



DB: Datenbanken

Integrität

Prof. Dr. Ludger Martin

Gliederung

- ★ Einführung
- ★ Arten von Integrität
- ★ Allgemeine Aussagen
- ★ Trigger

Einführung

- ★ Integritätskontrolle bezeichnet die Verhinderung semantischer Fehler bzw. semantisch unsinniger Zustände der Datenbank
- ★ Durch Einhaltung und Überwachung von geforderten Integritätsbedingungen
- ★ Der Aufwand zur Überwachung der Integrität ist oft sehr hoch

Einführung

★ Mögliche Zuständigkeiten

- ★ Integritätskontrolle **durch Anwendungsentwickler** oder Benutzer (DBMS macht keine Kontrolle)
 - ★ **Transaktionen** sorgen für Integrität, nach Beenden einer Transaktion ist DB konsistent (auf DB ausgeführte Programme)
 - ★ **DBMS** übernimmt Kontrolle, weist Transaktionen zurück, die Inkonsistenz hervorrufen (spezieller Monitor des DBMS)
- Bei Integritätskontrolle durch DBMS muss sich Benutzer oder Anwendungsentwickler nicht um Integrität kümmern

Arten von Integrität

- ★ **Statische Bedingungen**, die mögliche Zustände einschränken. Zuständen werden boolesche Werte zugeordnet (z.B. wahr)
- ★ **Transitionale** oder halb-dynamische **Bedingungen**, schränken mögliche Zustandsübergänge ein, ein Paar von aufeinanderfolgenden Zuständen
- ★ **Dynamische Bedingungen**, schränken mögliche Zustandsfolgen ein, mehrere Zustände, die in zeitlicher Abfolge durchlaufen werden


Arten von Integrität

Beispiele für Bedingungen

1. Domain- oder Attributbedingungen

a) Festlegung von Unter- und Obergrenzen

```
CREATE TABLE medienartikel (  
    a_nr varchar(13) NOT NULL,  
    titel varchar(100) NOT NULL,  
    jahr int(4) NOT NULL  
        CHECK (jahr BETWEEN 1990 AND 2020),  
    PRIMARY KEY (`a_nr`)  
)
```



In MySQL/MariaDB wird die CHECK Klausel ignoriert! In Oracle und DB2 eingeschränkt!

Arten von Integrität

Beispiele für Bedingungen

1. Domain- oder Attributbedingungen

b) Festlegung von Werten

... CHECK bestellweg IN
('Internet', 'Postweg', 'Kurier')

In MySQL/MariaDB wird die
CHECK Klausel ignoriert! In Oracle
und DB2 eingeschränkt!

c) Ausschließung von NULL-Werten, NOT-NULL-Bedingung

Arten von Integrität

Beispiele für Bedingungen

2. **Tupel-Bedingungen** betreffen einzelne Tupel, Werte innerhalb einer Zeile
(z.B. Telefonnummer verschieden von Fax)

Arten von Integrität

Beispiele für Bedingungen

3. Relationen-Bedingungen betreffen alle Tupel einer Relation

a) Schlüssel-Bedingungen

PRIMARY KEY (typ, jahr)

b) Aggregat-Bedingungen

(z.B. die Summe der Gehälter aller Schauspieler eines Filmes darf eine durch das Budget gegebene Obergrenze nicht überschreiten)

c) Rekursive Bedingungen

(z.B. für Zugverbindungen, jeder Ort ist von jedem anderen aus erreichbar)

Arten von Integrität

Beispiele für Bedingungen

4. **Referentielle Bedingungen**, Verbindung zwischen Paaren von Relationen, Inklusions- und Exklusionsabhängigkeiten

$$Buch[ANr] \subseteq Medienartikel[ANr]$$

als SQL:

```
FOREIGN KEY (a_nr) REFERENCES  
    medienartikel (a_nr)
```

Allgemeine Aussagen

★ Beispiel Buch:

```
CREATE TABLE buch (  
  a_nr varchar(13) NOT NULL,  
  sprache varchar(5) NOT NULL,  
  auflage int(11) NOT NULL,  
  verlagname varchar(30) NOT NULL,  
  isbn varchar(29) NOT NULL UNIQUE,  
  kategorie varchar(12) NOT NULL  
    CHECK kategorie IN ('Belletristik',  
                        'Sachbuch'),  
  PRIMARY KEY (a_nr),  
  FOREIGN KEY (a_nr) REFERENCES ...  
  FOREIGN KEY (verlagname) REFERENCES ...  
)
```

In MySQL/MariaDB wird die CHECK Klausel ignoriert! In Oracle und DB2 eingeschränkt!

