

Spaß mit Datenbanken

Kurzer Rückblick

Normalformen

- **1NF:** Ein Relationenschema ist in 1. Normalform, wenn dessen Wertebereiche atomar sind.
- **2NF:** Ein Relationenschema ist in 2. Normalform, wenn es in 1. Normalform ist und jedes Nichtschlüsselattribut voll funktional vom Primärschlüssel abhängig ist.
- 3NF: Ein Relationenschema ist in 3. Normalform, wenn es sich in 2. Normalform befindet, und kein Nichtschlüsselattribut vom Primärschlüssel transitiv abhängig ist.

Structured Query Language

- SQL basiert auf relationaler Algebra
- Ist eine **deklarative** Sprache
- Ist eine **mengenorientierte** Sprache

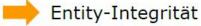
Constraints

- Stellen Korrektheit der Daten sicher
- Werden vom Datenbanksystem überwacht und sichergestellt
- Nicht passende Operationen werden zurückgewiesen

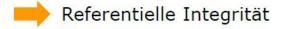
CONSTRAINT mit CHECK-Bedingung



CONSTRAINT mit PRIMARY KEY



CONSTRAINT mit FOREIGN KEY und REFERENCES



Beispiele

Professoren haben Rang C2, C3, C4

Matrikelnummer der Studierenden ist eindeutig

Anmeldung zur Übung nur bei aufrechter Inskription

Constraints

- NOT NULL NULL-Werte verboten
- CHECK(cond-exp) Attributspezifische Bedingung
- **UNIQUE** Jeder Wert darf nur einmal auftreten
- **PRIMARY KEY** Angabe eines Primärschlüssel
- FOREIGN KEY (attribute) REFERENCES TABLE (attribute) Angabe der referentiellen Integrität

Constraints - NOT NULL

- Schließt in Spalten den Wert "NULL" aus
- **NULL** steht für "Wert unbekannt" "Wert existiert nicht" etc.
- NULL kann in allen Spalten auftreten, außer bei NOT NULL und bei Schlüsselattributen.

BSP:

Name VARCHAR(30) NOT NULL

- Schlüssel identifiziert ein Tupel einer Relation
- Fremdschlüssel verweist auf ein Tupel einer in Beziehung stehenden Relation
- Referentielle Integritätsbedingungen entstehen zwischen Primär- und

Fremdschlüssel

```
Beispiel
CREATE TABLE Professoren (
 PersNr
              INTEGER PRIMARY KEY,
                                           Vatertabelle
              VARCHAR (30) NOT NULL,
 Name
 ...);
CREATE TABLE Vorlesungen (
 VorlNr
             INTEGER PRIMARY KEY,
 Titel
             VARCHAR (30),
 SWS
              INTEGER,
 gelesenVon INTEGER.
                                           Kindtabelle
 CONSTRAINT FK gelesenVon
   FOREIGN KEY (gelesenVon)
   REFERENCES Professoren (PersNr)
```

- Operationen auf der Kindtabelle sind immer unkritisch.
- Operationen auf der Vatertabelle (löschen, ändern des PK) muss entsprechend geregelt werden:
 - **CASCADE** Übernimmt die Änderung rekursiv für alle Tupel mit zugehörigem Fremdschlüssel.
 - SET NULL Setzt den Fremdschlüssel ggf. auf NULL
 - RESTRICT Verbiete operation, wenn Fremdschlüsselverweise vorhanden sind.
 - SET DEFAULT Setzt den Fremdschlüssel auf den Default-Wert

ON DELETE oder ON UPDATE

```
Beispiel
CREATE TABLE Professoren (
 PersNr
             INTEGER PRIMARY KEY,
 Name
             VARCHAR (30) NOT NULL,
 ...);
CREATE TABLE Vorlesungen (
 VorlNr
             INTEGER PRIMARY KEY,
 Titel
             VARCHAR (30),
 SWS
             INTEGER,
 gelesenVon INTEGER,
 CONSTRAINT FK gelesenVon
   FOREIGN KEY (gelesenVon)
                                         Referentielle Aktion -
   REFERENCES Professoren (PersNr)
                                         wird auf Vorlesungen-Tabelle
   ON DELETE SET NULL
                                         (=Kindtabelle) ausgeführt
```

