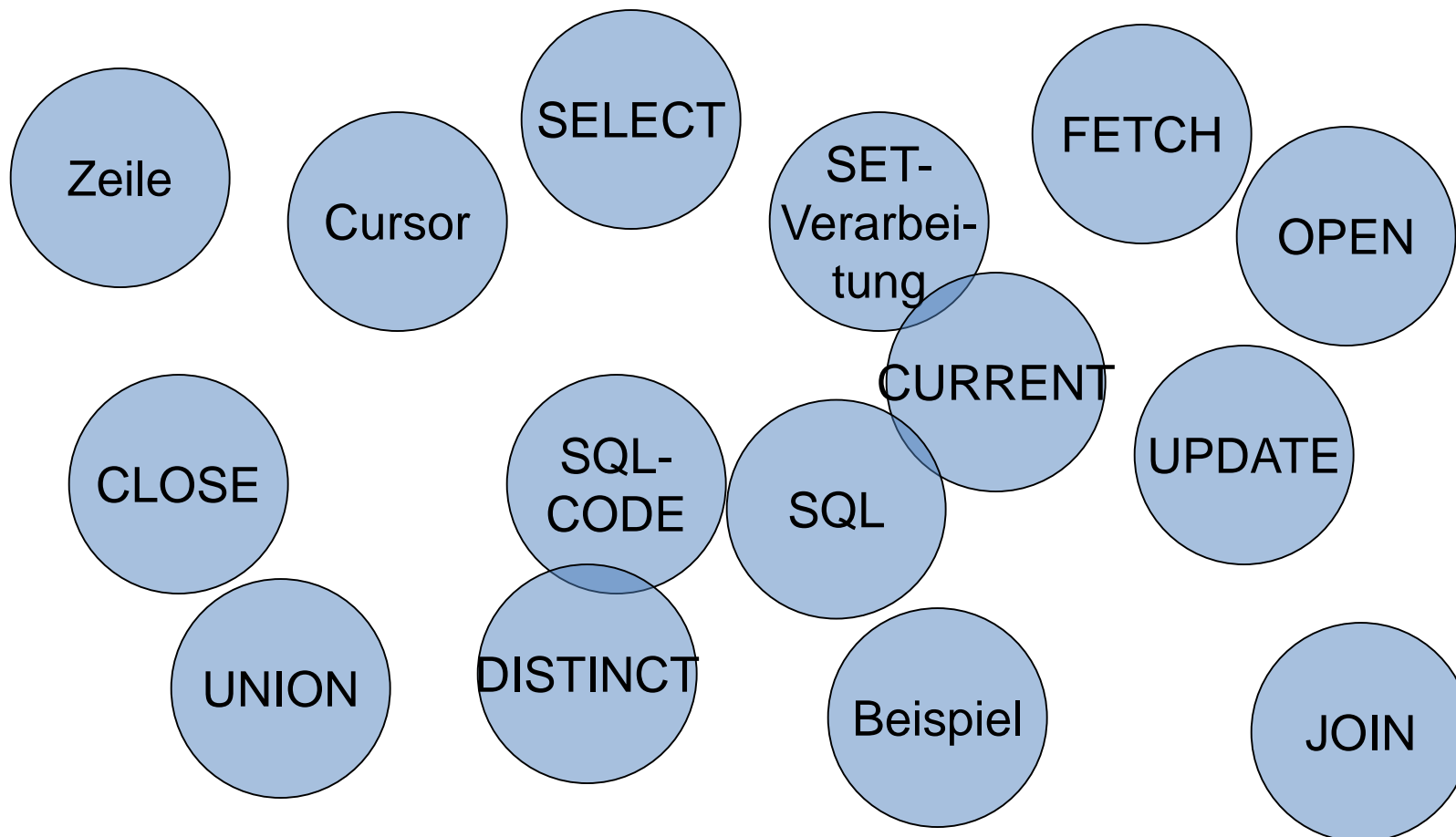
Übung(en)

