Datenbank-Systeme

Datenbank Schema Erstellung mit Structured Query Language (SQL)

Internationaler Frauenstudiengang Informatik

WiSe 2017/18

Renzo Kottmann

Kontakt:

- mail
- linkedin:http://www.linkedin.com/in/renzokottmann
- twitter: @renzokott

Wiederholung

- Entity Relationship Modellierung (ERM)
  - Entität, Attribute, Beziehungen
  - Wichtige Gute Diskussion zu: Keys
- Labor: Fertige Arbeitsumgebung
- Erster Blick auf SQL

Relationales Datenbankmodell

Generische Datenstruktur

- Relationen mit eindeutigen Namen
  - jede Relation ist eine Menge von Tupeln (Datensätzen) gleichen Typs
  - Die Struktur ist insofern generisch, als die Relationen und ihre Attribute (Spalten) beliebig gewählt werden können bzw. beim Einrichten der Datenbank angegeben werden müssen.

Operatoren

- relationale Algebra
- Daten
  - eintragen
  - ändern
  - löschen
  - abfragen
  - ableiten

Integritätsbedingungen

- Bestimmung von ein-eindeutigen Tupeln
- Einschränkungen vom Wertbereich bestimmter Datentypen

https://de.wikipedia.org/wiki/Datenbankmodell

Structured Query Language (SQL)

SQL ist eine Datenbanksprache

1. zur Definition von Datenstrukturen/Modellen
2. zum Bearbeiten (Einfügen, Verändern, Löschen)
3. zum Abfragen von darauf basierenden Datenbeständen
4. zur Rechtevergabe

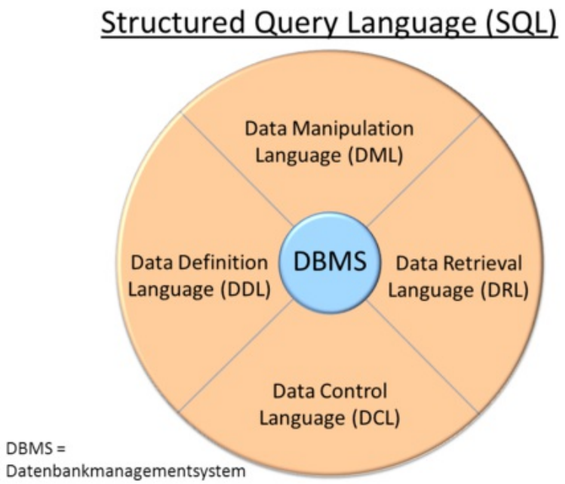
SQL Eigenschaften

- basiert auf relationaler Algebra
- an English angelehnt
- Deklarativ und funktional
- Fast alle Datenbanken verstehen SQL
- Standardisiert
  - PostgreSQL hat einer der besten Umsetzungen

Datenbank Anlegen

- `CREATE DATABASE`
  - Dokumentation

SQL Überblick



- DDL = Data Definition Language:

  Definition des Datenbankschemas

- DML = Data Manipulation Language:

  Ändern, Einfügen, Löschen und lesender Zugriff

- DRL = Data Retrievel Language

  Nicht standardisierte Bezeichnung des `SELECT` aus DML

- DCL = Data Control Language:

  Rechteverwaltung und Transaktionskontrolle

DDL: Create Table


```
CREATE TABLE teilnehmerin (
--Spalten Name Datentyp,
  vorname text,
  nachname text,
  matrikel_nr integer,
  email text,
  semester integer
);
```

s. http://www.postgresql.org/docs/9.6/interactive/ddl-basics.html und http://www.postgresql.org/docs/9.6/interactive/sql-createtable.html

SQL in PostgreSQL

- Scheinbar viele SQL Kommandos
  - Sehr guter Überblick in der PostgreSQL Dokumentation
- Die meisten sind **DDL** Kommandos
  - `CREATE`, `DROP` oder `ALTER`
    - Jeweils ein Eintrag pro Datenbank-Objekt
    - Folgen dem selben Syntax Schema

Integritätsbedingungen

> "Integritätsbedingungen beschreiben Annahmen, die über die Daten getroffen werden, beispielsweise ein bestimmter Datentyp, ein Wertebereich oder eine Abhängigkeitsbeziehung zwischen zwei Objekten."

