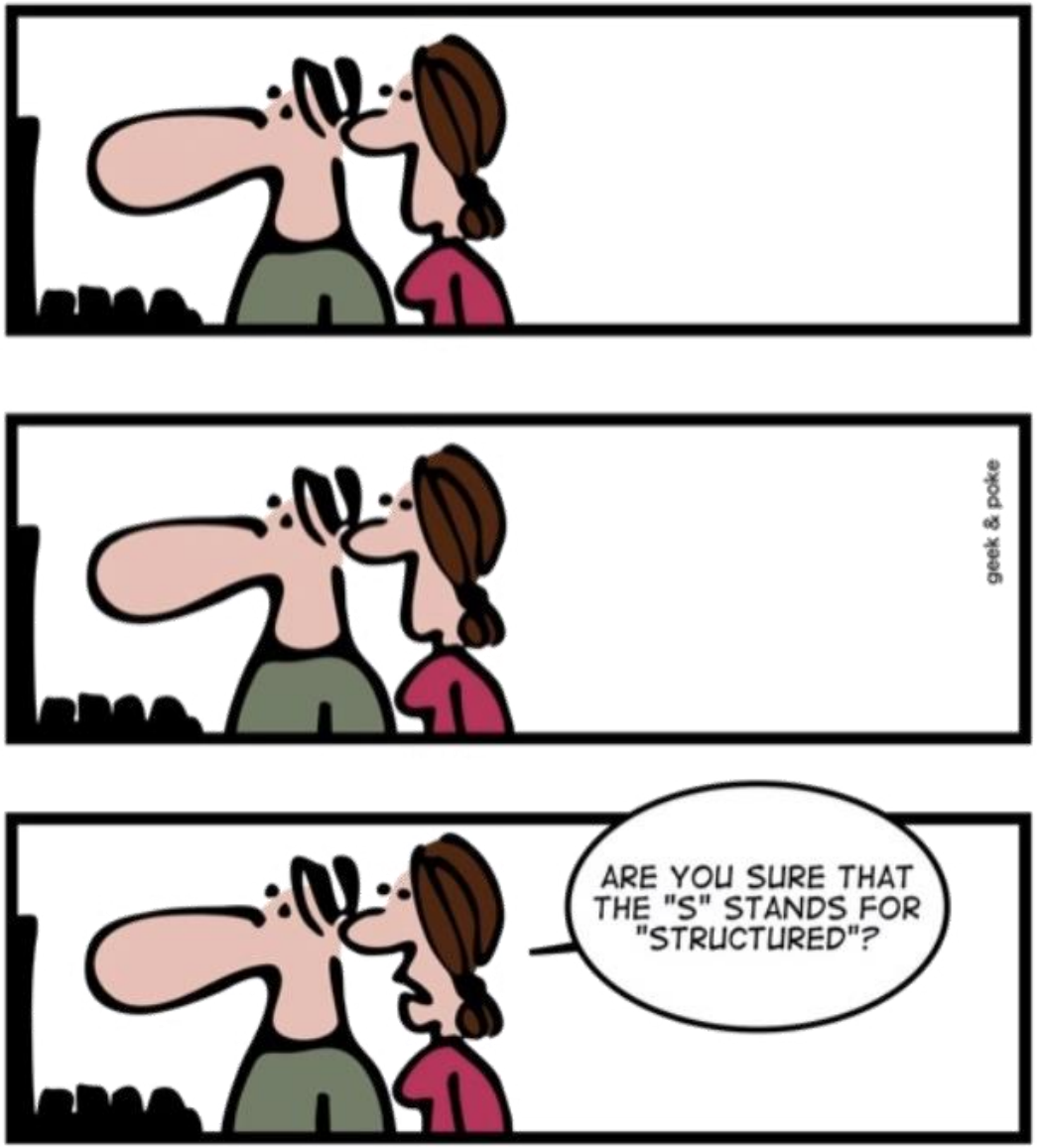
Datenbanken I (T2INF2004)

Foliensatz 5: SQL 1/3
(Einführung Apache Derby und DDL)