- Kapitel 10.2 Programm Lesen 1 Zeile



-
- DB2-Systemkatalog
 - DB2-Utilities
 - SQL im Anwendungsprogramm
 - • Cursor-Verarbeitung
 - Explain und Performance

Begriffe

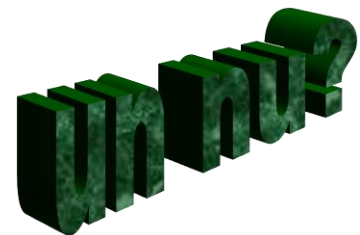


Hintergrund

- Die meisten Programmiersprachen sind reine Verarbeiter von einzelnen Zeilen.
- DB2 macht eine SET-Verarbeitung; dabei ist nicht klar, wie viele Zeilen als Ergebnis geliefert werden.

```
SELECT  LNR,    LNAME,    LSTATUS,    ORT
        INTO  :LNR,  :LNAME,  :LSTATUS,  :ORT
        FROM  L
        WHERE ORT = 'BERLIN'
```

L1	NEUMANN	30	BERLIN
L4	MEIER	10	BERLIN



Lösung

```
DECLARE xyz CURSOR FOR
  SELECT  LNR,  LNAME,  LSTATUS,  ORT
  FROM    L
  WHERE   ORT = 'BERLIN'
```

...

...

```
OPEN xyz
```

...

...

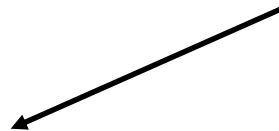
```
FETCH xyz INTO :LNR, :LNAME, :LSTATUS, :ORT (Loop)
```

...

```
IF "ENDE" (+100)
```

```
CLOSE xyz
```

LNR	LNAME	LSTATUS	ORT
L1	NEUMANN	30	BERLIN
L4	MEIER	10	BERLIN



- Unter der Cursor-Deklaration für den Cursor xyz steht der SELECT.
- Vor dem Zugriff auf die erste Zeile wird der Cursor geöffnet.
- Pro Fetch wird 1 Zeile übergeben (auch n Zeilen sind möglich: „multi-row-fetch“)
- Das Ende der Liste erkennt man am SQL-Code (+100).
- Nach der letzten Zeile wird der Cursor geschlossen.

Current-Zeile

- Durch den Befehl Fetch zeigt der Cursor in der Ergebnistabelle auf die aktuelle Zeile.
- Nach dem Fetch stehen die Daten (die gelesene Zeile) dem Anwendungsprogramm zur Verfügung.
- Die Daten können geprüft werden.
- Die Zeile kann auch verändert werden.

Current-Zeile verändern

- Befehl:

```
UPDATE    tabellename
SET       spalte = Ausdruck
          [, spalte = Ausdruck] ...
WHERE CURRENT OF cursor-name
```

- Erläuterung:
 - Die aktuelle Zeile wird verändert.
 - Der WHERE-Teil identifiziert den Cursor, der auf die zu verändernde Zeile zeigt.
 - Nächstes `_Fetch_` ändert den Cursor.

Current-Zeile löschen

- Befehl:

```
DELETE FROM tabellename  
WHERE CURRENT OF cursor-name
```

- Erläuterung:

- Die aktuelle Zeile wird gelöscht.
- Der WHERE-Teil identifiziert den Cursor, der auf die zu löschende Zeile zeigt.
- Aber wo steht der Cursor?

weitere Angaben bei declare cursor

- FOR UPDATE OF spaltenname
- DISTINCT
- ORDER BY / GROUP BY / HAVING
- UNION / JOIN
- built-in-Funktion
- FROM
- FOR FETCH / READ ONLY
- WITH HOLD
- WITH RR, RS, CS, UR
- OPTIMIZE FOR n ROWS

weitere Angaben bei declare cursor – Einschränkungen

- FOR UPDATE geht nicht gemeinsam mit ORDER BY
- FOR UPDATE geht nicht gemeinsam mit FOR FETCH / READ ONLY
- WITH HOLD hält die Positionierung auch nach einem Commit

