Working with DB2 UDB Objects Part 5 of 6

Tobias Reichelt

Martin-Luther-Universität Halle-Wittenberg

21. Juli 2006

Inhaltsverzeichnis

- Quellen
  - Working with DB2 UDB Objects: DB2 V8.1 Family Fundamentals certification prep, Part 5 of 6
    http://www-128.ibm.com/developerworks/edu/i-dw-db2-cert5v8-i.html

- große und flexible Ansammlung von Datentypen in DB2
- Basisdatentypen wie INTEGER, CHAR und DATE
- Möglichkeit zur Erstellung nutzerdefinierter Datentypen (UDTs) für komplexe und nicht traditionelle Anwendungsfälle
- eingebaute Datentypen (Built-in):
  Numerisch, String, Datetime, Datalink
- nutzerdefinierte Datentypen (User-defined):
  distinct, structured, reference type

Numerische Datentypen

- Integer zur Speicherung von ganzen Zahlen:
  SMALLINT: -2^{15} bis $2^{15}-1$ (2 Byte)
  INTEGER: -2^{31} bis $2^{31}-1$ (4 Byte)
  BIGINT: -2^{63} bis $2^{63}-1$ (8 Byte)
- Decimal für gebrochene Zahlen:
  Gesamtanzahl der Ziffern p, s davon hinter dem Komma
  DECIMAL(p, s) in $(p/2+1)$ Byte
- Float für Näherungswerte bestimmter Genauigkeit:
  REAL mit 1 bis 24 Ziffern (4 Byte)
  DOUBLE mit 25 bis 53 Ziffern (8 Byte)

String Datentypen

- Single-Byte Character Strings:
  CHAR für Zeichenketten fixer Länge bis zu 254 Byte
  VARCHAR für Zeichenketten variabler Länge bis 32 kB
- Double-Byte Character Strings:
  GRAPHIC für Zeichenketten fixer Länge bis 127 Zeichen
  VARGRAPHIC für variable Länge bis 16336 Zeichen
- Datentypen für sehr lange Zeichenketten bis 2GB
  LONG VARCHAR, CLOB (Character Large Object), LONG VARGRAPHIC, DBCLOB (Double-Byte CLOB), BLOB (Binary Large Object)

  Nicht physisch in der DB enthalten, d.h. zusätzliche Verarbeitung bei Datenzugriff notwendig

Datums- und Zeitdatentypen

- DATE, TIME: Darstellung abhängig vom Ländercode, der bei der Erstellung der Datenbank angegeben wird
- TIMESTAMP: yyyy-mm-dd-hh.mm.ss.nnnnnn
- als Zeichenkette repräsentiert, bei Updates in Anführungszeichen zu setzen
- in der Datenbank in internem Format gespeichert
- eingebaute Funktionen zur Manipulation: DAYOFWEEK, DAYNAME, DAYS, ...
- spezielle Register zur Erstellung der aktuellen Werte, basierend auf Systemdaten (CURRENT DATE)

Datalink Datentypen

- Datentyp zur Verwaltung externer Dateien, Dateien können im gleichen Dateisystem, auf dem gleichen Server oder auch auf einem entfernten Server liegen
- Speicherung einer Referenz in der DB, die von Anwendungen für sicheren Zugriff genutzt werden kann
- Einfügen über eingebaute Funktion DLVALUE, Parameter wie den Namen und Speicherort
- Abfrage über mehrere eingebaute Funktionen (abhängig von den Informationen, die angefragt werden)

Nutzerdefinierte Datentypen

- Distinct Types
  - neuer Datentyp, basierend auf eingebauten Datentypen
  - erbt dessen Eigenschaften
  - zur Sicherung, dass nur Werte desselben Typs verglichen werden können
- CREATE DISTINCT TYPE CANDOL AS DECIMAL(10, 2) WITH COMPARISONS
- CREATE DISTINCT TYPE USADOL AS DECIMAL(10, 2) WITH COMPARISONS
  - basieren beide auf dem eingebauten numerischen Typ DECIMAL(10, 2), dennoch kein Vergleich ohne Konvertierungsfunktion möglich

