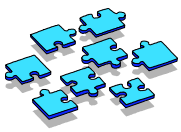
Neuerungen in SQL2003

Peter Pistor
Heidelberg, 24.06.2003



Bücher zu SQL99 und SQL2003

- Deutschsprachige Bücher
  - W.Panny, A.Taudes, *Einführung in den Sprachkern von SQL-99*, Springer, 2000.
  - G.Saake, W.Sattler, *Datenbanken & Java (JDBC, SQLJ und ODMG)*, dpunkt, 2000.
  - C.Türker, *SQL:1999 & SQL:2003 (Objekt-relationales SQL, SQLJ & SQL/XML)*, dpunkt, 2003.
  - D.Petkovic, *SQL objektorientiert*, Addison-Wesley, 2003.

Englische Bücher zu SQL99

- J.Melton, A.R.Simon, *SQL:1999 (Understanding Relational Language Components)*, Morgan Kaufmann, 2002.
- J.Melton, *Advanced SQL:1999*, Morgan Kaufmann, 2003.

SQL-Norm: Übersicht

- Teil 1: SQL/Framework
- Teil 2: SQL/Foundation
- Teil 3: SQL/CLI
- Teil 4: SQL/PSM
- Teil 9*: SQL/MED
- Teil 10 : SQL/SCHEMATA
- Teil 11*: SQL/OLB
- Teil 13*: SQL/JRT
- Teil 14 : SQL/XML (neuer Teilstandard)

(*: bereits Teil von SQL99, aber erst 2000 verabschiedet)

Teil 1: SQL/Framework

- Gemeinsame Definitionen und Konzepte
- Allgemeine Aussagen zur *conformance*
- Etwa 85 Seiten

Teil 2: SQL/Foundation

- Zentraler Teil der Norm
- *Information & Definition Schema* ausgegliedert (Teil 11)
- Spracheinbettung (außer Java), dynamisches SQL
- Prozedur-Schnittstelle: Teil 3
- **Traditionelles SQL** *und*
- **Objektorientiertes SQL**
- Etwa 1300 Seiten

Teil 3: SQL/CLI

- **CLI** = *Call-Level Interface*
- Bekannteste Implementierung: ODBC
- Änderungen gegenüber CLI99: Anpassung an Änderungen in SQL2003 und in ODBC 3.0
- Etwa 400 Seiten

Teil 4: SQL/PSM

- **PSM** = *Persistent Stored Modules*
- Imperative Sprachkonstrukte; Berechnungsvollständigkeit
- Entsprechende (aber weitgehend nicht normgerechte) Produkte: to PL/SQL, Transact-SQL, ...
- **Das** Vehikel zur Definition des Verhaltens nutzerdefinierter strukturierter Typen mit SQL-Mitteln
- Etwa 170 Seiten

Teil 9: SQL/MED

- **MED** = *Management of External Data*
- Einbindung externer Dateien über **DATALINKs** (Angereicherte URLs)
  - Spalten-Optionen für *recovery*, Zugangs- Kontrolle und Integritäts-Überwachung
  - Anwendung (z.B.): Verwaltung von Bild-Archiven
- Maskierung **externer Daten als SQL-Tabellen**:
  - *wrappers*: Satz von 3GL-Routinen
  - *foreign servers:* Definiert über *wrappers*
  - "foreign table": Definiert ü. einen "foreign server"
  - Externe Daten: Dateien, Mess-Fühler, ..., Daten-Banken (spez.: SQL-DBMS)
- Etwa 500 Seiten

Teil 10: SQL/OLB

- **OLB** = *Object Language Bindings*
- Einbettung von SQL-Code in Java
- Entstanden aus SQLJ Teil 0
  - Basiert auf aktuellem SQL (SQLJ: SQL92)
  - OLB: höhere Qualität!
- Etwa 360 Seiten

Teil 11: SQL/Schemata

- Legt Katalog-Strukturen und Inhalte fest
- Beschreibt Objekte einer Datenbank (z.B. Tabellen u. Sichten, Typen, Routinen, Trigger, Zeichensätze)
- Etwa 300 Seiten

Teil 13: SQL/JRT

- **JRT** = *Java Routines and Types*
- Java Routinen:
  - Schemaobjekte der SQL Datenbank
  - aufrufbar aus SQL-Anweisungen
- Java-Klassen dienen als Datentypen (z.B. bei Spalten von SQL-Tabellen)
- Basiert auf SQLJ Part 1
- Etwa 200 Seiten

Teil 14 (neu): SQL/XML

- Neuer Basis-Datentyp: XML
- Abbildungen zwischen SQL und XML
- Operationen zum Erstellen und Bearbeiten von XML-Dokumenten
- Sehr aktiver Teil der SQL-Norm; voraussichtlich ein Schwerpunkt der Normungsarbeit der nächsten Jahre
- Etwa 280 Seiten

SQL92 -> SQL99: was war neu?