- **FETCH FIRST n ROWS ONLY**
 - dann Achtung bei Sortierung!
 - Hinweis: Auch bei einfachem SELECT möglich!!
- **FETCH BEFORE / AFTER**
- **FETCH PRIOR (!)**
- **FETCH SENSITIVE / INSENSITIVE**
 - beachtet Löcher durch DELETE
- **FETCH NEXT ROWSET**
 - Rowset-Verarbeitung / multi-row-fetch

CURSOR

gemischt

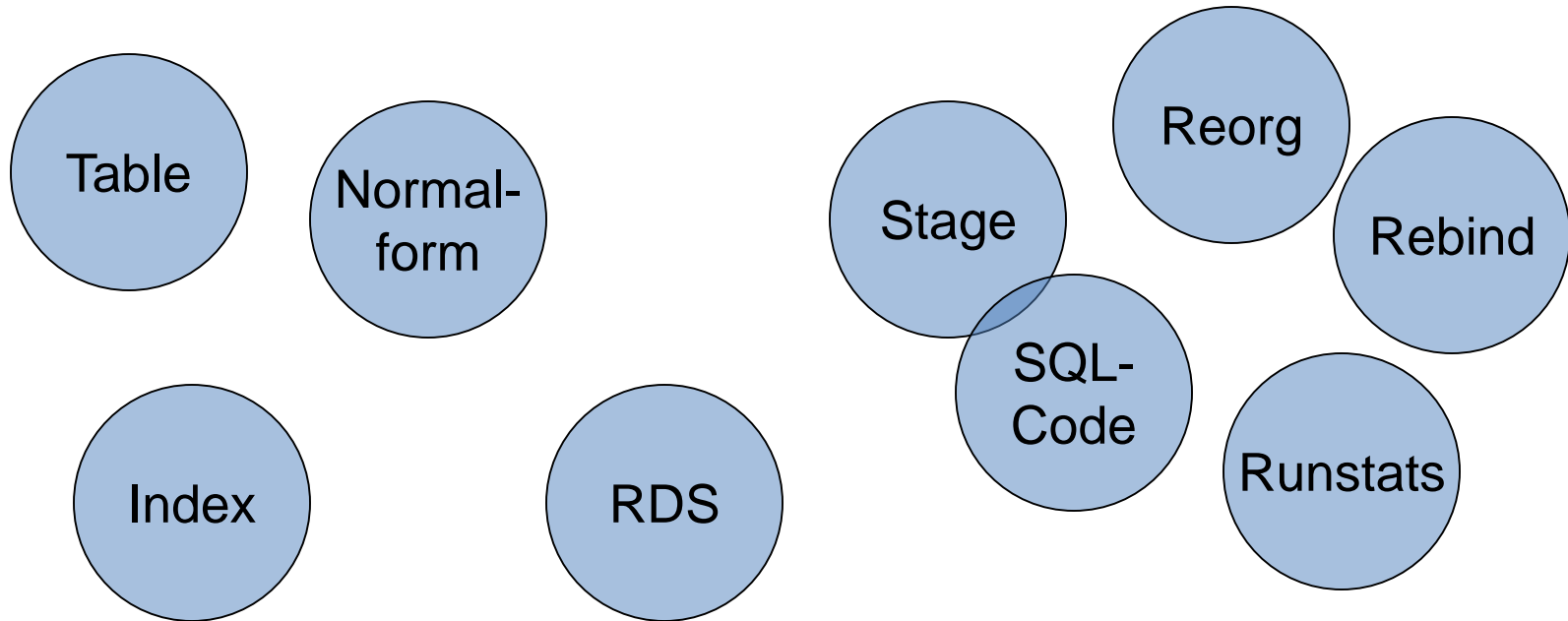
Übung(en)

- Kapitel 11.1 Lesen 50 Zeilen
- Kapitel 11.2 Lesen und Update 1 Zeile
- Kapitel 11.3 multi-row-fetch



-
- DB2-Systemkatalog
 - DB2-Utilities
 - SQL im Anwendungsprogramm
 - Cursor-Verarbeitung
 - • Explain und Performance