Uli Seelbach, DHBW Mannheim, 2023

Foliensatz freundlicherweise zur Verfügung gestellt von Mirko Schick

SQL - Structured Query Language



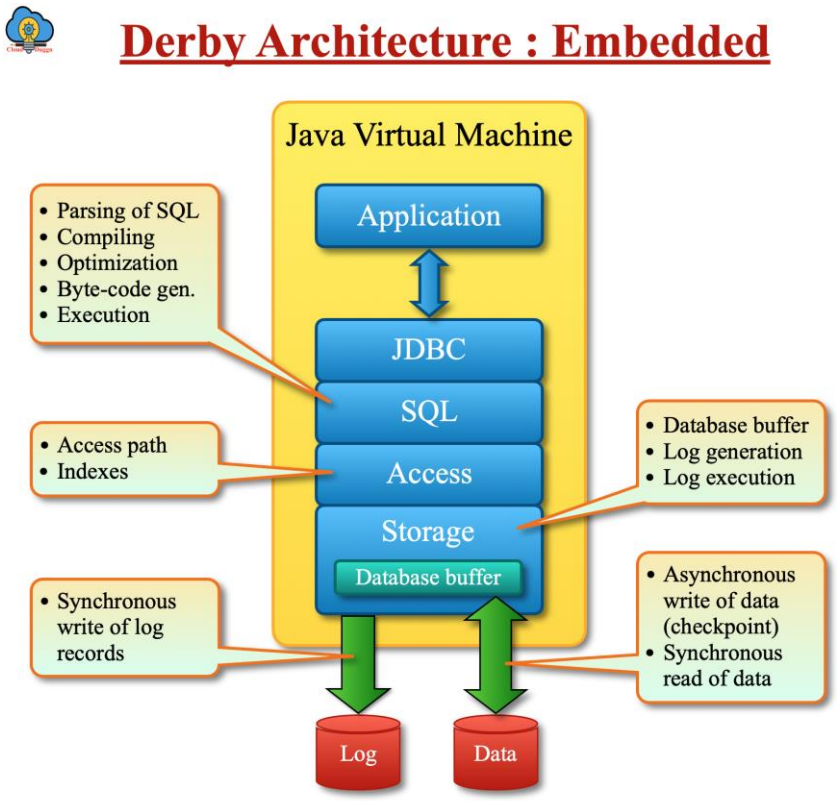
Apache Derby
Zugriff

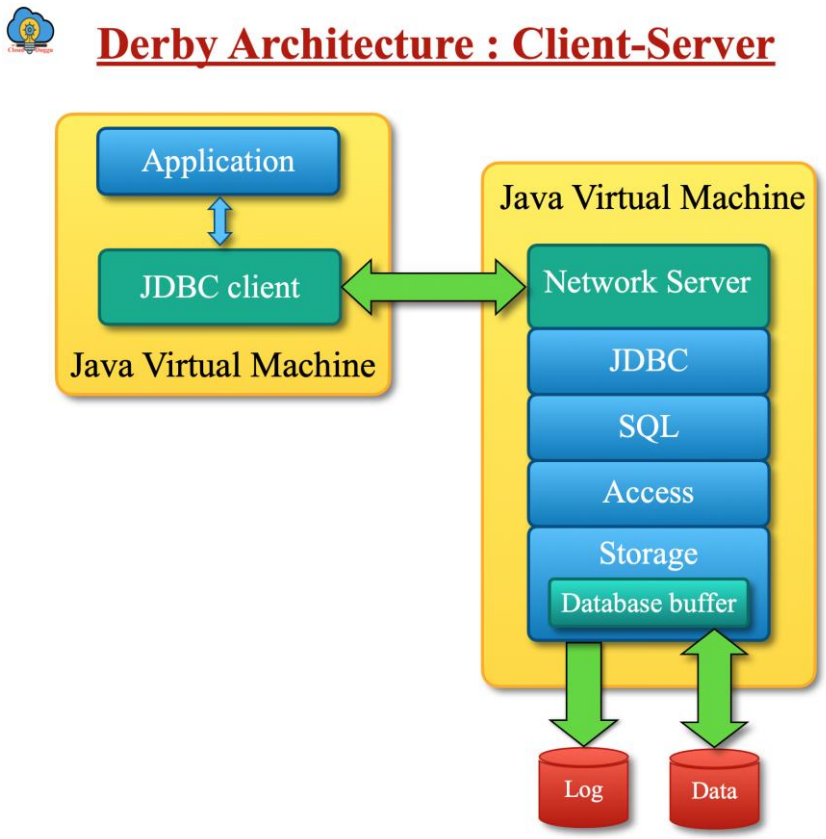
- Apache Derby läuft entweder „embedded“ (innerhalb einer JVM gemeinsam mit der Anwendung)...
- ... oder als Server-Prozess in einer separaten JVM

- In Client/Server-Konfiguration über TCP/IP (DRDA-Protokoll) ansprechbar, z.B. über JDBC oder auch ODBC
- Kommandozeilentool: ij
- GUI-basierte Tools (Eclipse, DBVisualizer, ...)