Ziatat Wikipedia: Integritätsbedingung

Constraints

> Mit Constraints (deutsch „Einschränkung“) werden in diversen Programmiersprachen Bedingungen definiert, die zwingend vom Wert einer Variablen erfüllt werden müssen, damit der Wert ins System übernommen werden kann. In Datenbanksystemen finden Constraints rege Anwendung um den Wertebereich (Domain) von Attributen einzuschränken und Werte auf deren Zulässigkeit zu überprüfen.

modifiziert von Wikipedia: Constraint

Datentypen sind erste Constraints

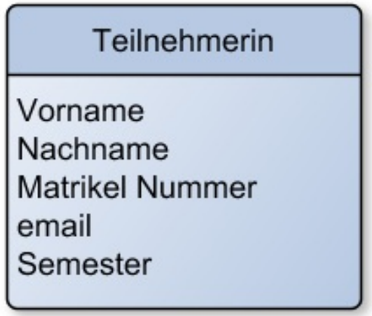
- PostgreSQL stellt viele Datentypen zur Verfügung
- Auch eigene Datentypen können definiert werden
- Verwende Datentypen so strikt wie möglich

```
CREATE table teilnehmerin (
  --Spaltenname Datentyp,
  vorname text,
  nachname text,
  matrikel_nr integer PRIMARY KEY,
  email text,
  semester integer
);
```

Weitere Contraints für Datenintegrität

- Primary Keys
- NULL or NOT NULL Constraints
- DEFAULT VALUES
- CHECK Constraints
- Unique Constraint

DML: Daten Einfügen




```
INSERT INTO teilnehmerin
  (vorname, nachname, matrikel_nr, email, semester)
VALUES
  ('renzo','kottmann',007,'renzo@007.bond', 0);
```

s. http://www.postgresql.org/docs/9.6/interactive/dml-insert.html und http://www.postgresql.org/docs/9.3/interactive/sql-insert.html

Primary Keys

- Die Eindeutigkeit jedes Eintrags wird durch den `PRIMARY KEY` Ausdruck sichergestellt

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname text,
  nachname text,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text,
  semester integer
);
```

NULL or NOT NULL Constraints

- Implizit ist jedes Attribut einer Tabelle `NULL`
  - d.h. kann leer sein
- Nicht bei `PRIMARY KEYS`
- oder Schlüsselwort `NOT NULL`

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname text,
  nachname text,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text NOT NULL,
  semester integer
);

Die Verwendung von `NOT NULL`
implementiert hier die Anforderung:
"Zu jeder Teilnehmerin muss es
eine E-mail geben."
```

Für welche Attribute ist NOT NULL noch sinnvoll?

Weitere Massnahmen zur Gestaltung der Datenintegrität

- DEFAULT VALUES
- CHECK Constraints
- Unique Constraint

Default Values Constraint

- Für jedes Attribut kann man einen Standard-Wert festlegen
- der Standwert wird dann eingetragen, wenn
  - kein Wert angegeben wurde
  - oder explizit auf `DEFAULT` gesetzt wurde

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname text,
  nachname text,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text NOT NULL,
  semester integer DEFAULT 3
);


Die Verwendung von `DEFAULT`
implementiert hier die Anforderung:
"Bei jeder Teilnehmerin ist das Standard
```

Default Values Constraint

```
INSERT INTO teilnehmerin (vorname, nachname, matrikel_nr, email)
VALUES ('renzo','kottmann',007,'renzo@007.bond');
```

führt zu einem Eintrag mit:

vorname	nachname	matrikel_nr	email	semester
renzo	kottmann	7	renzo@007.bond	3

Check Constraint

- Ein Wert in ein oder mehreren Spalten muss einer boolschen Funktion entsprechen
  - es muss `TRUE` ergeben

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname text CHECK ( vorname != '' ),
  nachname text,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text NOT NULL CHECK ( email ~ '
  semester integer DEFAULT 3
);
```

Unique Constraint

- Alle Werte einer- oder meherer Spalte(n) müssen eindeutig sein
- Damit werden weitere Schlüssel implementiert

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname text CHECK ( vorname != '' ),
  nachname text,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text NOT NULL CHECK ( email ~ '.
  semester integer DEFAULT 3
);
Jede Teilnehmerin muss eine andere E-Mai
```

Logisch gesehen: Primary Key vs. Unique

- Ein `PRIMARY KEY` ist nichts anderes als ein `UNIQUE NOT NULL`
- D.h. es kann mehrere Schlüssel geben, aber nur einer wird als `PRIMARY KEY` gewählt

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname text CHECK ( vorname != '' ),
  nachname text,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text NOT NULL CHECK ( email ~ '
  semester integer DEFAULT 3
);
Jede Teilnehmerin muss eine andere E-Mai
```

SQL Kommando: CREATE DOMAIN

- Definition von eigenen Datentypen basierend of existierenden

```
CREATE DOMAIN person_name AS text CHECK (
  VALUE ~ '^[A-Z][a-z]*'
);
```

```
CREATE TABLE teilnehmerin (
  -- Spaltenname Datentyp,
  vorname person_name CHECK ( vorname != '' ),
  nachname person_name,
  -- Simpler Primary Key
  matrikel_nr integer PRIMARY KEY,
  email text NOT NULL CHECK ( email ~ '.*@.*' ) UNIQUE,
  semester integer DEFAULT 3
);
```

Danke fuer die Zusammenarbeit