SQL Integritätsregeln

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 3 (zu größeren Teilen überprüft)	**Umfang**: 3 (einige wichtige Fakten fehlen)	**Quellenangaben**: 1 (fehlen großteils)	**Quellenarten**: 5 (ausgezeichnet)	**Konformität**: 5 (ausgezeichnet)

Inhaltsverzeichnis

1 Definition

Die Inhalte einer Datenbank sollten idealerweise fehlerfrei und in sich schlüssig sein. Beim Anlegen von Tabellen werden auch gleich deren Integritätsregeln festgelegt. Somit werden fehlerhafte Datensätze gar nicht erst angenommen.

Es gibt folgende Integritätsregeln:

Primärschlüssel
Jeder Datensatz wird eindeutig ansprechbar.
Fremdschlüssel
Tabellen werden eindeutig verknüpft.
UNIQUE
Es wird kein Wert doppelt abgespeichert.
CHECK
Fehleingaben werden verhindert durch Überprüfungsregeln.
DEFAULT
Standardwerte können gesetzt werden.

1.1 Primärschlüssel (PRIMARY KEY)

Mit einem Primärschlüssel werden alle Datensätze eindeutig identifizierbar. Beim Erstellen von Datensätzen muss das Feld dieser Spalte gefüllt werden. Ist das nicht der Fall oder ist dieser Wert schon vorhanden wird der Datensatz nicht angenommen. Die Spalte die als Primärschlüssel bestimmt wird, sollte dafür auch geeignet sein, wie beispielsweise die Matrikelnummer jedes Studenten.

```
CREATE TABLE student
      (
      s_matr     INTEGER    NOT NULL,
      s_name     VARCHAR    NOT NULL,
      s_vorname  VARCHAR,

      CONSTRAINT student_pk
      PRIMARY KEY (s_matr)
      );
```

Hier wird der Primärschlüssel durch PRIMARY KEY gesetzt und in den Klammern der betreffende Spaltenname. Mit CONSTRAINT bekommt der Primärschlüssel einen Namen, durch den er später ansprechbar wird. Man könnte PRIMARY KEY auch direkt hinter den Spaltennamen setzen, wenn die Tabelle nur einen Primärschlüssel bekommt.

1.2 Fremdschlüssel (FOREIGN KEY)

Fremdschlüssel dienen zur Verknüpfung von Tabellen und beziehen sich auf den Primärschlüssel einer anderen Tabelle. Dadurch werden die zwei Tabellen von einander abhängig. Wird eine Spalte als Fremdschlüssel gesetzt, können nur noch Werte eingesetzt werden, die in der verknüpften Tabelle, Spalte Primärschlüssel, eingebbar sind.

```
CREATE TABLE studiengang
      (
      l_id         INTEGER    NOT NULL,
      l_name       VARCHAR    NOT NULL,

      CONSTRAINT studiengang_pk
      PRIMARY KEY (l_id)
      );

CREATE TABLE student
      (
      s_matr            INTEGER    NOT NULL,
      s_name            VARCHAR    NOT NULL,
      s_vorname         VARCHAR,
      s_studiengang     VARCHAR,

      CONSTRAINT student_pk
      PRIMARY KEY (s_matr),

      FOREIGN KEY (s_studiengang) REFERENCES studiengang (l_id)
      );
```

Man legt den Fremdschlüssel mit FOREIGN KEY direkt an die Spaltendefinition an oder danach. Auch hier könnte man diesen mit CONSTRAINT benennen. Mit REFERENCES wird die Tabelle, die „Vatertabelle“, angegeben, auf die sich der Fremdschlüssel bezieht und in den Klammern den Primärschlüssel. Die Tabelle, auf die verwiesen wird, muss jedoch schon vorhanden sein.

Wird ein Datensatz aus der Vatertabelle gelöscht, müssen auch die Datensätze aus der abhängigen Tabelle behandelt werden. Dies dient zur Bewahrung der Datenintegrität.