Nutzerdefinierte Datentypen

- Distinct Types
  - DB2 erstellt automatisch Konvertierungsfunktionen zur Konvertierung zwischen Basis- und Nutzerdatentypen
- CREATE TABLE ITEMS (
  ITEMID CHAR(5),
  PRICE CANDOL
)
- INSERT INTO ITEMS VALUES('ABC11', CANDOL(30.50))
  - Konvertierung von DECIMAL(30.50) in CANDOL, damit INSERT erfolgreich

Nutzerdefinierte Datentypen

- Structured Types
  - bestehen aus mehreren Spalten von eingebauten Typen z.B. Adresse aus Straße, Stadt, PLZ
  - können auch hierarchisch strukturierte Untertypen haben
  - somit können hierarchisch angeordnete Objekte in der DB gespeichert werden
- Reference Types
  - Referenzen auf Zeilen anderer Tabellen möglich, aber keine Sicherstellung der Beziehung zu der Tabelle (Referentielle Integrität wird durch Constraints sichergestellt)

DB2 Extenders

- Unterstützung für komplexe, nicht traditionelle Datentypen
- implementiert mit nutzerdefinierten Datentypen (UDT) und Funktionen (UDF) zur Manipulation der Daten
- danach nutzbar bei Erstellung von Tabellen (UDT) und Arbeit auf den Daten (UDF)
- Installation separat in jede DB, die sie nutzen können soll
- von IBM und unabhängigen Softwareentwicklern, z.B. zur direkten Speicherung von Video-, Audio-, XML-Daten in der DB

- alle Daten in Tabellen gespeichert, in Form von Zeilen
- beinhaltet eine oder mehrere Spalten mit zugehörigen Datentypen
- definiert mit CREATE TABLE oder GUI Tool von DB2

- DB enthält Menge von Tabellen, genannt *System Catalog* , mit Informationen zu allen Objekten der DB
- SYSCAT.TABLES : eine Zeile pro Tabelle
- SYSCAT.COLUMNS : eine Zeile pro Spalte jeder Tabelle
- keine Änderungen über INSERT/ UPDATE/ DELETE, nur automatisch nach Änderungen am DB-Schema (CREATE, ALTER, RUNSTATS)

- CREATE TABLE BOOKS (
  BOOKID INTEGER,
  BOOKNAME VARCHAR(100),
  ISBN CHAR(10)
)
- CREATE TABLE MYBOOKS LIKE BOOKS
- zweite Variante erzeugt eine neue Tabelle mit gleichem Relationsschema wie die erste Tabelle (oder Sicht), die Daten oder Constraints werden nicht kopiert
- Daten über INSERT einzeln oder kombiniert einfügen
- IMPORT nutzt INSERT für kleinere Datenmengen
- LOAD fügt Zeilen direkt in die Datenseiten ein (für große Datenmengen, schneller als IMPORT, da ganze Blöcke)