Angabe der Kategorien soll flexibler werden

Allgemeine Aussagen

★ Einführung einer neuen Tabelle Buchkategorien:

```
CREATE TABLE buchkategorien (  
    kategorie varchar(25) NOT NULL  
);
```

```
INSERT INTO buchkategorien VALUES  
    ('Belletristik'),  
    ('Sachbuch');
```

Allgemeine Aussagen

★ Ändern der Tabelle Buch:

```
CREATE TABLE buch (  
  a_nr varchar(13) NOT NULL,  
  sprache varchar(5) NOT NULL,  
  auflage int(11) NOT NULL,  
  verlagname varchar(30) NOT NULL,  
  isbn varchar(29) NOT NULL UNIQUE,  
  kategorie varchar(12) NOT NULL  
    CHECK kategorie IN (SELECT DISTINCT  
                        kategorie FROM buchkategorien),  
  PRIMARY KEY (a_nr),  
  FOREIGN KEY (a_nr) REFERENCES medienartikel ...  
  FOREIGN KEY (verlagname) REFERENCES verlag ...  
)
```

In MySQL/MariaDB wird die
CHECK Klausel ignoriert! In Oracle
und DB2 eingeschränkt!

Allgemeine Aussagen

- ☆ Es wird verlangt, dass das Attribut `kategorie` aus Werten der Tabelle `buchkategorien` besteht.
 - Flexible Erweiterung der zulässigen Buchkategorien
 - Definition von Bedingungen auf Tupelebene
- ☆ Aussagen können auch außerhalb der Tabellendefinition angegeben werden
 - ★ Definition der Aussagen für Tabellen, mit Mengen

Allgemeine Aussagen

★ Allgemeine Aussagen:

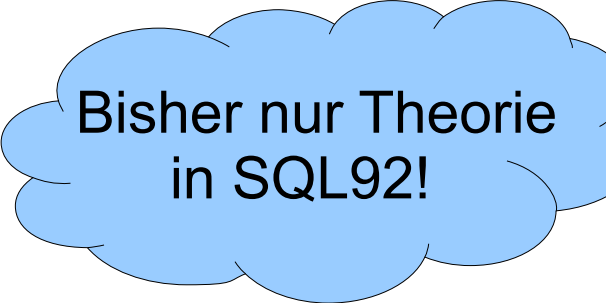
Bisher nur Theorie
in SQL92!

```
CREATE ASSERTION gueltige_kategorie
CHECK (NOT EXISTS (
    SELECT DISTINCT kategorie FROM
                                buchkategorien
EXCEPT
    SELECT DISTINCT kategorie FROM buch
))
```

Überprüfen, dass alle vorgesehenen Buchkategorien in buch vorkommen

Allgemeine Aussagen

- ★ Aussagen können mit `DROP ASSERTION` gelöscht werden
- ★ Unabhängig von Tabellendefinition
- ★ Gleiche Ausdruckskraft bietet `CHECK` in Tabellendefinition



Bisher nur Theorie
in SQL92!

Allgemeine Aussagen

★ Manchmal ist es notwendig, Aussagen erst verzögert zu prüfen

★ Beispiel:

$$\textit{Angebot} [ANr] \subseteq \textit{Medienartikel} [ANr]$$

$$\textit{Medienartikel} [ANr] \subseteq \textit{Angebot} [ANr]$$

Wenn beide Zusicherungen gelten, kann nie ein Datensatz eingefügt werden!

→ Prüfung einer Aussage kann verzögert werden

Allgemeine Aussagen

- ★ **Zusicherung erst nach Transaktionsende prüfen:**

```
CREATE TABLE angebot (  
    ...  
    CONSTRAINT fk_angebot  
        FOREIGN KEY (a_nr) REFERENCES  
        medienartikel (a_nr)  
        INITIALLY DEFERRED,  
    ...  
)
```

In MySQL/MariaDB nicht unterstützt!
Unterstützt in Oracle!

- ★ **Zusicherung sofort prüfen (Standardwert):**

```
SET CONSTRAINT fk_angebot  
    INITIALLY IMMEDIATE
```

Allgemeine Aussagen

- ★ Mit der Option `DEFERRABLE` bzw. `NOT DEFERRABLE` kann die Ausschaltung der Zusicherung erlaubt oder verboten werden
- ★ `INITIALLY DEFERRED` und `NOT DEFERRABLE` sind zusammen nicht erlaubt



In MySQL/MariaDB nicht unterstützt!
Unterstützt in Oracle!

