Schlüsselformen und ihre Anwendung in SQL

Primärschlüssel und weitere Konsistenzbedingungen

Syntax:

CREATE TABLE (key\_id DATENTYP [Bedingungen],…,

PRIMARY KEY (key\_id))[Autoinc Startwert];

Bedingungen (unter anderem):

* NOT NULL: Die so gekennzeichnete Spalte darf niemals leer bleiben
* UNIQUE: Eine solche Spalte muss einzigartige Werte aufweisen (nicht notwendig bei einem Primärschlüssel, da dieser sowieso UNIQUE sein muss!)
* AUTO\_INCREMENT: Diese Spalte wird automatisch von Datensatz zu Datensatz hochgezählt. Die Spalte muss dafür natürlich einen geeigneten Datentyp aufweisen.

Nach den Angaben zur Tabelle (siehe Syntax oben) kann man vor dem abschließenden Semikolon (;) noch einen Startwert für das AUTO\_INCREMENT angeben – ansonsten würde bei 0 oder 1 gestartet.  
**Beachte:** Auch fehlgeschlagene Inserts (und entsprechendes), die die AUTO\_INCREMENT-Spalte betreffen, setzen den Zähler um 1 weiter!

Der aktuelle Zählwert kann mit der LAST\_INSERT\_ID() – Funktion ermittelt werden. Dies funktioniert aber nur innerhalb der selben Verbindung zur Datenbank (also: im selben Fenster in DBeaver (bei den normalen Einstellungen!)

Nach den Angaben zu den Spalten kann man dann explizit den **Primärschlüssel** setzen mit PRIMARY KEY (Spaltenname).  
Dies bringt zunächst einmal keine weiteren Vorteile, außer dass NOT NULL und UNIQUE erzwungen werden. Allerdings müssen eventuell später gesetzte **Fremdschlüssel** sich auf Primärschlüssel beziehen!

Außerdem wird die PRIMARY KEY-Spalte (oder Spalten) intern indiziert; es gibt also eventuelle Geschwindigkeits-Vorteile!

Zusammengesetzte Schlüssel

Syntax:

CREATE TABLE (key1 DATENTYP [Bedingungen], key2 DATENTYP…,

PRIMARY KEY (key1, key2,…));

Der Primärschlüssel kann sich auch aus mehreren Spalten zusammensetzen; dabei muss die Gesamtheit der angegebenen Spalten eine einzigartige Kombination sein. Die Syntax funktioniert analog dazu; man gibt einfach mehrere Spalten bei PRIMARY KEY() an.

Fremdschlüssel

Syntax:

CREATE TABLE (Spalte1…, Spalte2…, [PRIMARY KEY(…)], CONSTRAINT Constr\_Name FOREIGN KEY (Spalte) REFERENCES primary\_key\_tabelle (primary\_key\_spalte) ON UPDATE CASCADE ON CASCADE);

Ein Fremdschlüssel wird mit dem CONSTRAINT-Keyword erzeugt; diesem „Constraint“ muss man einen Namen geben, um ihn später auch wieder löschen zu können.  
Anschließend folgt FOREIGN KEY, dann der Fremdschlüssel-Spaltenname.

Diese Spalte muss dann auf die entsprechende Tabelle/Spalte des Primärschlüssels zugreifen; dies geschieht durch REFERENCES primary\_key\_tabelle (primary\_key\_spalte).

Außerdem muss noch angegeben werden, was passiert, wenn der referenzierte Primärschlüssel geändert oder gelöscht wird mit ON UPDATE… und ON DELETE… :

* CASCADE: Hier wird der geänderte Schlüssel auch an die Fremdschlüssel („kaskadenartig“) weitergegeben; es werden also auch die betroffenen Fremdschlüssel-Felder geändert.  
  Beim Löschen des primären Datensatzes wird analog kaskadenartig auch der Fremdschlüssel-Datensatz gelöscht!
* RESTRICT: (Default): Veränderungen am Primärschlüssel werden verhindert. NO ACTION bewirkt das Gleiche.
* SET NULL: Primärschlüssel kann geändert werden; Fremdschlüssel-Datensätze werden NULL gesetzt.

Beachte auch: Ohne bereits vorhandenen Primärschlüssel kann kein Fremdschlüssel gesetzt werden! Dementsprechend muss man die Tabellen in der richtigen Reihenfolge anlegen.

Mit dem CONSTRAINT-Schlüsselwort können auch weitere Bedingungen an die Konsistenz der Daten gestellt werden (z.B. Spalte > 0 etc.)