Exkurs Tablespaces

- Tabellen werden in Tablespaces gespeichert
- DB2 platziert Tabellen in Standard-Tablespace, wenn nicht explizit angegeben
- ```
  Create Table Books (
    Bookid Integer,
    Bookname Varchar(100),
    Isbn Char(10)
  ) In Bookinfo
  ```
- Einfluss auf Performance und Pflege der DB
  ```

- Eigenschaften lassen sich mit ALTER TABLE ändern:
  - Spalten hinzufügen
  - Primärschlüssel hinzufügen/ löschen
  - Constraints hinzufügen/ löschen
  - Länge von VARCHAR Spalten ändern
- ALTER TABLE BOOKS ADD BOOKTYPE CHAR(1)
- Spalten können nicht gelöscht werden, der Tablespace kann nicht verändert werden (solche Änderungen nur, wenn die Daten gesichert werden, die Tabelle gelöscht und neu angelegt wird)
- DROP TABLE löscht die Tabelle inkl. Daten/ Indexe/ Constraints

- zu Spalten können Restriktionen angegeben werden:
  NOT NULL, DEFAULT, GENERATED
- CREATE TABLE Books (
  Bookid Integer NOT NULL
  GENERATED ALWAYS AS IDENTITY
  (START WITH 1, INCREMENT BY 1),
  Bookname Varchar(100) WITH DEFAULT 'TBD',
  Isbn Char(10)
)
- GENERATED ALWAYS AS IDENTITY erzeugt eine eindeutige Nummer

- Generated Always berechnet automatisch Werte
- Create Table Authors (
  Authorid Integer Not Null Primary Key ,
  Lname Varchar(100),
  Fname Varchar(100),
  Fictionbooks Integer,
  Nonfictionbooks Integer,
  Totalbooks Integer Generated Always As
  (Fictionbooks + Nonfictionbooks)
)

- Integritätsbedingungen, die in jedem Datenbankzustand erfüllt sein müssen
- vom DB-Manager überwacht/ durchgesetzt
- können innerhalb CREATE TABLE oder später mit ALTER TABLE erstellt werden

- 3 Arten
  - Unique Constraints
  - Referential Integrity Constraints
  - Table Check Constraints

Unique Constraints

- erzwingen eindeutige Werte in einer oder mehreren Spalten, jede davon muss NOT NULL sein
- definiert über UNIQUE oder PRIMARY KEY
- nur ein Primärschlüssel pro Tabelle erlaubt, aber mehrere eindeutige Spalten
- PRIMARY KEY, wenn in anderen Tabellen darauf zugegriffen wird, sonst UNIQUE (alternative Schlüssel)
- um die Eindeutigkeit zu erzwingen erstellt DB2 automatisch einen "Unique Index"

Unique Constraints

- optional einen Namen angeben, zur besseren Verwaltung über den Systemkatalog
- ```
  Create Table Books (
    Bookid Integer Not Null Primary Key,
    Bookname Varchar(100),
    Isbn Char(10) Not Null
      Constraint Booksisbn Unique
  )
  ```
- keine doppelten Indexe oder Constraints erlaubt, z.B. Alter Table Books Add Constraint Unique(Bookid) verweigert, da bereits ein Constraint über Bookid wg. Primärschlüssel
  ```

Referential Integrity Constraints

- definieren Beziehungen zwischen Tabellen
- CREATE TABLE AUTHORS (
  AUTHORID INTEGER NOT NULL PRIMARY KEY,
  LNAME VARCHAR(100),
  FNAME VARCHAR(100))
- CREATE TABLE BOOKS (
  BOOKID INTEGER NOT NULL PRIMARY KEY,
  BOOKNAME VARCHAR(100),
  ISBN CHAR(10),
  AUTHORID INTEGER REFERENCES AUTHORS)
- AUTHORS ist Elterntabelle, BOOKS abhängige Tabelle

Referential Integrity Constraints

- DB2 erzwingt die Einhaltung bei Updates
- nur gültige Werte in abhängige Tabelle eintragen: in Elterntabelle muss ein Tupel mit dem referenzierten Wert existieren
- Löschen aus der Elterntabelle: Restrict, No Action, Cascade, Set Null (wenn Tupel referenziert)
- Restrict, No Action: Löschen wird verweigert (Standard)
- Cascade: Löschen aller abhängigen Tupel
- Set Null: Fremdschlüsselwerte auf Null setzen
- Update des Schlüssels in Elterntabelle: Restrict, No Action

Table Check Constraints

- Einschränkung der Werte einer Spalte
- Definition in Create Table oder Alter Table
- Alter Table Books Add Booktype Char(1)
  Check (Booktype In ('F', 'N'))
- Modifikation nur durch löschen und neu erstellen
- durch DB2 bei Insert/ Update/ Delete sichergestellt

- Sichten sind virtuelle Tabellen, Daten aus Tabellen abgeleitet
- erlauben spezielle Anpassung an Nutzergruppen/ Anwendungen
- Beschränkung, welche Tupel/ Spalten gelesen oder aktualisiert werden dürfen
- Sichtdefinition basiert auf Tabellen oder anderen Sichten (verschachtelte Sicht), Spalten können dabei umbenannt werden
- SYSCAT.VIEWS : eine Zeile pro definierter Sicht
- SYSCAT.VIEWDEP : eine Zeile pro abhängige tabellenähnliche Objekte
- SYSCAT.TABLES und SYSCAT.COLUMNS analog Tabellen