Begriffe



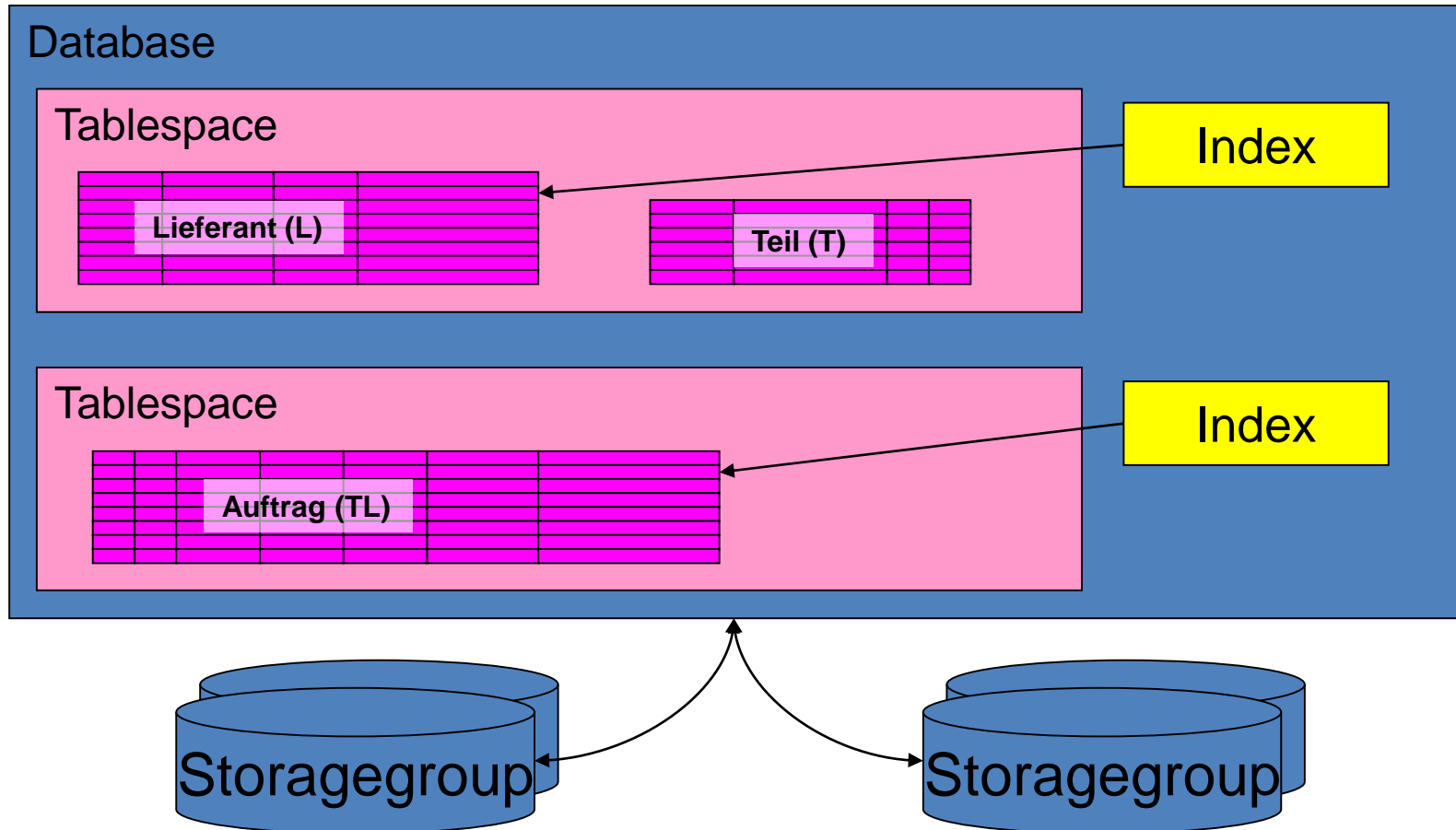
warum und wozu?

- Ziel der Programmierung ist es, ein optimales Programm und daher optimale SQL-Zugriffe zu kodieren.
- Performance ist wichtig
- schlechte Performance kostet Geld
- Doch wie kommt man zu einer optimalen Anwendung?
- Wann sind welche Faktoren zu berücksichtigen?
- Und: Was habe *ich* damit zu tun?

