Praktikum Datenbanken / DB2
Woche 4: SQL als DDL, Tabellen und Constraints

Raum: LF 230 Bearbeitung: 9.-11. Mai 2005

Datum	
Gruppe	
Vorbereitung	
Präsenz	

Aktuelle Informationen unter:
`http://www.is.informatik.uni-duisburg.de/courses/dbp_ss03/`

Tabellen in IBM DB2

Tabellen

Eine relationale Datenbank speichert Daten als eine Menge von zweidimensionalen Tabellen. Jede Spalte der Tabelle repräsentiert ein Attribut des Relationenschemas, jede Zeile entspricht einer spezifischen Instanz dieses Schemas. Daten in diesen Tabellen werden mit Hilfe der Structured Query Language (SQL) manipuliert, einer standardisierten Sprache zur Definition und Manipulation von Daten in relationalen Datenbanken:

```
SELECT <Daten>
FROM <Tabelle>
WHERE <Bedingung>
```

Schemata

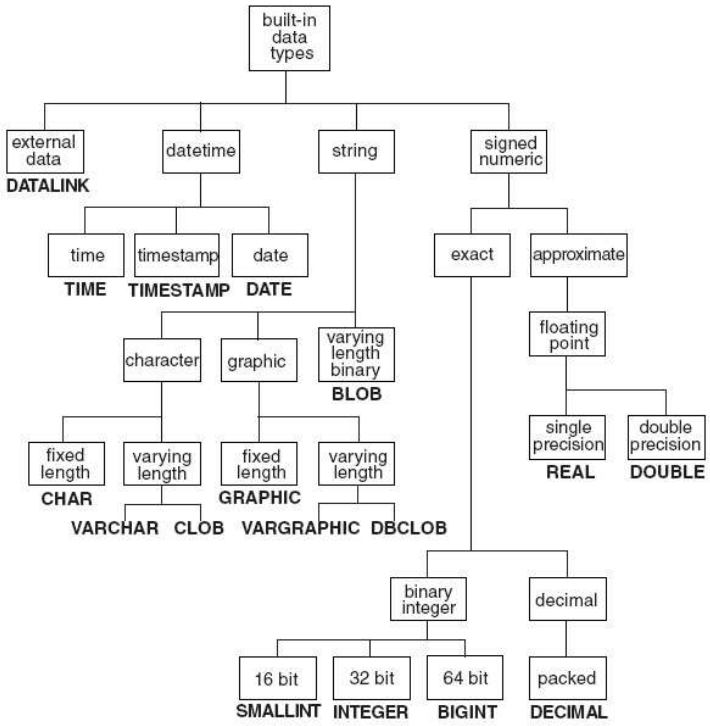
Ein Schema ist eine Sammlung von benannten Objekten (Tabellen, Sichten, Trigger, Funktionen,...). Jedes Objekt in der Datenbank liegt in einem Schema. Objektnamen müssen innerhalb eines Schemas eindeutig sein, ein Schema entspricht also in etwa einem Namensraum. Ein neues Schema kann mit Hilfe des Befehls (db2) `create schema {name}` angelegt, und mit (db2) `drop schema {name} restrict` gelöscht werden.

Ein Benutzer, der über die Datenbankberechtigung IMPLICIT_SCHEMA verfügt, kann ein Schema implizit erzeugen, wenn er in einem CREATE-Statement ein noch nicht existierendes Schema verwendet. Dieses neue, von DB2 erzeugte Schema, gehört standardmäßig SYSTEM und jeder Benutzer hat das Recht, in diesem Schema Objekte anzulegen.

Datentypen

db2s1e80.pdf, S. 92ff

DB2 kennt die in der nachstehenden Grafik abgebildeten Datentypen. Weitere Datentypen können vom Benutzer definiert werden (mehr dazu zu einem späteren Zeitpunkt).