- Relationales Modell:
  - Rekursive Anfragen
  - Konstrukte zur Verdichung-Analyse von Daten (OLAP)
- Aktive Datenbanken: Trigger
- Typsystem:
  - Basis-Datentyp: Boolean
  - Konstruierte Typen: ARRAY, ROW, REF
  - *distinct types*: verkapselte Basis-Datentypen
  - Strukturierte Typen: Verkapselung, Vererbung
- Tabellen:
  - *"object IDs"*; Tabellen-Hierarchien

SQL92 -> SQL99: was war neu?

- Authorisierungs-Modell: Rollen-Konzept
- Einbettung in Java (SQL/OLB)
- Nutzung von Java-Routinen und Klassen in SQL (SQL/JRT)
- Einbindung externer Daten (MED)

SQL99 -> SQL2003: was ist neu?

- Fehlerkorrekturen (alle Teile)
- Neue Konstrukte in SQL/Foundation
- Neuer Teil: SQL/XML
- Berücksichtigung von Änderungen im Teil *Foundation* in übrigen Teilen
- Fazit: (bis auf SQL/XML) konservative Weiterentwicklung
- Thema heute: **SQL/Foundation**

Neues in SQL/Foundation

- Verbesserte Untersützung von Kollektionen; spez.: MULTISET
- Neuer Basis-Datentyp: BIGINT
- Neue Prädikate
- Neue OLAP-Funktion: TABLESAMPLE
- Generierte Spalten
- Sequenzgeneratoren, Identitätsspalten
- MERGE-Anweisung

Kollektionen in SQL99

- Nur ARRAYs
  - Abfragbar mit SFW; Ergebnis: Tabelle
  - aber: Tabelle->ARRAY fast unmöglich
  - Übliche ARRAY-Operationen: Element-Zugriff, Verkettung, Länge, Konstruktoren.
  - Aber: Kein *update in place*
- Keine (Multi-)Mengen unterstützt
- Tabellen außerhalb des Typkonzepts; Tabelle als Funktionsergebnis verboten

Multiset in SQL2003

- Konstruktor; Beispiele:
  - MULTISET[ ]: leere Menge
  - MULTISET[NULL, 1, 1, 3]
- Umwandlung eines Abfrageergebnisses in eine Multi-Menge
  - MULTISET(SELECT .. FROM .. WHERE)
  - Vereinfacht Zuweisung Tabelle->Kollektion
- Vereinigung, Durchschnitt:
  - M1 MULTISET UNION M2
  - M1 MULTISET INTERSECTION M2

Multiset in SQL2003

- Kardinalität:
  - CARDINALITY(M1)
- Anfragen:

  SELECT p FROM **UNNEST (myPrimes) PP(p)**
  WHERE P < 13
  - UNNEST erzeugt temporäre Tabelle
- Mengenorientierte Prädikate:
  - 2 MEMBER myPrimes -- was bei NULL-Elementen?
  - myPrimes IS A SET
  - myPrimes SUBMULTISET yourPrimes
- "Strippen" einelementiger Mengen
  - ELEMENT (MULTISET[1]) -- ist der Skalar 1

Multiset in SQL2003: Aggregatfunktionen

- COLLECT:
  - Erzeugt aus der durch GROUP BY erzeugten temporären Gruppe eine Multimenge
- FUSION:
  - Erzeugt aus einer durch GROUP BY erzeugten temporären Gruppe von Multimengen deren Vereinigung
- INTERSECTION:
  - Erzeugt aus einer durch GROUP BY erzeugten temporären Gruppe von Multimengen deren Schnitt

Generierte Spalten

- Normale Spalten in Basistabellen: *base columns*
- Generierte Spalten: Werte aus einer/mehreren Basisspalten der gleichen Zeile berechnet, z.B.

```
CREATE TABLE EMPLOYEES
( EMP_ID INTEGER,
  SALARY DECIMAL(7,2),
  BONUS DECIMAL(7,2),
  TOTAL_COMP GENERATED ALWAYS AS
    ( SALARY + BONUS ),
  HR_CLERK GENERATED ALWAYS AS
    ( CURRENT_USER )                    )
```

