Leibniz Universität IT Services

Leibniz Universität Hannover

S(tructured)Q(uery)L(anguage) Bedingungen für Datenfelder

Bedingung (Constraint)

- Definition von Regeln, die während der Dateneingabe in ein Feld überprüft werden.
- Bedingungen, die ein Datenfeld zwingend erfüllen muss. Andernfalls können die Daten nicht gespeichert werden.
- Wahrung der Identität von Daten. Korrekte und vollständige Speicherung der Daten in den entsprechenden Feldern.

Integritätsregeln

- `Primary Key`. Jeder Datensatz ist eindeutig identifizierbar.
- `Foreign Key`. Tabellen werden eindeutig verknüpft. Zu jedem Fremdschlüssel in einer Detailtabelle ist ein Primärschlüssel in der Master-Tabelle vorhanden.
- `Not Null`. Primärschlüssel dürfen nicht leer sein. Das Attribut-Wert ist zwingend erforderlich.
- `Unique`. Es wird kein Wert doppelt gespeichert. Der Attribut-Wert existiert nur einmal in einer Tabelle.
- `Default`. Standardwerte können gesetzt werden.
- `Check`. Mit Hilfe von Überprüfungsregeln werden Fehler bei der Eingabe vermieden.

Spalten-Bedingung (Column-Constraint)

```
CREATE TABLE IF NOT EXISTS tblWare (

produktcode VARCHAR(10) NOT NULL PRIMARY KEY,
produktname VARCHAR(200) NOT NULL UNIQUE,
produktbeschreibung TEXT,
listenpreis NUMERIC(8,3)
    NOT NULL DEFAULT 1.99
    CHECK (listenpreis > 0.99
        AND listenpreis < 100.00),
warengruppe_id INTEGER
    REFERENCES tblWarengruppe(id),
aufgenommenAm TIMESTAMP DEFAULT now()
);
```

Aufbau

produktcode	VARCHAR(10)	NOT NULL PRIMARY KEY
feldname	feldtyp	Constraint

- Die Bedingung bezieht nur auf ein Datenfeld.
- Festlegung bei der Definition eines Datenfeldes.
- Für ein Datenfeld können mehrere Bedingungen definiert werden. Die Befehle werden durch ein Leerzeichen getrennt.

Tabellen-Bedingung (Table-Constraint)

```
CREATE TABLE IF NOT EXISTS tblWare (
produktcode VARCHAR(10) NOT NULL ,
produktname VARCHAR(200) NOT NULL,
produktbeschreibung TEXT,
listenpreis NUMERIC(8,3) NOT NULL DEFAULT 1.99 ,
warengruppe_id INTEGER
aufgenommenAm TIMESTAMP DEFAULT now(),

CONSTRAINT idTblWare PRIMARY KEY(produktcode),
CONSTRAINT verweis FOREIGN KEY(warengruppe_ID)
           REFERENCES tblWarengruppe(id),
CHECK (listenpreis > 0.99
           AND listenpreis < 100.00),
UNIQUE(produktname)
);
```

Aufbau

PRIMARY KEY	(	produktcode	)
UNIQUE	(	produktname	)
Constraint	(	Feld1, feld2, ...	)

- Die Bedingung bezieht sich auf mehr als ein Datenfeld.
- Die Bedingungen werden am Ende der Tabellendefinition aufgelistet.
- Dem Namen des Constraint folgt eine Liste von Feldern, die die Bedingung erfüllen müssen. Die Liste wird durch runden Klammern begrenzt. Die Elemente werden durch ein Komma getrennt.

Zusammenfassung von Bedingungen

`CONSTRAINT`	`idTblWare`	`PRIMARY KEY(produktcode)`
		`UNIQUE(produktname)`
`CONSTRAINT`	`name`	`Constraint`

- Dem Schlüsselwort `CONSTRAINT` folgt ein Namen. Der Name ist einmalig in der Datenbank
- Darunter werden alle Bedingungen aufgeführt, die unter diesem Namen zusammengefasst werden sollen.