`SMALLINT`	kleine Ganzzahl
`INTEGER`	Ganzzahl
`BIGINT`	große Ganzzahl
`REAL`	Fließkommazahl mit einfacher Genauigkeit
`DOUBLE`	Fließkommazahl mit doppelter Genauigkeit
`DECIMAL(`p`,`s`)`	Dezimalbruch der Länge p mit s Nachkommastellen
`CHAR(`n`)`	String der Länge n (max. 254)
`VARCHAR(`n`)`	String variabler Länge mit maximal n Zeichen (max. 32672)
`BLOB(`sF`)`	Binary Large Object (z.B. BLOB(2 M) für einen 2MB großen BLOB
`DATE`	Datum
`TIME`	Time
`TIMESTAMP`	Zeitstempel
`DATALINK`	Verweis auf eine Datei außerhalb der Datenbank

Zu jedem Datentyp gehört ein NULL-Wert. Dieser ist von allen anderen Werten des Datentyps zu unterscheiden, da er nicht einen Wert an sich darstellt, sondern das Fehlen eines Wertes anzeigt. Der Wert 0 im Gehaltsfeld eines Angestellten könnte z.B. bedeuten, dass der Angestellte ehrenamtlich tätig ist, während der NULL-Wert bedeuten könnte, dass das Gehalt nicht bekannt ist. IBM DB2 bietet Prädikate an, die es z.B. in Anfragen erlauben, zu prüfen, ob ein NULL-Wert vorliegt oder eine andere Ausprägung des Datentyps.

Defaultwerte sind Werte, die vom DBMS eingetragen werden, wenn für das betreffende Attribut kein Wert angegeben wird. Sie bieten die Möglichkeit, Attribute automatisch mit Daten zu versehen, z.B. einer bestimmten Konstante, dem Namen des Benutzers (USER), der aktuellen Uhrzeit (CURRENT TIME), dem aktuellen Datum (CURRENT DATE) oder automatisch generierten Werten (GENERATE). Wird kein spezieller Defaultwert angegeben, so wird der NULL-Wert als Defaultwert verwendet.

SQL als DDL

SQL vereint in sich eine *Data Definition Language*, eine *Data Manipulation Language* und eine *Query Language*. In dieser Sitzung sollen zunächst nur die DDL-Funktionen von SQL zum Einsatz kommen.

Erstellen von Tabellen

Jede Gruppe kann aufgrund der vorgegebenen Rechte in von ihnen erstellten Datenbanken neue Tabellen anlegen, eventuell mit impliziter Erstellung eines neuen Schemas (siehe oben). Zum Anlegen einer Tabelle benutzt man das CREATE TABLE-Statement. Der Benutzer, der die Tabelle erstellt, erhält automatisch das CONTROL-Recht auf dieser Tabelle.

db2s2e80.pdf, S. 332

Beim Erstellen einer Tabelle wird auch ein Primärschlüssel angegeben. Ein Primärschlüssel besteht aus den angegebenen Attributen, die keine NULL-Werte zulassen dürfen und für die Einschränkungen bezüglich der möglichen Datentypen gelten. DB2 legt automatisch einen Index auf dem Primärschlüssel an. Jede

Tabelle kann nur einen Primärschlüssel haben, dieser kann jedoch aus mehreren Attributen bestehen.

```
CREATE TABLE employee (
   id           SMALLINT NOT NULL,
   name         VARCHAR(50),
   department   SMALLINT CHECK (department BETWEEN 1 AND 5),
   job          CHAR(10) CHECK (job in ('Prof','WiMi','NiWiMi')),
   hiredate     DATE,
   salary       DECIMAL(7,2),

   PRIMARY KEY (ID),
   CONSTRAINT yearsal CHECK (YEAR(hiredate) > 1996
                             OR SALARY > 1500)
)
```

Als Shortcut kann man Tabellen auch mit folgendem Statement erstellen:

(db2) `CREATE TABLE name LIKE other_table`

Dies übernimmt die Attribute der Tabelle mit den exakt gleichen Namen und Definitionen aus einer anderen Tabelle oder einem View.