Dafür gibt es vier Fälle:

NO ACTION
Wird aus der Vatertabelle etwas gelöscht, passiert in der abhängigen Tabelle nichts.
CASCADE
Die Datensätze der abhängigen Tabelle werden mitgelöscht oder aktualisiert.
SET DEFAULT
Der Schlüsselwert der abhängigen Tabelle wird auf DEFAULT gesetzt.
SET NULL
Wenn etwas in der Vatertabelle geändert oder gelöscht wird, wird der abhängige Wert auf NULL gesetzt.

Das könnte dann folgendermaßen aussehen:

```
CREATE TABLE student
      (
      s_matr            INTEGER    NOT NULL,
      s_name            VARCHAR    NOT NULL,
      s_vorname         VARCHAR,
      s_studiengang     VARCHAR,

      CONSTRAINT student_pk
      PRIMARY KEY (s_matr),

      FOREIGN KEY (s_studiengang) REFERENCES studiengang (l_id)

      [ON DELETE {NO ACTION | CASCADE  |SET DEAFAULT | SET NULL}]
      [ON UPDATE {NO ACTION | CASCADE  |SET DEAFAULT | SET NULL}]
      );
```

1.3 UNIQUE

Eine weitere Möglichkeit um zu verhindern, dass ein Wert in einer Spalte öfter vorkommt, außer dem Primärschlüssel, lautet UNIQUE. Das Datenbanksystem verhindert damit automatisch, dass ein Wert doppelt eingetragen wird.

Es gibt zwei Möglichkeiten UNIQUE zu setzten. Einmal direkt hinter der Spaltendefinition oder danach. Dabei können auch gleichzeitig mehrere Spalten angegeben werden.

```
CREATE TABLE student
      (
      s_matr            INTEGER    NOT NULL,
      s_name            VARCHAR    NOT NULL,
      s_vorname         VARCHAR,
      s_studiengang     VARCHAR,

      CONSTRAINT student_pk
      PRIMARY KEY (s_matr),

      UNIQUE (s_name, s_studiengang [...])
      );
```

1.4 CHECK

Mit CHECK lassen sich Regeln selbst definieren. Bestimmte Spalten werden dann beim Eingeben von Datensätzen überprüft und diese daraufhin zugelassen oder nicht. Das heißt man kann testen, ob eine Eingabe zu bestimmten Werten gehört, oder in einem bestimmten Verhältnis zu einem anderen Wert steht.

```
CREATE TABLE buch
      (
      b_nr              INTEGER    NOT NULL,
      b_titel           VARCHAR    NOT NULL,
      b_verlag          VARCHAR,
      b_anzahl          INTEGER,
      b_bestellmenge    INTEGER,

      CONSTRAINT buch_pk
      PRIMARY KEY (b_nr),

      CHECK (b_bestellmenge <= b_anzahl)
      );
```

Auch hier kann CHECK direkt an die Spaltendefinition oder an das Ende der Tabellendefinition. Auch kann hier wieder mit CONSTRAINT gearbeitet werden.

1.5 DEFAULT

Man kann Standardwerte vorgeben, die dann, wenn nicht anderes angegeben wird, eingesetzt werden. Dadurch werden keine NULL-Werte gespeichert.

```
CREATE TABLE buch
      (
      b_nr              INTEGER    NOT NULL,
      b_titel           VARCHAR    NOT NULL,
      b_verlag          VARCHAR,
      b_anzahl          INTEGER DEFAULT '0',
      b_bestellmenge    INTEGER,

      CONSTRAINT buch_pk
      PRIMARY KEY (b_nr)
      );
```

Hier wird DEFAULT einfach an die Spaltendefinition hinten dran gehängt und der Wert angegeben. Somit wird automatisch wenn nichts anderes angeben wird '0' eingesetzt. Man könnte auch die CHECK Klausel mit DEFAULT erweitern.

2 Quellen

Throll, M.; Bartosch, O. (2007): Einstieg in SQL

Kategorien:
Glossar
Datenmanagement