Komplexität des DB2

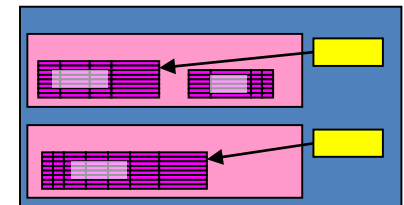
- DB2 in sich sehr komplex
- mehrere verschiedene Buffer Pools
 - BP2-Pool für Daten
 - BP3-Pool für Indexes
- viele DB-Objekte wie
 - Tablespace, Table, View, Index ...
- Umgang damit – KISS ist (lebens)notwendig
- Zitat Einstein: “Alles sollte so einfach wie möglich sein, aber nicht noch einfacher.”

DB2-Objekte – Klassifizierung



- Tablespace (Tabellenraum)
 - ist ein DB2-interner Name für einen oder mehrere VSAM-Dateien zur Speicherung der Daten
 - enthält die Daten von einer oder mehreren Tabellen
 - ist unterteilt in Pages einer Größe von 4k oder 32k
 - wird auf der Platte immer in 4k-VSAM-CIs gespeichert
- Indexspace
 - ist die Speicherform des Index
 - wird implizit beim CREATE INDEX angelegt

- Index
 - Es können und dürfen (beliebig) viele Indexe definiert werden.
 - 1 Index ist verantwortlich für Reihenfolge im Tablespace – Clustering Index.
 - Wahl des Index = Frage nach (fachlichen) Zugriffen



Ziel von DB2 und SQL

- kodieren des WAS nicht des WIE 😊
- Aber:
Modellierung⁽¹⁾, Wartung^(1,2,3) und Zugriff⁽²⁾ haben großen Einfluss auf das WIE. ☹️

⁽¹⁾ Datenmodell, Aufbau Tabellen, Aufbau Indizes

⁽²⁾ SQL

⁽³⁾ Änderung von Datenmengen, Art der Daten, Art der Abfragen ...

the 5 horsemen of performance

- Modellierung der Tabellen
- passende Nutzung der Runstats
- geeignete Nutzung von Reorgs
- angemessene Nutzung der Indexe
- richtiges Kodieren der SQLs

Modellierung der Tabellen

- Normalisierung – Design-Qualität
- 1. Normalform
- 2. Normalform
- **3. Normalform**
- 4. Normalform
- 5. Normalform

every entity depends
on the key
the whole key
and nothing but the key

- Statistik zu einer Tabelle
Beispiel:
 - Anzahl der Zeilen
 - letzter Runstats
 - Anzahl pages
 - Anzahl indexpages
 - etc.
 - also alles, was ein Optimizer für seinen Zugriff braucht.

geeignete Nutzung von Reorgs

- Reorg heißt u.a.
 - Neuaufbau der Tabelle
 - Neuaufbau des Index (Clustering)
- Ziel (denke an VSAM ;-)
 - leere Bereiche füllen
 - Überlaufbereiche neu anlegen
 - etc.
- Folgerung:
regelmäßig Reorg durchführen
... spätestens wenn Clusterratio <95%

angemessene Nutzung der Indexe

- Zugriff muss durch Index unterstützt werden
 - Ausnahme: Minitabellen
- Ergebnis:
 - Tablespacescan wird vermieden
 - Non-matching Indexscan wird vermieden
 - oft werden interne Sorts nicht mehr benötigt
 - ascending / descending – ab V8 automatisch
- wichtigsten Index clustern
 - also nicht immer den primary index!

- Ist das wirklich wichtig?
- Beispiel:
 - Briefträger ist ein INSERT-Operator
 - Straße ist die Tabelle
 - Briefkästen sind die Pages der Tabelle, in die eingefügt werden soll
 - Sortierung nach Name ... ☹️
 - Sortierung nach Straße und Hausnummer ... 😊

- Can you KISS the SQL?
 - KISS: keep it simple and stupid
- Predicates korrekt und schnell kodiert?
 - >, <
 - Umformatierung notwendig?
- Subqueries wirklich notwendig?
 - IN ist schneller
- Gibt es (komplexe) Funktionen?
 - kostet CPU; besser im Programm?