Primärschlüssel (Primary key)

- Eindeutige Identifizierung eines Datensatzes in einer Tabelle.
- Sobald ein neuer Datensatz angelegt wird, muss der Schlüsselwert gesetzt werden.
- Während der Existenz des Datensatzes wird der Schlüssel niemals geändert.

… definieren

```
id SERIAL NOT NULL PRIMARY KEY,
```

```
PRIMARY KEY(id),
```

- Der Primärschlüssel kann mit Hilfe einer Spalten-Bedingung oder Tabellen-Bedingung definiert werden.
- Der Befehl `PRIMARY KEY` kennzeichnet Felder als Schlüsselfelder.
- Schlüsselfelder dürfen nicht leer sein. Sie sind einmalig.

Fremdschlüssel (Foreign key)

- Platzhalter für Datensätze, die in einer anderen Tabelle abgelegt sind.
- Verweis auf einen Datensatz in einer übergeordneten Tabelle.
- Ein Fremdschlüssel kann beliebig oft in einer untergeordneten Tabelle vorkommen, kommt aber nur einmal in der übergeordneten Tabelle vor.
- Primär- und Fremdschlüssel müssen vom gleichen Feldtyp sein.

… definieren

```
warengruppe_id INTEGER
      REFERENCES tblWarengruppe(id),
```

```
FOREIGN KEY(warengruppe_ID)
      REFERENCES tblWarengruppe(id),
```

- Der Fremdschlüssel kann mit Hilfe einer Spalten-Bedingung oder Tabellen-Bedingung definiert werden.
- Der Befehl `FOREIGN KEY` kennzeichnet Felder als Fremdschlüssel.
- Mit Hilfe des Befehls `REFERENCES` wird das Feld definiert, auf welches verwiesen wird.

Verweis auf ein Feld

`REFERENCES`	`tabelle`	`(feld)`

- Dem Schlüsselwort `REFERENCES` folgt der Tabellenname, in dem das passende Gegenstück zu dem Fremdschlüssel definiert ist.
- Dem Tabellennamen folgt in runden Klammern der Feldname, auf das der Fremdschlüssel verweist.

Eindeutiger Attribut-Wert

```
produktname VARCHAR(200) NOT NULL UNIQUE,
```

```
UNIQUE(produktname)
```

- Felder, deren Wert nur einmal in der Tabelle vorkommen dürfen, können mit Hilfe einer Spalten-Bedingung oder Tabellen-Bedingung gekennzeichnet werden.
- Die Werte in diesen Feldern sind einzigartig in einer Tabelle.
- Der Befehl UNIQUE steht immer nach der Angabe `DEFAULT [wert]` in einer Spaltenbedingung.

Vermeidung eines undefinierten Wertes

```
produktname VARCHAR(200) NOT NULL UNIQUE,
```

- Ein Datenfeld hat den Wert Null. Der Attribut-Wert ist undefiniert.
- Der Befehl `NOT NULL` erzwingt eine Eingabe in das Feld. Die Information muss definiert werden.
- Die Bedingung `NOT NULL` kann nur in einer Felddefinition gesetzt werden.

Setzen eines Standardwertes

```
listenpreis NUMERIC(8,3) NOT NULL DEFAULT 1.99,
aufgenommenAm TIMESTAMP DEFAULT now()
```

- Falls bei der Neu-Eingabe eines Datensatzes das Feld leer ist, wird der Standardwert genutzt.
- Das Datenfeld wird automatisch mit dem Standardwert gefüllt, wenn das Feld null (leer) ist.
- Dem Befehl `DEFAULT` folgt ein Wert. Der Wert kann durch eine Funktion berechnet werden.