Gelöscht werden Tabellen mit dem DROP-Statement. Dazu benötigt man das CONTROL-Recht für die Tabelle, das DROPIN-Recht im Schema der Tabelle, das DBADM- oder das SYSADM-Recht. Wird eine Tabelle gelöscht, so werden in allen Tabellen, die diese Tabelle referenzieren, die zugehörigen Fremdschlüsselbeziehungen gelöscht. Alle Indexe, die auf der Tabelle bestehen, werden gelöscht.

db2s2e80.pdf, S. 512

Schlüssel und Constraints

Primärschlüssel:
Mit PRIMARY KEY wird für die Tabelle ein Primärschlüssel festgelegt. Dieser besteht aus den angegebenen Attributen. Die Attribute im Primärschlüssel dürfen keine Nullwerte zulassen. Soll der Schlüssel nur aus einem Attribut bestehen, dann genügt es, das Schlüsselwort hinter dem jeweiligen Attribut anzugeben:

```
id SMALLINT NOT NULL PRIMARY KEY,
```

Sekundärschlüssel:
Durch das Schlüsselwort UNIQUE wird ein Sekundärschlüssel festgelegt. Auch für diesen wird automatisch ein Index erstellt und es gelten die gleichen Einschränkungen wie für Primärschlüssel. Attribute eines Sekundärschlüssels dürfen nicht schon Teil eines anderen Schlüssels sein. Beispiel:

```
jobnr INTEGER NOT NULL UNIQUE,
```

Fremdschlüssel:
Fremdschlüssel werden benutzt, um zu erzwingen, dass ein oder mehrere Attribute einer abhängigen Tabelle nur Werte annehmen dürfen, die auch als Werte eines Schlüssels einer anderen Tabelle auftreten. Dieses Konzept wird auch *Referentielle Integrität* genannt. Der Benutzer muss das REFERENCES- oder ein

übergeordnetes Recht auf der referenzierten Tabelle besitzen. Die Tabelle, auf die verwiesen wird, muss existieren und die referenzierten Attribute müssen dort als Schlüssel definiert sein.

Fremdschlüssel werden entweder als *Constraint* (s.u.) oder mit dem Schlüsselwort REFERENCES angegeben. In diesem Beispiel wird durch das Attribut `jobnr` die Spalte `internal_number` der Tabelle `jobs` referenziert:

```
jobnr INTEGER REFERENCES jobs(internal_number)
```

Checks:
Bei der Definition eines Tabellenattributes kann zusätzlich zum Datentyp eine Einschränkung der Werte angegeben werden. Dies wird mit dem Schlüsselwort CHECK eingeleitet:

```
department SMALLINT CHECK (department BETWEEN 1 AND 5),

job CHAR(10) CHECK (job in ('Professor','WiMi','NichtWiMi')),
```

Constraints:
Checks und *Fremdschlüssel* können auch als benannte *Constraints* mit dem Schlüsselwort CONSTRAINT angegeben werden:

```
CONSTRAINT jobref FOREIGN KEY jobnr REFERENCES jobs (int_no)

CONSTRAINT yearsal CHECK (YEAR(hiredate) > 1996 OR SALARY > 1500)
```

Index

Ein Index ist ein Datenbankobjekt, das den Zugriff über bestimmte Attribute einer Tabelle beschleunigen soll und dies in den allermeisten Fällen auch tut. Man kann auch (UNIQUE)-Indexe verwenden, um die Einhaltung von Schlüsselbedingungen sicherzustellen.

Werden bei der Tabellendefinition Primärschlüssel- (PRIMARY KEY) oder Sekundärschlüsselbedingungen (UNIQUE) angegeben, so legt DB2 selbständig Indexe an, um die Einhaltung dieser sicherzustellen. Ein einmal angelegter Index wird dann automatisch bei Änderungen an den Inhalten der entsprechenden Tabelle gepflegt. Dies ist natürlich mit einem erhöhten Verwaltungsaufwand verbunden.

Näheres zu Indexen und den zur Realisierung verwendeten Datenstrukturen in der Vorlesung **Datenbanken** oder in der begleitenden Literatur (siehe Blatt der ersten Woche).