- Keine Input-Variable in den Predicates?
 - beim BIND keine Werte bekannt!
- Gibt es Korrelationen zwischen 2 Tabellen?
 - bei abhängigen Veränderungen kann der ACCESS-Pfad nicht vordefiniert werden.
- Berechnungen sinnvoll kodiert?
 - WHERE $S + (:h * S) > 50$
 - WHERE $S > 50 / (1 + :h)$

- Materialisierung sinnvoll?
 - bei großen Datenmengen kann es sinnvoll sein, vorab Daten zu lesen.
- Gibt es verschlüsselte Daten?
 - Encryption / Decryption kostet Zeit
- Isolation Level korrekt?
 - Kann evtl. UR benutzt werden?
- Cursor mit guter OPTIMIZE Klausel?
 - OPTIMIZE FOR n ROWS
- etc. siehe Kurs PROP (Programm- und Ressourcen-Optimiertes Programmieren)

richtiges Kodieren der SQLs – etc.

- siehe Kurs PROP (Programm- und Ressourcen-Optimiertes Programmieren)
- ! • siehe Kurs SQL-Performance von Hr. Denne
- Übrigens: Die SQL-Reference von IBM ist inzwischen mehr als 20 MB groß! ☹ ☹ ☹

Unterstützung durch Explain

- Explain gibt standardisierte Informationen zu dem Zustand eines DBRM / Package / Plan.
- Betrachtet wird der Zugriffspfad für die einzelnen SQLs.
- Anstoß über Option EXPLAIN=YES oder EXPLAIN=ALL beim Bind des Package.
 - Alternativ kann auch im Programm der Explain als SQL-Befehl eingefügt werden. Dies sollte aber im Normalfall unterbleiben!

Explain-Daten

- Die Informationen werden in Explain-Tabellen abgelegt.
- Explain-Tabellen können abgefragt werden
 - Visual Explain
 - DB2-Commands `dynexpln`, `db2expln`, `db2exfmt`
 - tolle DB2-Tools
 - DB2 Catalog Manager von BMC

Zugriffsplan und Optimierung – 1

- Bei der Kompilierung einer SQL-Anweisung, schätzt der DB2-Optimizer den Ausführungsaufwand der verschiedenen Methoden ab, die die Anforderung erfüllen würden.
- Auf der Grundlage dieser Abschätzung wählt der DB2-Optimizer den Zugriffsplan aus, den es für optimal hält. Ein Zugriffsplan gibt die Reihenfolge von Operationen an, die erforderlich sind, um eine SQL-Anweisung auszuführen.

Zugriffsplan und Optimierung – 2

- Wenn ein Anwendungsprogramm gebunden wird, wird ein Package erstellt.
- Dieses Package enthält Zugriffspläne für alle statischen SQL-Anweisungen in dem entsprechenden Anwendungsprogramm.
- Die Zugriffspläne für dynamische SQL-Anweisungen werden zum Zeitpunkt der Ausführung der Anwendung erstellt.

Runstats

- Für die Erstellung eines effizienten Zugriffplans müssen die Tabellenstatistiken aktualisiert sein.
 - Befehl: RUNSTATS als DB2-Command
- Erforderlich, wenn Änderungen in der DB2-Anwendungsumgebung gemacht wurden wie
 - Table, View, Index
 - größere Datenmengenänderung
- DBAs kümmern sich darum

Stage1, Stage2, Indexable – 1

- Die erforderlichen Daten werden an verschiedenen Stellen in DB2 gefiltert.
- vor dem Zugriff auf die Index-Leaf-Pages
- nach dem Zugriff auf die Index-Leaf-Pages, aber vor dem Daten-Zugriff
- vor der Rückgabe von Datenmanager an die Relational Data Services
- vor der Rückgabe von den Relational Data Services an den Anforderer