Nutzung von Literalen

```
feld01 INTEGER DEFAULT 1,
feld02 NUMERIC(8,3) DEFAULT 2.3,
feld03 VARCHAR(3) DEFAULT 'xxx',
feld04 DATE DEFAULT '2017-07-11',
feld05 TIME DEFAULT '05:15 PM',
feld06 TIMESTAMP DEFAULT '2017-07-11 05:15 PM',
feld07 BOOLEAN DEFAULT TRUE,
feld08 BOOLEAN DEFAULT FALSE
```

Nutzung von Funktionen

```
aufgenommenAm TIMESTAMP DEFAULT now()
```

- Der Rückgabewert der Funktion wird als Standardwert genutzt.
- Die implementieren Funktionen sind abhängig vom verwendeten Datenbanksystem.
- Funktionen, die in PostgreSQL implementiert sind, werden auf der Seite https://www.postgresql.org/docs/current/static/functions.html aufgelistet.

Erläuterung der Funktion

```
aufgenommenAm TIMESTAMP DEFAULT now()
```

- Die Funktion `now()` liefert den aktuellen Zeitstempel zurück.
- Einigen Funktionen werden in den runden Klammern Parameter übergeben. Falls die Klammer leer ist, werden der Funktion keine Parameter übergeben.

Überprüfung der eingegebenen Information

```
CHECK (listenpreis > 0.99 AND listenpreis < 100.00)
```

- Dem Befehl `CHECK` folgt in runden Klammern eine Bedingung.
- Eine Bedingung bildet Ja / Nein-Fragen ab.
- Wenn die eingegebene Information die Bedingung erfüllt, wird diese in der Tabelle gespeichert.

Bedingung

`listenpreis`	`>`	`0.99`
`datenfeld`	`vergleichsoperator`	`wert`

- Die eingegebenen Informationen werden mit einem Wert verglichen.
- Der Wert kann als Literal angegeben werden. Mit Hilfe einer Funktion kann der Wert berechnet werden.
- Bedingungen können verknüpft werden.

Vergleichsoperatoren

ist ...	Operator
gleich	=
ungleich	<>
	!=
kleiner	<
kleiner gleich	<=
größer	>
größer gleich	>=

Oder-Verknüpfung

`bedingung`	`OR`	`bedingung`

- Eine der beiden Bedingungen muss zutreffen.
- Falls die linke Bedingung zutrifft, wird die rechte Bedingung nicht mehr untersucht.

Und-Verknüpfung

`bedingung`	`AND`	`bedingung`

- Beide Bedingungen muss zutreffen.
- Falls die linke Bedingung nicht zutrifft, wird die rechte Bedingung nicht mehr untersucht.

Nicht-Verknüpfung

NOT	(	bedingung	)

- Die Bedingung wird negiert.
- TRUE → FALSE.
- FALSE → TRUE.

Nutzung von Funktionen

```
CHECK (CHAR_LENGTH(produktcode) >= 5),
```

- Der Rückgabewert der Funktion wird mit einem anderen Wert verglichen.
- Die implementieren Funktionen sind abhängig vom verwendeten Datenbanksystem.
- Funktionen, die in PostgreSQL implementiert sind, werden auf der Seite https://www.postgresql.org/docs/current/static/functions.html aufgelistet.

Erläuterung

```
CHECK (CHAR_LENGTH(produktcode) >= 5),
```

- Der Funktion `CHAR_LENGTH` wird das zu untersuchende Datenfeld in den runden Klammern übergeben.
- Die Funktion gibt die Anzahl der Zeichen der neu eingegebenen Daten in dem Datenfeld `produktcode` zurück.
- Wenn die Anzahl kleiner als fünf ist, wird der neue Datensatz nicht gespeichert.

Vergleich mit anderen Datenfeldern

```
CHECK (verkaufspreis >
                (einkaufspreis + einkaufspreis * 0.1))
```

- Das Feld `einkaufspreis` in der selben Tabelle definiert wie das Datenfeld `verkaufspreis`.
- Der Inhalt des Feldes `verkaufspreis` muss größer als der Einkaufspreis plus 10% sein.