- SELECT-Teil der Definition bestimmt, ob Sicht *read-only* oder *updatable*
- Spalte READONLY in SYSCAT.VIEWS gibt Auskunft
- WITH CHECK OPTION erlaubt es dem Nutzer nicht, Tupel zu aktualisieren/ einzufügen, die außerhalb der Darstellung der Sicht liegen (Bedingungen der Sichtdefinition müssen erfüllt sein)
- CREATE VIEW NONFICTIONBOOKS AS
  SELECT * FROM BOOKS WHERE BOOKTYPE = 'N'
  WITH CHECK OPTION
- INSERT INTO NONFICTIONBOOKS VALUES(..., 'F') wird verweigert

- bei geschachtelten Sichten Angabe von CASCADED (Standard) oder LOCAL möglich
- CASCADED: alle Bedingungen, jeder Ebene, müssen erfüllt sein
- CREATE VIEW NONFICTIONBOOKS AS
  SELECT * FROM BOOKS WHERE BOOKTYPE = 'N'
- CREATE VIEW NONFICTIONBOOKS1 AS
  SELECT * FROM NONFICTIONBOOKS
  WHERE BOOKID > 100
  WITH CASCADED CHECK OPTION
- INSERT INTO NONFICTIONBOOKS1 VALUES(10, ..., 'F') wird verweigert

- LOCAL: die Bedingungen der so spezifizierten Sicht müssen erfüllt sein
- CREATE VIEW NONFICTIONBOOKS AS
  SELECT * FROM BOOKS WHERE BOOKTYPE = 'N'
- CREATE VIEW NONFICTIONBOOKS2 AS
  SELECT * FROM NONFICTIONBOOKS
  WHERE BOOKID > 100
  WITH LOCAL CHECK OPTION
- INSERT INTO NONFICTIONBOOKS2 VALUES(10, ..., 'F') wird verweigert
- INSERT INTO NONFICTIONBOOKS2 VALUES(120, ..., 'F') funktioniert

Unique/ Nonunique Indexe

- geordnete Liste von Schlüsselwerten einer oder mehrerer Spalten
- sichern die Eindeutigkeit von Spaltenwerten, erhöhen die Performance
- Nonunique Indexe erlauben Duplikate, Unique Indexe erlauben max. einen NULL-Wert
- automatisch erstellt für PRIMARY KEY und UNIQUE
- möglich Sortierung: ascending (Standard), descending, bidirectional
- DB2-Optimierer nutzt Indexe, um Anfragen schneller beantworten zu können

Unique/ Nonunique Indexe

- Create Index IBookname On Books(Bookname)
- Create Index I2Bookname On Books(Authorid Desc , Bookname Asc)
- wenn 2 Anwendungen verschiedene Sortierung brauchen: Create Index BiBookname On Books(Bookname) Allow Reverse Scans
- Erstellung kann dauern, da für jedes Tupel Schlüsselwert bestimmt und dann sortiert wird
- Speicherung in Tablespaces, Angabe bei Create Table mit Option Indexes In , danach nicht mehr änderbar
- Drop Index , keine Modifikation definierter Indexe

Clustering Indexe

- maximal einer pro Tabelle
- nützlich, wenn Daten oft in bestimmter Reihenfolge angefragt werden
- definiert die Reihenfolge, in der die Daten in der DB gespeichert sind
- bei INSERT wird versucht die Zeile möglichst nah bei Zeilen mit ähnlichem Schlüssel zu speichern
- schneller, wenn Anfragen auf gespeicherte Reihenfolge
- CREATE INDEX IAUTHBKNAME ON BOOKS(AUTHORID, BOOKNAME) CLUSTER

Included Columns

- zusätzliche Spalten-Daten im Index speichern (unabhängig vom Index)
- Performancegewinn, wenn alle Daten aus dem Index geholt werden können (Index-Only Auswertung)
- kann nur auf Unique Indexe angewendet werden
- Create Unique Index Ikookid On Books(Bookid) Include(Bookid)
- Select Bookid, Bookname From Book Order By Bookid

- Nutzer kann nicht direkt auf Indexe zugreifen
- permanent gespeichert, brauchen also Speicherplatz
- Aktualisierung indexierter Werte ist teuer, deswegen meist Unsinn alle Spalten einer Tabelle in den Index zu packen
- *Index Advisor* hilft bei der Bestimmung von Indexen