- Indexable
 - Die Daten werden über den Index gesucht und direkt gelesen.
- Index-Screening
 - filtern innerhalb Durchsuchen des Index
- Stage1-Prädikate
 - filtern der Daten im Datenmanager
- Stage2-Prädikate
 - filtern der Daten bei Relational Data Services

Zugriffswege (Bezeichnungen des Explain) – 1

- TBSCAN Tablespacescan
- IXSCAN Indexscan
- IXAND zwei Indexscan mit AND
- MSJOIN Merge und Scan Join
- NLJOIN Nested Loop Join
- HSJOIN Hash Join
- RIDSCN Record ID Scan
- SORT sortieren
- TEMP aufbauen temporäre Tabelle
- UNION verbinden von Erg.mengen

Zugriffswege (Bezeichnungen des Explain) – 2

- UNIQUE eliminieren von Tupeln
- FILTER filtern von Tab / Erg.mengen
- FETCH lesen
- GRPBY gruppieren
- DELETE löschen von Tupeln
- INSERT einfügen von Tupeln
- UPDATE ändern von Tupeln / Spalten
- RETURN Ergebnismenge
- STAR JOIN Sternverknüpfung

- es kann kein Index benutzt werden
- Indexsuche ist aufwändig wegen / wenn
 - Table zu klein
 - Grad der Index-Clusterung ist gering
 - Es wird der größte Teil der Tabelle gelesen.
- entspricht sequentiellm Lesen (aller Daten)

Zugriffswege – IXSCAN – Arten – 1

- Matching Index Scan
ACCESSSTYPE = 'I', MATCHCOLS > 0
 - lesen auf höchstem Index-Niveau
- Equal Unique Index Access
ACCESSSTYPE = 'I', MATCHCOLS = #ind-sp
 - fast schnellster Zugriff
- IN-List Index / ACCESSSTYPE = 'N'
 - Index-Scan, wenn IN kodiert wurde

Zugriffswege – IXSCAN – Arten – 2

- Non-Matching Index Scan
ACCESSSTYPE = 'I' / MATCHCOLS = 0
 - höchste Index-Ebene kann nicht benutzt werden
- One-Fetch-Index Scan / ACCESSSTYPE = 'I1'
 - Direktzugriff mit MIN-Funktion
- Index only / INDEXONLY = 'Y'
 - Alle Daten aus Index heraus ableitbar.

Zugriffswege – komplexe Zugriffswege – 1

- Nested Loop Join / METHOD = 1
 - sucht für jede Zeile aus Tab1 Zeilen aus Tab2
- Merge Scan Join / METHOD = 2
 - mindestens 1 Tabelle wird vorgefiltert, evtl. sortiert und zwischengespeichert (materialisiert); danach werden Daten gemergt
- Hybrid Join / METHOD = 4
 - komplexe Methode, die Duplikate ausfiltert; es wird immer materialisiert
- UNION
 - erzeugt mehrere Zwischentabellen

Zugriffswege – komplexe Zugriffswege – 2

- Nested Table Expression
 - “dynamischer View” erzeugt Zwischentabelle, die evtl. direkt in Resulttabelle übertragen wird
- Subquery / QBLOCKNO > 1
 - Daten werden variabel ausgefiltert
 - Non-Correlated Subquery
 - Tab1 wird einmalig durchsucht und gegen Tab2 geprüft
 - Correlated Subquery
 - Übereinstimmungen werden pro Zeile gesucht

Zugriffswege – komplexe Zugriffswege – 3

- Multiple Index Scan / `ACCESSTYPE = 'M'`
 - mehrere Indexe werden benutzt
 - Kandidaten gefiltert (RID-Liste) und anschließend mit AND / OR verknüpft; danach werden Daten-Pages gelesen
- Sortierung / `METHOD = 3` / `SORTxx = 'Y'`
 - sortiert werden Workfile oder RID-Liste
- Unterstützung durch DBAs!

-
- DB2-Systemkatalog
 - DB2-Utilities
 - SQL im Anwendungsprogramm
 - Cursor-Verarbeitung
 - Explain und Performance