Apache Derby

Aufbau





SQL-Datentypen

Derby

- Obwohl SQL standardisiert ist, hat fast jedes RDBMS andere Datentyp-Bezeichnungen und einige Eigenarten
- Wichtigste Datentypen in Derby:

Datentyp	Beschreibung
SMALLINT	Kleine Ganzzahl (2 Byte)
INTEGER	Ganzzahl (4 Byte)
BIGINT	Ganzzahl (8 Byte)
REAL	Gleitkommazahl (4 Byte)
DOUBLE	Gleitkommazahl (8 Byte)
DECIMAL(p, s) / NUMERIC	Dezimalzahl
BOOLEAN	Wahrheitswert
CHAR, VARCHAR	Zeichenkette fest / variabel lang
DATE, TIME, TIMESTAMP	Datum, Uhrzeit, Zeitpunkt
BLOB, CLOB, XML	Large objects, XML

SQL-Datentypen

CHAR vs. VARCHAR

- CHAR oder CHARACTER
  - Für Zeichenketten fester Länge
  - CHAR(30) bedeutet exakt 30 Byte (nicht 30 Zeichen!)
  - Nur CHAR ist gleichbedeutend mit CHAR(1)
  - Bei kürzeren Zeichenketten wird mit Leerzeichen aufgefüllt
  - Bei Vergleichen wird die kürzere Zeichenkette “verlängert”
- VARCHAR oder CHAR VARYING oder CHARACTER VARYING
  - Für Zeichenketten variabler Länge
  - Längenfeld wird intern mitgespeichert
  - VARCHAR(30) bedeutet höchstens 30 Bytes lang
  - Bei Vergleichen werden Leerzeichen am Ende ignoriert

SQL-Datentypen

Oracles „Varchar2“ und „Char“

- VARCHAR2
  - Für Zeichenketten unbekannter, aber maximal bestimmbarer Länge
  - VARCHAR2(30) bedeutet maximal 30 Byte (Bei single byte charset also auch 30 Zeichen) lang.
    - Wenn weniger Zeichen, werden auch nur 30-x Zeichen gespeichert
  - Leere Zeichenketten werden entgegen SQL-Standard wie NULL behandelt
  - Warum VARCHAR2?
    - „VARCHAR“ ist für die Zukunft reserviert um eventuell einmal dem Standard zu genügen?

- CHAR
  - Für Zeichenketten bekannter gleicher Länge, schneller als VARCHAR2 (*)
  - CHAR(30) bedeutet genau 30 Bytes (oder Zeichen) lang
  - Trailing space padded

(*) vermutlich unmessbar

SQL-Datentypen

DECIMAL(p, s)

- Speichert Dezimalzahlen
- Insgesamt p Stellen, davon s Nachkommastellen (“precision” und “scale”)
- Einzig brauchbarer Datentyp für Geldbeträge u.dgl.

SQL
Statements

SQL in Derby (und anderen RDBMS) gliedert sich in folgende Bereiche:

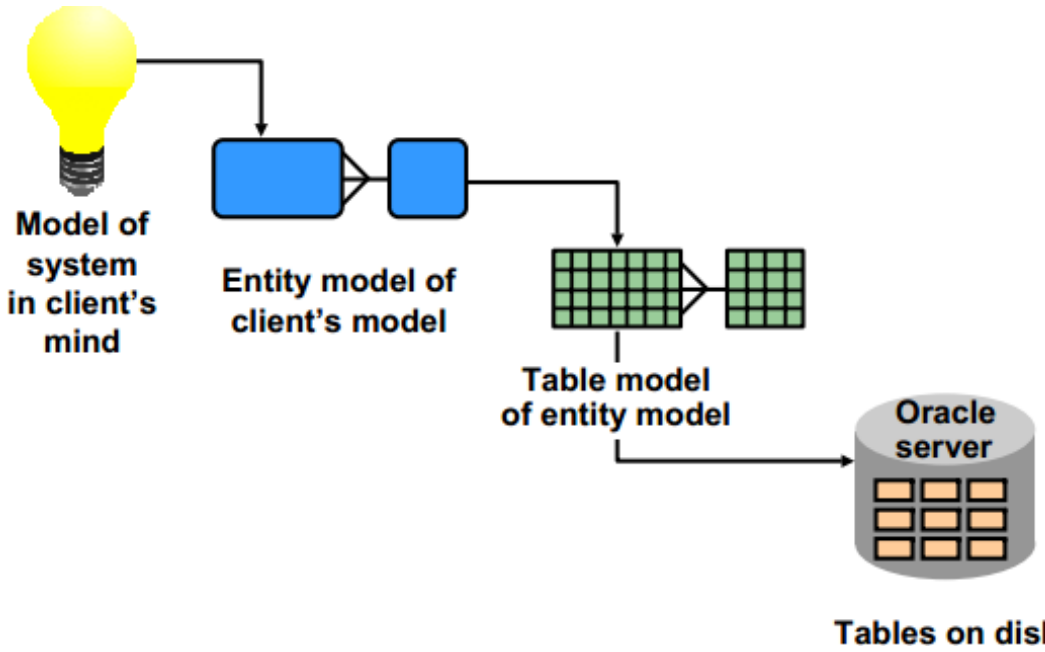
SELECT	Data manipulation language (DML)
INSERT	
UPDATE	
DELETE	
MERGE	
CREATE	Data definition language (DDL)
ALTER	
DROP	
RENAME	
GRANT	Data control language (DCL)
REVOKE	

SQL

Data Definition Language :: DDL

- Auf dem Weg vom konzeptionellen Modell bis hin zu den physischen Tabellen benötigen wir ein eigenes Sprachkonstrukt, welches auch RDBMS-spezifische verarbeiten kann: einen besonderen SQL-Dialekt zum Anlegen und Ändern von Datenbankobjekten wie Tabellen und Beziehungen: **Data Definition Language (DDL)**

- Es gibt viele Objekte in Datenbanken:
  - Table
  - View
  - User
  - Schema
  - Index
  - Constraint
  - Function
  - ...



SQL - DDL

Create Table

Pflichtangaben für das Anlegen einer Tabelle

- Name der Tabelle
- Spaltennamen
- Datentyp jeder Spalte
- „Größe“ jeder Spalte (Anzahl Zeichen etc.)
- Auslassen dieser Information ist manchmal eine implizite Angabe

Erweiterte Angaben

- Constraints
- Default-Werte für Spalten
- Kommentare zu Tabellen und deren Spalten
- „technische“ Besonderheiten (nicht Bestandteil der Vorlesung)

SQL - DDL

Constraints

Integrity constraint

- Primary key (entity integrity constraint)
- Foreign key (referential integrity constraint)

Propagation constraint

- Könnte man ansehen als speziellen integrity constraint, da Logikerweiterung
- Was macht das DBMS, wenn ein Fremdschlüsselwert geändert wird?

Value constraint

- Check-Constraint (Datenvalidierung)
- Unique-Constraint (nur eindeutige Attributwerte)
- NOT NULL (Attributwert darf nicht NULL sein

http://docs.oracle.com/cd/B19306_01/server.102/b1420_0/clauses002.htm

SQL - DDL

Create Table

- Ganz grob:

```
CREATE TABLE [schema_name.]table_name (
   col1_name datatype [DEFAULT col1_default] [constraint_list],
   … ,
   coln_name datatype [DEFAULT coln_default] [constraint_list],

   [CONSTRAINT tblconstraint1_name] constraint_type (col1,…,
   coln])
);
```

- Angabe der Constraints „inline“ auf Spaltenebene, falls nur eine Spalte betreffend oder aber auch separat definierbar auf Tabellenebene „out of line“ falls z.B. mehrere Spalten betroffen sind (geht nicht für NOT-NULL-Constraints)
- Genau beschrieben im Handbuch (refderby.pdf)