Trigger

- ★ Ereignis-Aktionen-Regeln
- ★ Dieses Programm wird *Trigger* genannt
- ★ Bei Ausführung von DB-Operation wird eine Aktion ausgeführt
- ★ *Triggerfeuernde* Operationen sind i.d.R. Update-Operationen
- ★ In Triggern kann sowohl auf *neue* als auch *alte Daten* zugegriffen werden (*Differenzenrelationen*)

Trigger

Die { } | gehören
nicht zur Syntax!

```
★ CREATE TRIGGER trigger_name
    { BEFORE | AFTER }
    { INSERT | DELETE | UPDATE }
    ON tbl_name FOR EACH ROW
    trigger_stmt
```

- ★ Aktivierung vor oder nach INSERT, DELETE oder UPDATE
- ★ Ein Trigger gilt für eine spezielle Tabelle
- ★ Ein Trigger wird vor oder nach jedem Einfügen eines Datensatzes ausgeführt (FOR EACH ROW)

Trigger

- ★ `trigger_stmt` ist die Anweisung, die ausgeführt wird
- ★ Mit `NEW` und `OLD` kann auf neuen bzw. alten Datensatz zugegriffen werden

Trigger

★ Neue Tabelle:

```
CREATE TABLE anfangsgehalt (  
    genre varchar(20) NOT NULL ,  
    gehalt int(5) NOT NULL,  
    PRIMARY KEY (genre)  
) ENGINE = InnoDB;
```

★ Daten:

```
INSERT INTO anfangsgehalt VALUES  
('Drama', 100);
```

Trigger

★ Weitere Tabelle:

```
CREATE TABLE schauspieler (  
    name varchar(20) NOT NULL,  
    genre varchar(20) NOT NULL ,  
    gehalt int(5) NULL,  
    PRIMARY KEY (name)  
) ENGINE = InnoDB;
```


Trigger

★ Automatische Berechnung des Anfangsgehalts (BEFORE-Trigger)

```
CREATE TRIGGER trg_schauspieler1
  BEFORE INSERT ON schauspieler
  FOR EACH ROW
  SET NEW.gehalt=
    (SELECT gehalt
     FROM anfangsgehalt
     WHERE genre=NEW.genre)
```

Trigger

- ★ Einen Datensatz ohne Gehalt einfügen

```
INSERT INTO schauspieler  
  (name, genre)  
VALUES ( 'Hans', 'Drama' )
```

- ★ Gehalt wird durch Trigger ergänzt

| name | genre | gehalt |
|------|-------|--------|
| Hans | Drama | 100 |

Trigger

★ Begrenze Gehaltszuwachs auf 30% (BEFORE-Trigger)

```
CREATE TRIGGER trg_schauspieler2
  BEFORE UPDATE ON schauspieler
  FOR EACH ROW
  SET NEW.gehalt =
    IF (NEW.gehalt > 1.3 * OLD.gehalt,
        1.3 * OLD.gehalt,
        NEW.gehalt)
```

Trigger

★ Zu hohe Gehaltssteigerung setzen:

```
★ UPDATE schauspieler  
    SET gehalt='200'  
    WHERE name='Hans'
```

★ Gehalt durch Trigger angepasst:

| name | genre | gehalt |
|------|-------|--------|
| Hans | Drama | 130 |

Trigger


★ Weitere Tabelle:

```
★ CREATE TABLE gehaltaenderung (  
    name varchar(20) NOT NULL ,  
    aenderung int(5) NOT NULL ,  
    zeit timestamp NOT NULL  
) ENGINE = InnoDB;
```

Trigger

★ Protokollierung der Änderung vom Gehalt (AFTER-Trigger)

```
CREATE TRIGGER trg_schauspieler3  
  AFTER UPDATE ON schauspieler  
  FOR EACH ROW  
  INSERT INTO gehaltaenderung  
    (name, aenderung)  
  VALUES (NEW.name,  
          NEW.gehalt - OLD.gehalt)
```



timestamp wird
automatisch gesetzt.

Trigger

★ Anpassung des Gehalts:

```
★ UPDATE schauspieler  
    SET gehalt='150'  
    WHERE name='Hans'
```

★ Automatische Protokollierung:

| name | aenderung | zeit |
|------|-----------|---------------------|
| Hans | 20 | 2009-11-16 00:30:23 |

Trigger

- ★ Ein Trigger kann alternativ auch nur für eine Anweisung (nicht Zeile) ausgeführt werden:
FOR EACH STATEMENT



z.B. PostgreSQL

- ★ Es kann eine Alternativanweisung angegeben werden:
INSTEAD OF



z.B. PostgreSQL
und Oracle

Literatur

- ★ Vossen, Gottfried: Datenmodelle, Datenbank-sprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- ★ Lubkowitz, M: Webseiten programmieren und gestalten, Galileo Press, 2004
- ★ Oracle: MySQL 5.7 Reference Manual, <https://dev.mysql.com/doc/refman/5.7/en/>
- ★ Oracle: Database SQL Language Reference, <https://docs.oracle.com/database/121/SQLRF/toc.htm>