- Ersatz für Trigger oder gescheite Zugriffspfad-unterstützung

Sequenzgeneratoren

- Erzeugt Sequenzen numerischer Werte, z.B. für Vergabe von Schlüsselwerten
- Starting value
- *Increment* (positiv: aufsteigend, negativ: absteigend)
- Minimal- und Maximal-Werte
- *Cycle*-Option
- Extern (explizit definiertes Schemaobjekt) or intern (Teil eines anderen Objekts, z.B. einer Identitätsspalte)

Sequenzgeneratoren

```
CREATE SEQUENCE mySeq AS INTEGER
START WITH 0
INCREMENT BY 1
MAXVALUE 100
CYCLE
```

- Andere Optionen:
  - NO CYCLE
  - NO MAXVALUE, MINVALUE, NO MINVALUE
- Vergabe von Werten **nicht** unter TX-Kontrolle; trotzdem im Zyklus keine Duplikate

Sequenzgeneratoren: Beispiele

- INSERT INTO TBL (COL1, COL2)
  VALUES (10,
  NEXT VALUE FOR mySeq)
- CALL myproc (**NEXT VALUE FOR** mySeq)
- SET J = J + **NEXT VALUE FOR** mySeq

Identitätsspalten: Beispiel

```
CREATE TABLE employees (
  EMP_ID INTEGER
    GENERATED ALWAYS AS IDENTITY
    START WITH 100
    INCREMENT 1
    MINVALUE 10
    NO MAXVALUE
    NO CYCLE,
  SALARY DECIMAL(7,2), ...   )
```

- Syntax wie bei Sequenzgenerator!

Identitätsspalten

- Maximal 1 Identitätsspalte/Basistabelle
- Enthält keine Nullwerte
- INSERT
  - GENERATED ALWAYS: Für die I-Spalte nur DEFAULT oder aber nichts vorgeben
  - GENERATED BY DEFAULT: Für diese Spalte einen Wert vorgeben; andernfalls wird der Wert generiert
- UPDATE: nur bei GENERATED BY DEFAULT erlaubt

MERGE-Operation

- Zweck: Abgleich einer Zieltabelle Z mit einer Quell-Tabelle Q
  - Zeilen in Z mit genau einer passenden Zeile in Q werden per UPDATE verändert
  - Zeilen in Q ohne Entsprechung in Z werden in Z eingefügt
- Vorteile: effizienter ausführbar und leichter verständlich als Spezifikation über 2 Anweisungen

MERGE-Operation

```
MERGE INTO archive Z
  USING (SELECT activity, descr FROM activities) Q
    ON (Z.activity = Q.activity)
WHEN MATCHED THEN
  UPDATE SET Z.description = Q.description
WHEN NOT MATCHED THEN
  INSERT (description, activity) VALUES(Q.description,
Q.activity)
```

- Mindestens 1 MATCH-Klausel
- Bei 2 Klauseln: Reihenfolge bestimmt, ob UPDATE oder INSERT-Trigger zuerst feuert

Neue OLAP-Funktion: TABLESAMPLE

- Zieht Stichproben aus großen Tabellen
- Syntax:

  TABLESAMPLE method (percentage) [repeatable]

- *method* **Bernoulli**: Stichprobe enthält etwa *percentage* % der Zeilen der Originaltabelle; Wahrscheinlichkeit einer Zeile in Ergebnis: *percentage* %; Auswahl einer Zeile unabhängig von den bereits ausgwählten Zeilen
- *method* **System**: Wie oben, jedoch keine Aussage bzgl. Unabhängigkeit der Zeilen

Kleinkram

- Neuer Datentyp: BIGINT
- Neues Prädikat: NORMALIZED
  - string_value IS [NOT] NORMALIZED
  - *Character set* von *string_value* muss UTF8, UTF16 oder UTF32 sein
  - Ergebnis ist true, wenn *string_value* gemäß Unicode 15 normalisiert ist

Konformität mit SQL2003

- *Core* SQL99
  - *Entry* SQL92
  - + Viel aus *Intermediate* SQL92
  - + Etwas aus *Full* SQL92
  - + Einige neue SQL99 *features*
- *Packages* & *Parts*
- *Core* SQL2003 und *Core* SQL99 identisch; **keiner hat es bisher**
- Markt entscheidet, was aus *Core* und aus den Packages angeboten wird, und was SQL20?? bringen wird