SQL
Schemata

- Die meisten RDBMS gruppieren Datenbankobjekte in **Schemata**
- Ein Schema ist ein Namensraum für Objekte
- Häufig werden über Schemata auch Benutzerrechte geregelt
- Ein Tabellenname kann „qualifiziert“ (schema.tabelle) oder „unqualifiziert“ angegeben werden (ebenso auch andere Objekte wie z.B. Sequences)
- Bei unqualifizierten Namen wird implizit durch das CURRENT SCHEMA qualifiziert (Default in Derby: „APP“)
- Systemtabellen liegen in Derby im Schema „SYS“

Derby:

- `CREATE SCHEMA` *`name`* zum Anlegen eines Schemas
- `DROP SCHEMA` *`name`* `RESTRICT` zum Löschen
- `SET CURRENT SCHEMA =` *`name`* legt das aktuelle Schema fest

SQL - DDL

Create Table :: BEISPIEL

```
CREATE TABLE student(
  StudentId   CHAR(5),
  Nachname    VARCHAR(35),
  Studiengang DECIMAL(3)
);
```

SQL - DDL

Constraints: Primary Key

- Kann nur einmal für eine Tabelle vergeben werden
- Darf keine NULL-Werte in den Datenfeldern beinhalten
  Nicht für alle Datentypen anwendbar
- Besteht aus 1 bis n Spalten
  - Wenn es sich um 2 oder mehr Spalten handelt (composite primary key), muss die Angabe auf Tabellenebene erfolgen
- Beispiel (Syntax müssen Sie nicht schreiben, aber lesen können):
  - column level:
    ```
    DeptId SMALLINT CONSTRAINT dept_deptid_pk PRIMARY KEY
    ```
  - table level:
    ```
    CONSTRAINT dept_deptid_pk PRIMARY KEY(DeptId)
    ```
- NOT NULL in Derby automatisch angenommen, wenn PK-Constraint

SQL - DDL

Constraints: Foreign Key

- Davon kann es mehrere pro Tabelle geben
- Besteht je aus 1 bis n Spalten, die auf den PK einer anderen Tabelle verweisen
  - Wenn es sich um 2 oder mehr Spalten handelt, muss die Angabe auf Tabellenebene erfolgen
  - Spaltennamen beider Tabellen müssen nicht identisch sein, Datentyp muss aber stimmen. Tupel referenzierender Tabelle muss in referenzierter Tabelle vorhanden sein oder Tupel in referenzierender Tabelle ist NULL
- Die referenzierte Tabelle und ihr PK muss schon existieren
- Beispiel (Syntax müssen Sie nicht schreiben, aber lesen können):
  - column level:

```
FacultyId char(5) CONSTRAINT student_facultyid_fk
REFERENCES faculty(FacultyId)
```

  - table level:

```
CONSTRAINT student_facultyid_fk
FOREIGN KEY(FacultyId) REFERENCES faculty(FacultyId)
```

SQL - DDL

Foreign Key Constraints: Propagation (Referenzaktionen)

Was passiert, wenn in der „Elterntabelle“ ein Datensatz gelöscht wird?

Möglichkeiten:

- Löschen wird nicht erlaubt (ON DELETE RESTRICT, ON DELETE NO ACTION)
- Löschen wird erlaubt und abhängige Datensätze werden ebenfalls gelöscht (ON DELETE CASCADE)
- Löschen wird erlaubt und in abhängigen Datensätzen wird der Fremdschlüssel auf NULL gesetzt (ON DELETE SET NULL)
- Wann ist welche Regel angebracht? Denken Sie auch ans E/R-Modell

Was passiert, wenn in der „Elterntabelle“ ein Primärschlüssel geändert wird?

- Viele RDBMS unterstützen lediglich ON UPDATE RESTRICT, d.h. eine Änderung ist nicht erlaubt

SQL - DDL

Constraints: NOT NULL

- Stellt sicher, dass ein Datenfeld keinen "unbekannten” Wert hat, also nicht NULL ist
- Leerzeichen, Leerstring oder die numerische 0 sind **nicht dasselbe** wie NULL
- Ausnahme: Eine leere Zeichenkette beim Oracle-Datentyp VARCHAR2 wird gleich behandelt wie NULL
- Nur auf Spaltenebene definierbar
- Beispiel:

```
Name VARCHAR(50) CONSTRAINT faculty_name_nn NOT NULL
```
oder
```
Name VARCHAR(50) NOT NULL
```

SQL - DDL

Constraints: Unique

- Stellt sicher, dass die Datenfelder des entsprechenden Attributs paarweise verschieden sind.
  NULLs gelten dabei als paarweise verschieden (RDBMS-abhängig)
- Nicht für alle Datentypen anwendbar
- Wenn Spalten schon ein PK sind, ist Definition als gesonderter Unique-Constraint nicht erlaubt
- Beispiel (Syntax müssen Sie nicht schreiben, aber lesen können):

  column level:

```
DeptName VARCHAR(12) [CONSTRAINT dept_deptname_uk] UNIQUE
```

  table level:

```
[CONSTRAINT dept_deptname_uk] UNIQUE(DeptName)
```

SQL - DDL

Constraints: Check

- Stellt sicher, dass die Datensätze einer definierten Bedingung genügen und erlaubt somit nur Datenänderungen, für die das Check-Prädikat zu „true“ auswertet
- Beispiel (Syntax müssen Sie nicht schreiben, aber lesen können):

  column level (darf sich nur auf die eine Spalte beziehen):

```
DeptId DECIMAL(2)
       CONSTRAINT dept_deptid_ck
       CHECK(DeptId BETWEEN 10 and 99)
```

  table level:

```
CONSTRAINT dept_deptid_ck
     CHECK ((DeptId >= 10) and (DeptId <= 99))
```

SQL - DDL

Deferrable Constraints

- Speziell Foreign Key Constraints können beim Einfügen oder Ändern „lästig“ sein:
  - Ein abhängiger Datensatz kann nur dann eingefügt werden, wenn es den „Eltern“-Datensatz bereits gibt
  - Zyklische Abhängigkeiten sind damit gar nicht möglich
- Abhilfe in Derby: „Deferrable Constraints“: Der Constraint wird erst am Transaktionsende geprüft
- Angabe durch Klausel INITIALLY DEFERRED in der Definition des Constraints
- NOT NULL Constraints sind nicht deferrable

- Achtung: aus einem Tool heraus funktioniert das nur, wenn Sie die Transaktionsgrenze selbst setzen können (Autocommit ausschalten)

SQL - DDL

Create Table, Namensgebung für Constraints

- Die Namensgebung für DB-Objekte sollte einheitlich durchgeführt werden
- Gängige Konvention ist folgende:
  - ***<table name>_<column name>_<constraint type>***
  - table name: Name der betreffenden Tabelle, die direkt beeinflusst wird
  - column name: Spalte, auf die der Constraint Einfluss hat – wird manchmal leer gelassen, wenn komplette Tabelle betroffen
  - constraint type: Abkürzung für den Typen
- Contraint-Typen-Abkürzungen:
  - Primary Key pk
  - Foreign Key fk
  - Unique uk
  - Check ck
  - Not Null nn
- NN-Costraints werden selten “manuell” benannt – Benennung wird dem DBMS überlassen

SQL - DDL

Create Table :: BEISPIEL

```
CREATE TABLE Student(
  StudentId    CHAR(5)      PRIMARY KEY,
  Nachname     VARCHAR(35) CONSTRAINT student_nachname_nn NOT NULL,
  Vorname      VARCHAR(35) NOT NULL,
  PLZ          CHAR(5)      NOT NULL DEFAULT '68161',
  Geburtstag   DATE         NOT NULL,
  Studiengang DECIMAL(3),

  CONSTRAINT student_un
      UNIQUE(Nachname, Vorname, Geburtstag),
  CONSTRAINT student_StudentId_ck
       CHECK(StudentId = upper(StudentId)),
  CONSTRAINT student_Studiengang_fk
     FOREIGN KEY (Studiengang) REFERENCES Studiengang(ID)
          ON DELETE RESTRICT
);
```

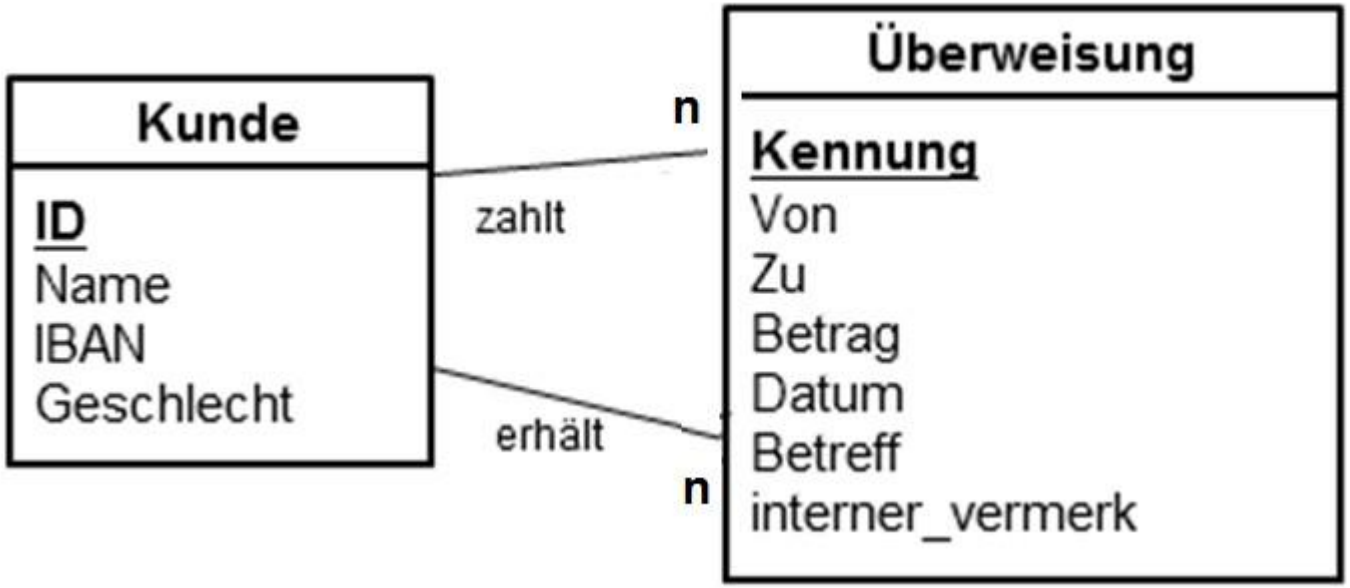
SQL - DDL

COMMENTs *(leider nicht in Apache Derby)*

- Es ist immer sinnvoll, Spalten und Tabellen näher zu beschreiben
- Neben der Benennung der Objekte ist dies eine der effektivsten Arten der Dokumentation
- Dazu müssen die Objekte jedoch erst einmal angelegt sein
- Für Tabellen gilt:
  - COMMENT ON TABLE [schema.]tabelle IS ‘Kommentar‘;
- Für Spalten gilt:
  - COMMENT ON COLUMN [schema.]objekt.spalte IS ‘Kommentar‘;

SQL - DDL

Create Table :: ÜBUNG



?

Sie sind Geldwäschebeauftragter und überwachen die institutsinternen Kontobewegungen. Sie haben eine DB mit allen Kunden der XY-Bank und allen Überweisungen von Kunden der XY-Bank an Kunden der XY-Bank. Wandeln Sie das oben abgebildete und halbfertige logische Schema in ein physisches Schema in Form von SQL um. Überlegen Sie sich sinnvolle Datentypen, Constraints und Kommentare.

Bearbeitungszeit: 15 Minuten

SQL - DDL

Create Table :: LÖSUNG

```
CREATE TABLE kunde (
  id          integer      NOT NULL
                           CONSTRAINT kunde_pk PRIMARY KEY,
  Name        varchar(50)  NOT NULL,
  iban        varchar(34)  NOT NULL
                           CONSTRAINT kunde_iban
                           CHECK (LENGTH(iban) >= 12),
  geschlecht  char         NOT NULL
                           CONSTRAINT kunde_geschlecht
                           CHECK (geschlecht IN ('M', 'W', 'D'))
);
```

SQL - DDL

Create Table :: LÖSUNG

```
CREATE TABLE überweisung (
    Kennung BIGINT          NOT NULL CONSTRAINT überweisung_pk
                                     PRIMARY KEY,
    von     INTEGER         NOT NULL CONSTRAINT überweisung_fk_von_kunde
                                     REFERENCES kunde,
    zu      INTEGER         NOT NULL CONSTRAINT überweisung_fk_zu_kunde
                                     REFERENCES kunde,
    betrag  decimal(14, 2) NOT NULL CONSTRAINT überweisung_betrag
                                     CHECK (betrag > 0.0),
    datum   date            NOT NULL,
    betreff varchar(140),
    interner_vermerk clob
);
```

SQL - DDL

Wichtige DDL-Kommandos für Tabellen:

- Spalten hinzufügen
  ALTER TABLE tabellenname ADD (spaltenname datentyp);
- Constraints hinzufügen
  ALTER TABLE tabellenname ADD
  [CONSTRAINT constraint_name] constraint_type (column, ...);
- Spalten ändern
  ALTER TABLE tabellenname ALTER COLUMN spaltenname SET DATA TYPE neuerTyp;
- Spalten / Constraints löschen
  ALTER TABLE tabellenname DROP spaltenname | CONSTRAINT constraint;
- Tabellen löschen
  DROP TABLE tabellenname;
- Tabellen leeren und Speicher freigeben
  TRUNCATE TABLE tabellenname;
- Tabelle umbenennen
  RENAME TABLE tbl_name TO new_tbl_name;

SQL - DDL

Benennung von Objekten

- Derby kennt “Ordinary identifiers” und “Delimited identifiers”
- Für Ordinary identifiers gilt:
  - Muss mit einem Buchstaben beginnen und darf nur Buchstaben, Ziffern und den Unterstrich enthalten
  - „Buchstaben“ im Sinne des Unicode-Standards
  - Werden automatisch nach UPPERCASE konvertiert
  - Darf kein reserviertes Wort sein
- Delimited identifiers werden mit doppelten Hochkommata (") eingeschlossen
  - Hier sind alle Zeichen erlaubt
  - Reservierte Worte sind ebenfalls erlaubt
  - Nützlich zur „lesbaren“ Benennung von Spalten, z.B. für Reports
- Objektname muss innerhalb des Schemas eindeutig sein

SQL - DDL

weitere wichtige Datenbankobjekte

- Schemata
  - Gruppieren andere Datenbankobjekte in einen eigenen Namensraum
- Indizes und Views
  - Das brauchen wir jetzt noch nicht
  - Wird in den nächsten Vorlesungen betrachtet
- Synonyms
  - Alternative Namen für Objekte
- Sequences
  - Sequenzgeneratoren (in MySQL gibt es diese nicht)
- Roles
  - Dienen zur vereinfachten Administration von Rechten
- Procedures und Functions
  - Benutzerdefinierte Prozeduren und Funktionen

Übung

Im Hörsaal oder als Aufgabe bis zur nächsten Vorlesung

@

Erstellen Sie in der Derby-Datenbank ein neues Schema „UNI“

Erstellen Sie die 7 Tabellen des Uni-Schemas, denken Sie dabei an folgende Punkte:

- **Alle Tabellen mit jeweiligen Primärschlüsseln**
- **Alle Fremdschlüsselbeziehungen**
- **Noten können nur im Bereich zwischen 0.7 und 5.0 liegen**
- **Der Rang eines Profs ist entweder C2 oder C3 oder C4**
- **Professoren teilen ihre Räume nicht mit anderen Profs**
- **NOT NULL überall, wo sinnvoll**
- **Löschregeln (ON DELETE)**

→ Freiwillige Selbstkontrolle: UNI-Schema DDL.sql

→ Üben Sie die DDL-Kommandos, die in der Vorlesung besprochen wurden

Füllen Sie im Anschluss die erstellen Tabellen mit Daten mit Hilfe der Datei „UNI-Schema.DML“. Die SQL-INSERT-Operation werden wir in der nächsten Vorlesung besprechen.

Hausaufgaben

bis zur nächsten Vorlesung

@

Portieren Sie das HR-Beispielschema von Oracle nach Derby.

Dies ist ein Beispielschema mit Datenbankobjekten für eine fiktionale Personalabteilung.

Laden Sie das HR-Schema von hier: Oracle Sample Schemas 23c · GitHub (https://github.com/oracle-samples/db-sample-schemas/releases/tag/v23.2)

auf Ihren PC und passen Sie das Skript „hr_create.sql“ so an, dass es erfolgreich gegen die Derby-Datenbank läuft.

Notieren Sie sich bitte, auf welche Probleme Sie dabei gestoßen sind und wie Sie diese gelöst haben.