Ändern der Tabellenstruktur

Nachträgliche Änderungen mit ALTER TABLE

```
BSP:
ALTER TABLE Professoren ADD (Titel VARCHAR(30));
ALTER TABLE Professoren MODIFY (Titel VARCHAR(40));
ALTER TABLE Professoren DROP COLUMN Titel;
```

Ändern der Tabellenstruktur

Nachträgliche Löschung mit DROP TABLE

```
BSP:
DROP TABLE Professoren;
DROP TABLE Professoren CASCADE;
```

Mit CASCADE werden 'abhängige' Objekte ebenfalls gelöscht

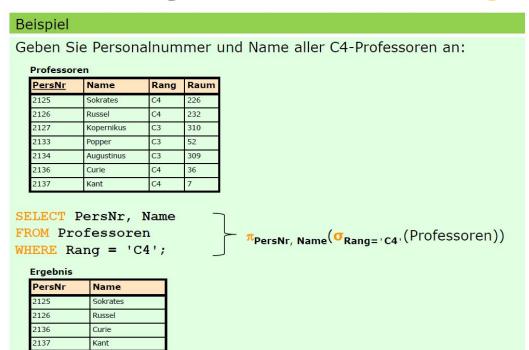
Jede Bedingung **muss** bei Veränderungen am Datenbestand überprüft werden (Rechenzeit)

- Anfrage: Folge von Operationen
 - Berechnet Ergebnisrelation aus Basisrelation
- Benutzer formuliert "Was will ich haben?", und nicht "Wie komme ich an das ran?"
- Ergebnis einer Anfrage ist wieder eine Relation und kann wieder als Eingabe für die nächste Anfrage verwendet werden
- Syntaktisch korrekte Anfragen können nicht zu Endlosschleifen oder unendlichen Ergebnisse führen

Keywords

- SELECT: Projektionsliste, Abfrage von Daten
- FROM: zu verarbeitende Relation
- WHERE: Selektions-, oder Verbundbedingungen
- GROUP BY: Gruppierung für Aggregatfunktionen
- HAVING: Selektionsbedingungen f
 ür Gruppen
- ORDER BY: Sortierung der Ergebnisrelation

SELECT attribute FROM tabelle WHERE bedingungen



Beispiel

SELECT *
FROM Professoren;

Professoren

<u>PersNr</u>	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

DISTINCT: Ergebnismenge ist frei von Duplikaten

Duplikatelimination

Geben Sie alle Rangbezeichnungen für Professoren ohne Duplikate aus.

SELECT DISTINCT Rang
FROM Professoren;



Ergebnis
Rang
C4

Beispiel ohne DISTINCT:

Keine Duplikatelimination

SELECT ALL Rang
FROM Professoren;



Ergebnis

Rang
C4
C4
C3
C3
C3
C3
C4
C4

ALIASNAME:

- Benennt Spalte in Ergebnisrelation.
- Wird direkt nach dem Spaltennamen angegeben.
- Keyword: AS

Spaltenüberschrift

SELECT PersNr AS Personalnummer, Name Familienname FROM Professoren;

Ergebnis	
PersNr Personalnummer Familienname	Name
2125 Sokrates	
2126 Russel	
2127 Kopernikus	
Popper	
2134 Augustinus	
2136 Curie	
2137 Kant	

Sortierung:

- Klausel steht am Ende der Anfrage.
- Keyword: ORDER-BY

Beispiel

SELECT PersNr, Name, Rang
FROM Professoren
ORDER BY Rang DESC, Name ASC;

Ergebnis

PersNr	Name	Rang
2136	Curie	C4
2137	Kant	C4
2126	Russel	C4
2125	Sokrates	C4
2134	Augustinus	C3
2127	Kopernikus	C3
2133	Popper	C3