Das Anlegen eines Index geht mit dem CREATE INDEX-Statement, das Löschen entsprechend über DROP INDEX. Man benötigt dazu mindestens das INDEX- oder das CONTROL-Recht für die Tabelle, sowie das CREATEIN-Recht im angegebenen Schema. Der Benutzer, der einen Index anlegt, erhält auf diesem das CONTROL-Recht.

db2s2e81.pdf, S. 268

Über das Schlüsselwort UNIQUE kann man bei der Erstellung eines Index angeben, dass keine Tupel in allen Werten der angegebenen Attribute übereinstimmen dürfen. Sollten zum Zeitpunkt der Indexgenerierung Tupel in der Tabelle

dagegen verstoßen, so wird eine Fehlermeldung ausgegeben und der Index nicht angelegt.

Es ist nicht möglich, zwei gleiche Indexe anzulegen. Dabei darf ein Index 16 Attribute umfassen und die Attribute dürfen zusammen nicht länger als 1024 Bytes sein. Die Ordnung ASC bzw. DESC gibt an, ob die Indexeinträge in aufsteigender bzw. absteigender Reihenfolge angelegt werden. Nur wenn UNIQUE angegeben wurde, können über die INCLUDE-Klausel weitere Attribute in den Index aufgenommen werden, für die aber keine Schlüsselbedingung gelten soll.

Beispiel:

```
CREATE INDEX studios
   ON produktion (studio)
```

Indexe werden nicht explizit aufgerufen, sondern vom Datenbank-Managementsystem implizit bei der Bearbeitung von Anfragen herangezogen.

Übersicht über die neuen Befehle

create schema *name*	erstelle ein neues Schema (einen Namensraum für Datenbankobjekte)
drop schema *name* restrict	lösche ein existierendes Schema
create table *name*	erstelle eine neue Tabelle
drop table *name*	lösche eine existierende Tabelle
create index *name* on *table*	erzeuge einen neuen Index auf einer Tabelle
describe table *name* show detail	zeige Details zu einer existierende Tabelle an
describe indexes for table *name*	zeige die Indexe auf der Tabelle *name* an

Aufgaben

Aufgaben zur Vorbereitung

(a) Überlege Dir zu den in dritte Normalform gebrachten Relationen aus der letzten Woche die passenden Datentypen für die Attribute. Für einige Attributdomänen gibt es keine direkte Entsprechung unter den Datentypen von DB2. Wähle passende aus.

(b) Drücke folgende Einschränkungen in SQL aus:

(*i*) Die Fläche eines Sees soll keine negativen Werte annehmen können.

(*ii*) Die Länge einer Grenze soll eine positive Zahl sein.

(*iii*) Das Datum der Unabhängigkeit eines Landes soll nicht nach dem aktuellen Datum liegen.

(*iv*) Die Summe der Prozentanteile des primären, sekundären und tertiären Sektors (drei Attribute einer Tabelle) sollen insgesamt nicht mehr als 100 ergeben. Weniger als 100% soll durch Rundungsfehler oder wegen eventuell fehlender Angaben erlaubt sein.

(*v*) Die Position einer Stadt nach geographischer Länge und Breite kann nur zwischen -180 und 180 (Länge). bzw. -90 und 90 (Breite) liegen.

(c) Welche Fremdschlüselbeziehungen zwischen Relationen ergeben sich aus Deiner bisherigen Modellierung der Miniwelt.

Präsenzaufgaben

(a) Erstellt in Eurer Datenbank `almanach` die für Eure Relationen benötigten Tabellen. Gebt dazu das komplette CREATE TABLE-Statement an. Vergesst nicht, alle nötigen Primärschlüssel und Fremdschlüssel anzugeben. Übernehmt die Constraints aus der Vorbereitung wo sinnvoll in Eure Tabellendefinitionen.

(b) Erstellt wenn nötig zusätzliche Indexe auf Euren Tabellen. Erstellt mindestens einen Index und gebt den Befehl dazu an: