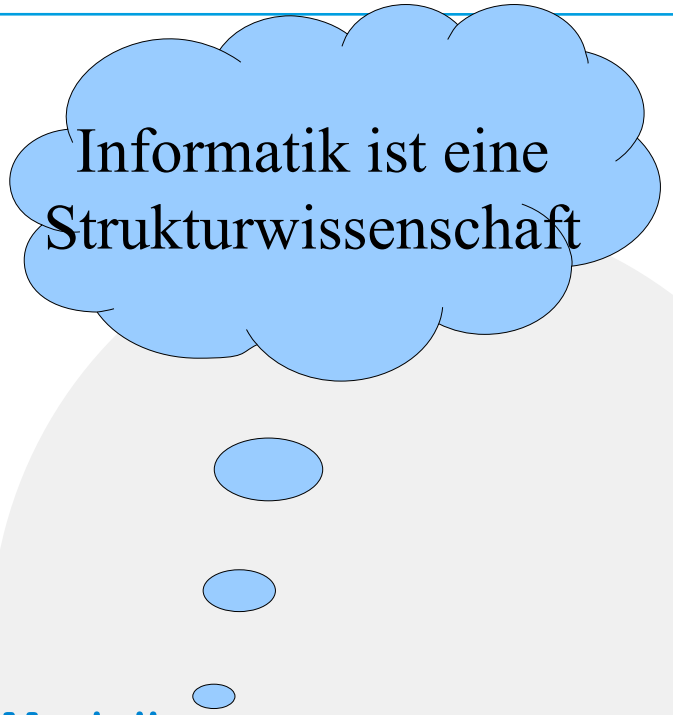


Teil II

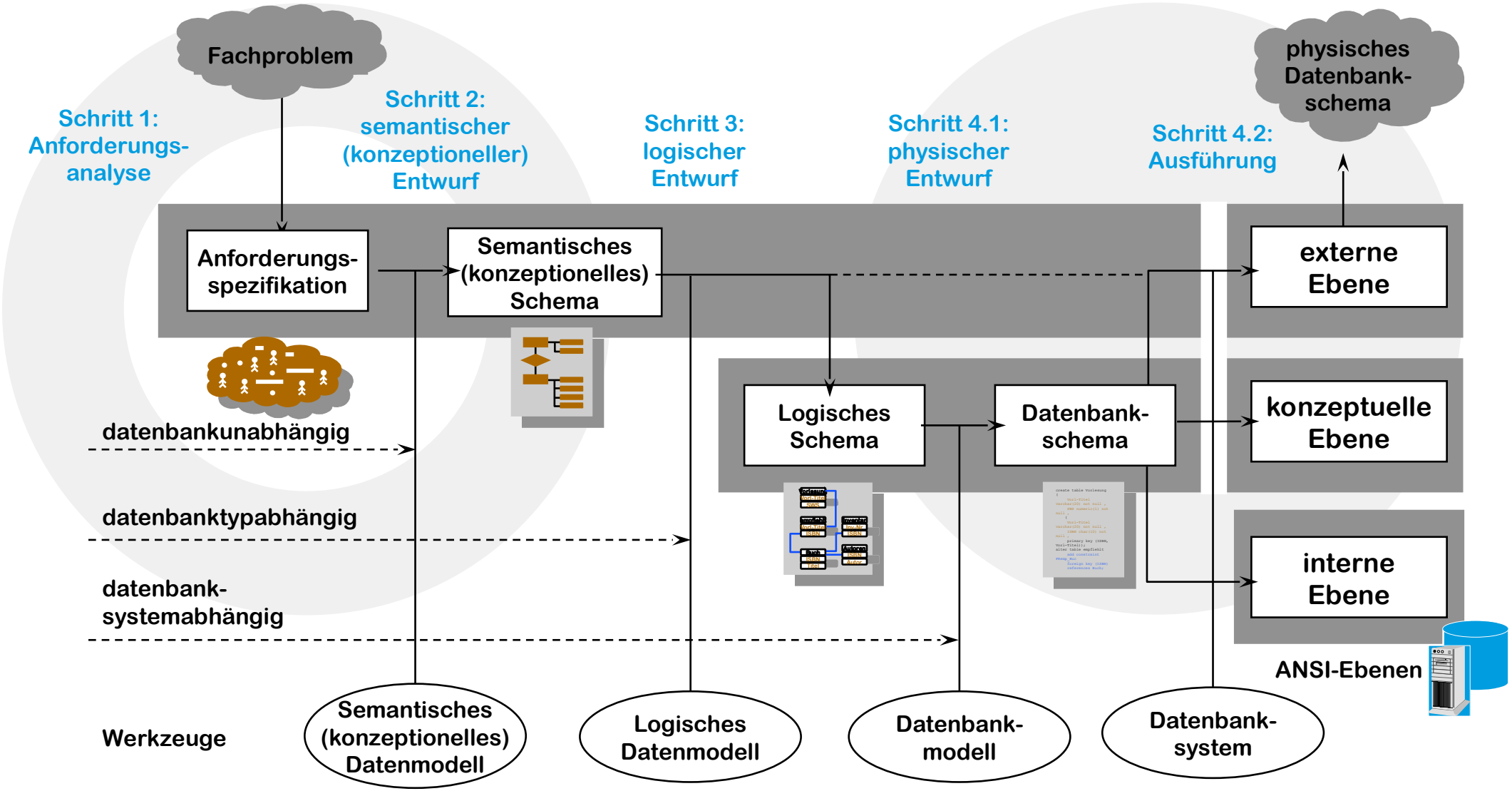
Von Daten und ihren Modellen

Robert Hartmann (SoSe 2024)

**basierend auf Folien von
Prof. Dr. Harm Knolle**

**Fachbereich Informatik
Hochschule Bonn-Rhein-Sieg**

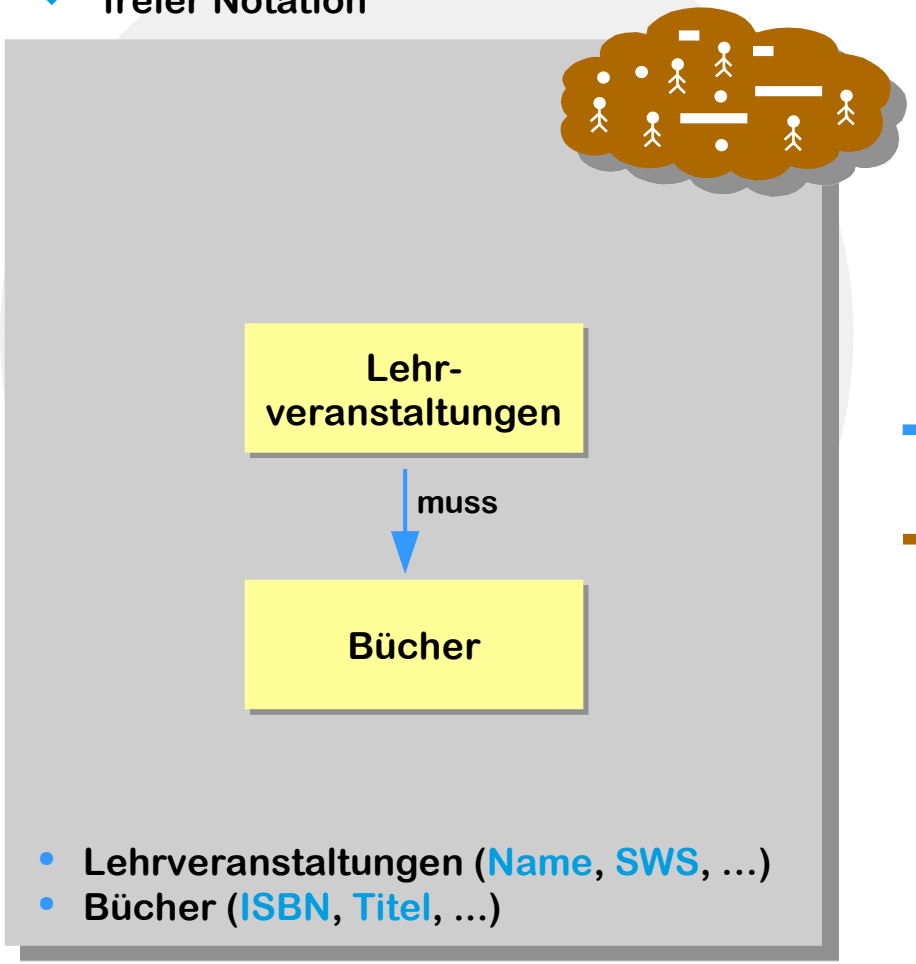
- Vom Fachproblem zur Datenbank -



- Miniwelt vs. Semantisches Datenschema -

Miniwelt (Diskursbereich) mittels

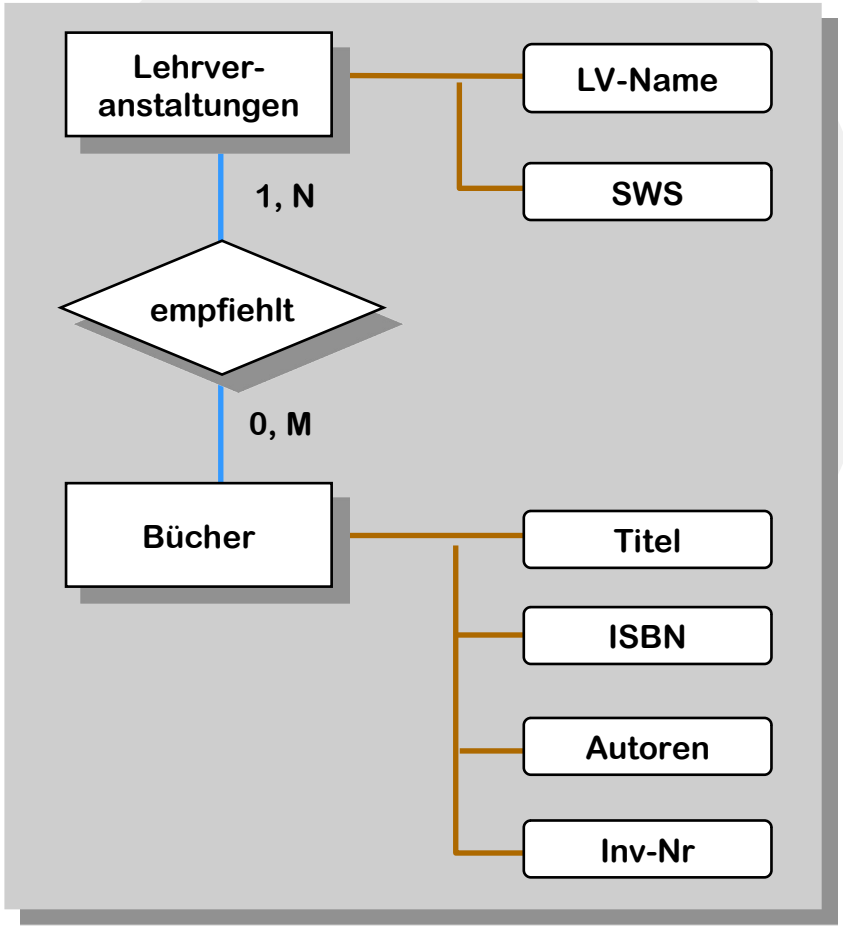
- freier Notation



- Lehrveranstaltungen (Name, SWS, ...)
- Bücher (ISBN, Titel, ...)

Semantisches (konzeptionelles) Schema mittels

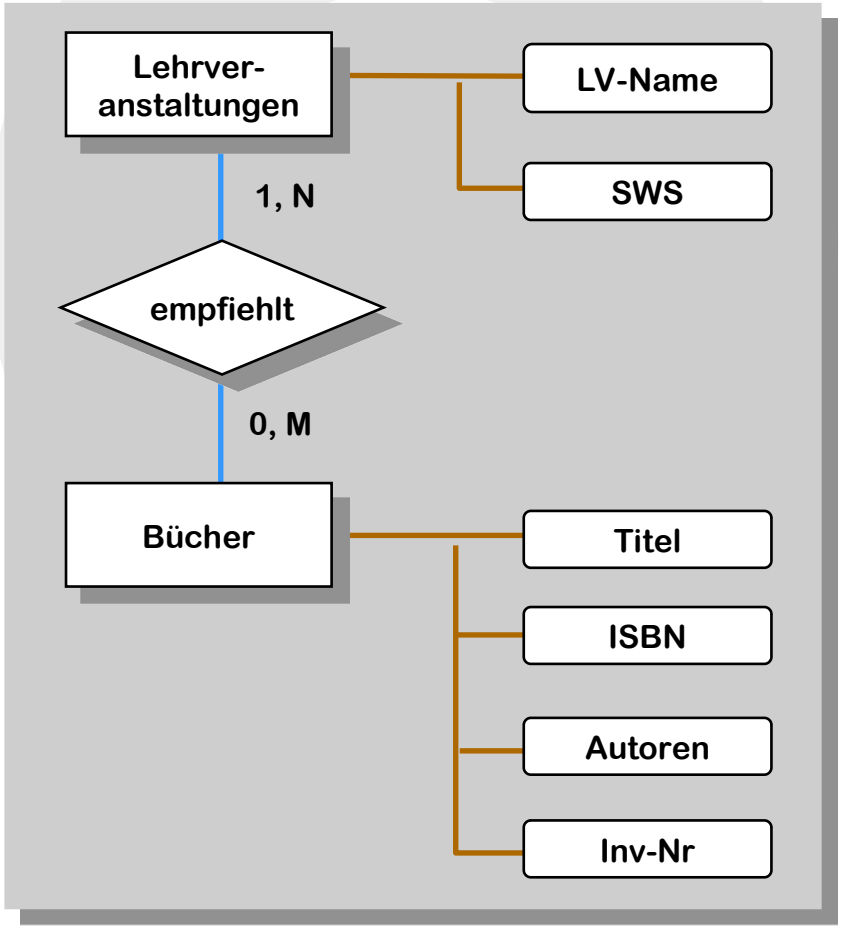
- „Entity-Relationship"-Datenmodell



- Semantisches vs. logisches Datenschema -

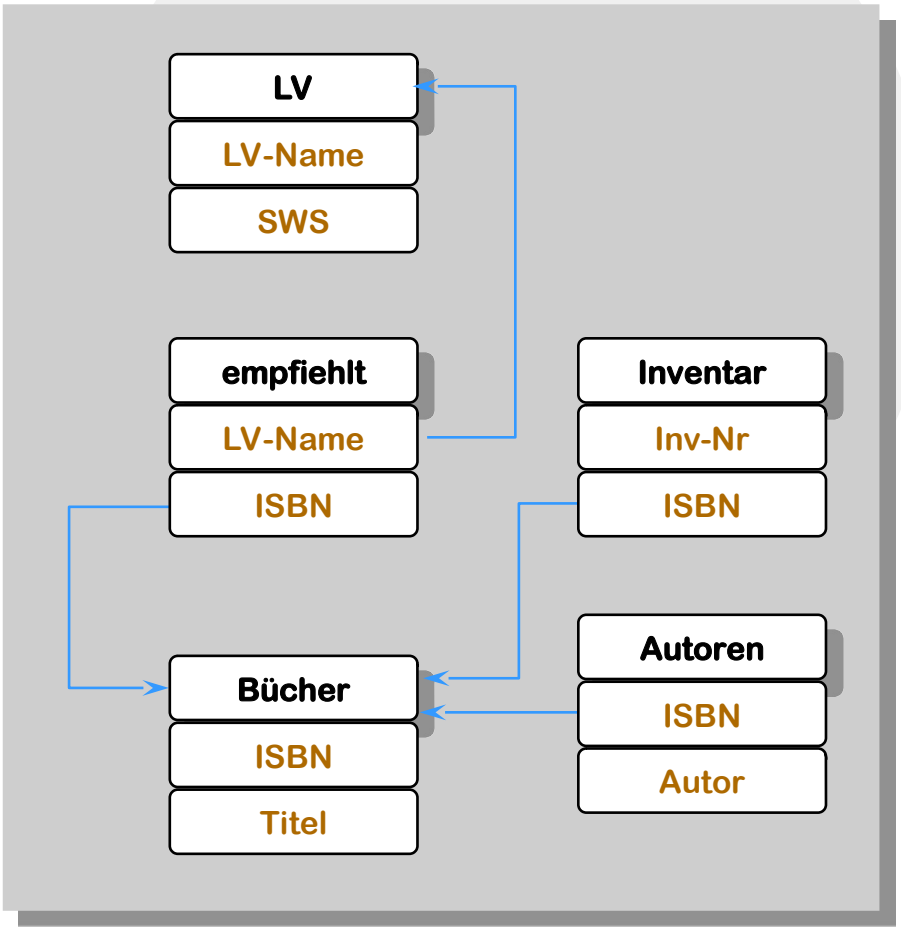
Semantisches (konzeptionelles) Schema mittels

- „Entity-Relationship"-Datenmodell, z.B. nach Abrial



Logisches Schema mittels

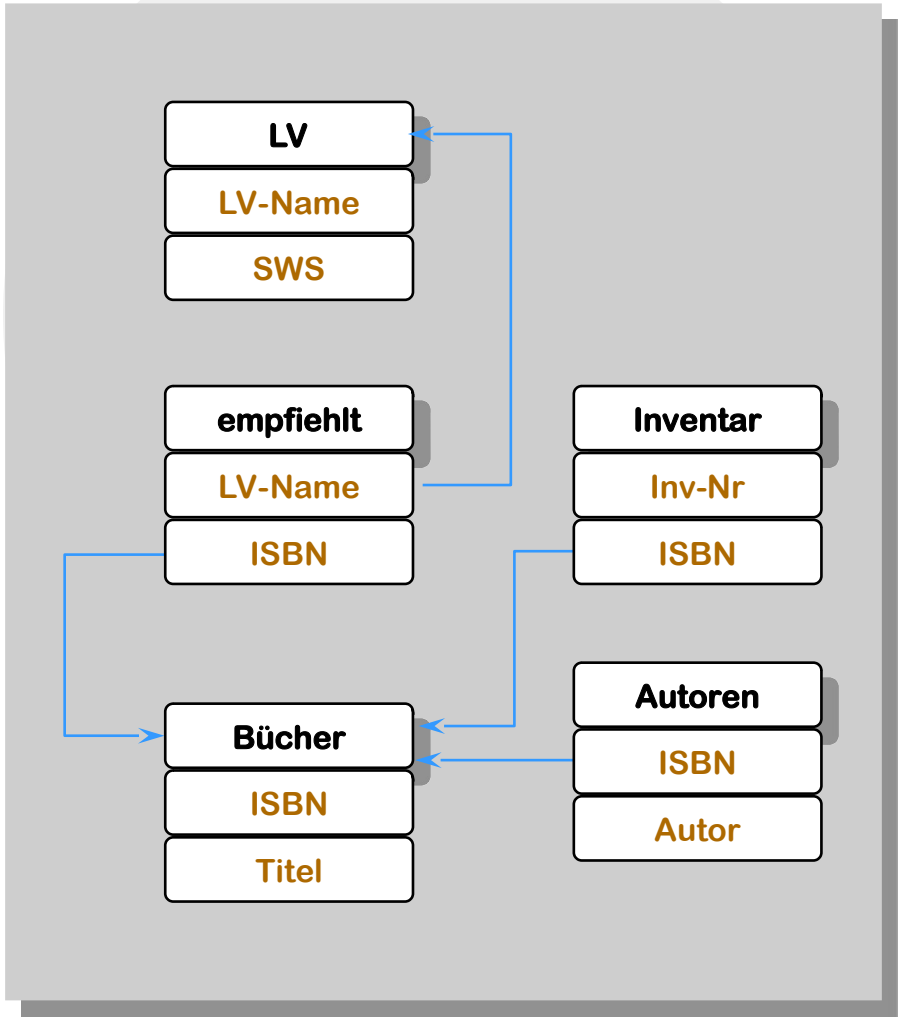
- Relationen-Datenmodell



- Logisches Datenschema vs. Datenbankschema -

Logisches Schema mittels

- Relationen Datenmodell



Datenbankschema mittels

- DDL der SQL

```
create table Lehrveranstaltung(
     LV-Name varchar(20) not null,
     SWS numeric(1) not null,
     primary key (LV-Name));

create table Buecher(
     Titel varchar(20) not null,
     ISBN char(10) not null,
     primary key (ISBN));

create table empfiehlt(
     LV-Name varchar(20) not null,
     ISBN char(10) not null,
     primary key (ISBN, LV-Name));

alter table empfiehlt
     add constraint RK_emp_Buc
     foreign key (ISBN)
     references Buecher;

alter table empfiehlt
     add constraint RK_Vor_emp
     foreign key (LV-Name)
     references Lehrveranstaltung;

. . .
```

- Kapitel 5 - Datenbankmodelle -

Inhalt

Überblick

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL

- Datenbankmodelle -

Ziel

- Erlernen der Syntax einer relationalen Datendefinitionssprache
- Erstellung von Datenbankschemata der konzeptuellen und der externen Ebene
- Transformation eines logischen Schemas in ein Datenbankschema

Hilfsmittel

- Konzepte des Datenbankmodells
- Datendefinitionsanweisungen (DDL) der relationalen Datenbanksprache "Structured Query Language" (SQL)

Inhalt

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL

Literatur

- KeEi15
  - Kapitel 4 „Relationale Anfragesprache“
    - bis einschl. 4.4, 4.16 - 4.18
  - Kapitel 5 „Datenintegrität“
    - bis einschl. 5.6
- Ku15
  - Kapitel 4: „Die Datenbanksprache SQL“
    - 4.7 bis einschl. 4.7.6
- SSH18
  - Kapitel 8: „SQL – der relationale Datenbankstandard“
    - bis einschl. 8.1

- Einführung -

Inhalt

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL

Überblick

- Logisches Modell vs. Datenbankmodell
- Datenbanksprachen zur Datendefinition

- Logisches Modell vs. Datenbankmodell -

Logisches Modell

- Darstellung der Miniwelt (einer Fachabteilung) des Unternehmens in Abhängigkeit vom Typ des Datenbanksystems (hier: Relationales System)
- in der Regel graphische oder formale Beschreibung mit wenigen Konzepten eines logischen Datenmodells (hier: Relationales Datenmodell)
- beinhaltet typmäßige, jedoch keine wertmäßigen Aussagen des Diskursbereichs (der Miniwelt)
- fungiert als Schnittstelle zwischen dem Anwendungsentwickler und dem Datenbankadministrator
- häufig (teil-)automatisierte Ableitung aus einem semantischen Modell möglich
- bildet die Grundlage für den (systemspezifischen) Entwurf der Datenbank

Datenbankmodell

- systemspezifische Darstellung der Miniwelt, die von einem konkreten Datenbanksystem "verstanden" wird
- in der Regel Beschreibung durch eine textuelle Sprache mit einem Befehlssatz (z.B. SQL), die vom System interpretiert und ausgeführt wird
- Ergebnis einer erfolgreichen Ausführung ist die Erstellung physischer Datenbankstrukturen auf einem dauerhaften Speichermedium
- Verwaltung des physischen Modells ist Aufgabe der Datenbankadministration
- häufig (teil-)automatisierte Ableitung aus einem logischen Modell möglich
- bildet die Grundlage für den internen und anwendungsunabhängigen Entwurf der Speicherstrukturen einer Datenbank

- Datenbanksprachen zur Datendefinition -

Realisierung des Datenbankschemas

- Datenbanksystemhersteller orientieren sich zur Definition und Manipulation der Daten in der Datenbank an bestimmten Datenmodellen, halten sich aber nicht immer vollständig daran
- zur Umsetzung des logischen Schemas in ein konkretes Datenbankschema bieten die Hersteller von Datenbanksystemen sogenannte Datenbanksprachen an
- die Sprachelemente dieser Sprachen sind somit die Konzepte des Datenbankmodells
- die Sprachen unterschiedlicher Hersteller unterscheiden sich vielfach auch dann, wenn sie das gleiche logische Datenmodell unterstützen (Dialekte)
- analog zu den logischen Datenmodellen werden auch hier vier grundsätzlich unterschiedliche Sprachen angeboten

Hierarchische Datenbanksprachen

- älteste Datenbanksprache mit stark abnehmender Bedeutung aber noch häufig im Betrieb

Netzwerk Datenbanksprachen

- in den 70'er Jahren stark favorisiert, aber heute nahezu bedeutungslos

Relationale Datenbanksprachen

- derzeit größte praktische Bedeutung

Objekt-orientierte Datenbanksprachen

- wurde als Nachfolger der relationalen Datenbanksprachen gehandelt
- erfolgreicher dagegen sind die sogenannten objekt-relationalen Datenbank(misch)sprachen

NoSQL

- in der Regel keine deskriptive Sprache

- Datenbanksprachen -

Inhalt

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL

Überblick

- Historische Datenbanksprachen
- Heutige Datenbanksprachen
- SQL - Structured Query Language

- Historische Datenbanksprachen -

Hierarchische Systeme

- IMS (Information Management System) von IBM
- Sprache
  - DL/I (Data Language/I) von IBM
- Datendefinition
  - basiert auf hierarchische Anordnung einzelner Datenelemente (Segmente)
  - N:M-Beziehungen werden indirekt über logisch redundante Datenelemente realisiert (Zeiger)

Netzwerk Systeme

- z.B. UDS von Siemens
- Sprache
  - CODASYL DBTG Language
    - COnference On DAta SYstem Languages
    - Data Base Task Group
- Datendefinition
  - basiert auf netzartige Anordnung von Datenelementen (Records), die über 1:N-Beziehungen (Sets) miteinander verbunden sind
  - N:M-Beziehung direkt über mehrere 1:N-Beziehungen darstellbar

- Heutige Datenbanksprachen -

Relationale Systeme

- Standard fast aller Systemanbieter
- Sprache
  - SEQUEL (System R)
  - QUEL (INGRES)
  - SQL (INFORMIX, ORACLE, MS SQL-Server, PostgreSQL, MySQL, ...)
  - QBE (MS-ACCESS)
- Datendefinition
  - basiert auf zweidimensionalen Tabellen
  - Beziehungen über Fremdschlüssel und spezielle “Beziehungs-Tabellen”
  - basiert auf Relationenalgebra und -kalkül
  - mengenorientiert

Objektorientierte Systeme

- Objektrelationale Systeme (z.B Oracle seit Version 8 (1997), ...)
- Objektorientierte Systeme (O^2, POET, ...)
- Sprache
  - ODMG-93 : Erweiterung von C++ (Objekt Data Management Group)
  - ab SQL-99 : objekt-relationale SQL-Erweiterung
  - Transact-SQL, PL/SQL: Definition von anwendungsspezifischen Operationen auf Objekten (Programming Language/SQL)
- Datendefinition
  - direkte Definition semantisch komplex strukturierter (relationaler) Objekte
  - Erweiterungen von C++ oder Java
  - Erweiterungen relationaler Sprachen

- SQL - Structured Query Language -

Sprachliche Fassung des relationalen Datenmodells

- Konzepte zur Implementierung eines relationalen logischen Modells
- Sprachgrunglage relationale Algebra
  - DDL - Data Definition Language
  - DML - Data Manipulation Language
- Konzepte für interne Ebene (Kapitel 8)
  - Datenstrukturdefinition
- Konzepte für konzeptuelle und externe Ebene (dieses Kapitel)
  - Tabellen, anwendungsspezifische Sichten (virtuelle Tabellen)
- Konzepte für Datenbankbetrieb (Vertiefung im Hauptstudium)
  - Transaktion. Mehrbenutzerkontrolle
  - Datenschutz, Datensicherheit
- zahlreiche Dialekte kommerzieller Systeme

Normierung

- SEQUEL: 70er Jahre - System R , IBM
- SQL-86: 1986 - ANSI/ISO-Standard
  - ISO: International Organisation for Standardisation
  - ANSI: American National Standards Institute
- SQL-89: 1989 - Erweiterungen zu SQL-86
- SQL-92 (SQL 2): 1992, „derzeitiger“ Standard (fast) aller kommerzieller SQL-Systeme
- SQL-99: objektrelationale Grundlagen
- SQL-2003: objektrelationaler Standard, XML
- SQL-2006: Integration von XML
- SQL-2008: u.a. Optimierung Trigger
- SQL-2011: u.a. zeitbezogene Daten
- SQL-2016/19: aktueller (theoretischer) Standard
- SQL-2023: Unterstützung JSON als Datentyp

- Datendefinitionssprache (DDL) von SQL -

Inhalt

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL

Überblick

- Entwicklung
- Abbildung relationales Schema auf DB-Schema
- Abbildung semantisches Schema auf DB-Schema

- Entwicklung der Grundlagen -

Theorie nach CODD (1982)

- Attribute
- Wertebereiche
- Relationenschemata
- Primärschlüssel
- Fremdschlüssel

SQL-89

- Relationenschemata
- Attribute
- einfache Wertebereiche
- nur Simulation von Schlüsseln
- keine Definition von Fremdschlüsseln

SQL-92

- Relationenschemata
- Attribute
- Wertebereiche (Domänen)
- Primärschlüssel
- Fremdschlüssel

SQL-99

- Objekt-relationale Erweiterungen

- Abbildung relationales Schema auf DB-Schema -

Entity Realtionship Modell	Relationenmodell	physisches Modell mit SQL
Entity	Tupel	Zeile (Tupel)
Entity-Typ	Relationenschema	Tabellenschema
Attribut	Attribut	Spalte (Attribut)
Domäne (Wertebereich)	Domäne (Wertebereich)	Domäne (Wertebereich)
Datentyp	Datentyp	Datentyp
Beziehungs-Typ	Relationenschema, Fremdschlüssel	Tabellenschema, Fremdschlüssel
Schlüssel	Schlüssel, Primärschlüssel	Schlüssel, Primärschlüssel
Kardinalität	Fremdschlüssel, wertspezifische Bedingungen	Fremdschlüssel, wertspezifische Bedingungen
(kein Konzept)	Fremdschlüssel	Fremdschlüssel
(kein Konzept)	wertspezifische Bedingungen	wertspezifische Bedingungen
strukturiertes Attribute	Relationenschemata, Schlüssel, Primärschlüssel, Fremdschlüssel, wertspezifische Bedingungen	Tabellenschemata, Schlüssel, Primärschlüssel, Fremdschlüssel, wertspezifische Bedingungen
Generalisierung / Spezialisierung	Relationenschemata, Schlüssel, Primärschlüssel, Fremdschlüssel, wertspezifische Bedingungen	Tabellenschema, Schlüssel, Primärschlüssel, Fremdschlüssel, wertspezifische Bedingungen

- Abbildung semantisches Schema auf DB-Schema -

Problem

- Normalisierung des semantischen Schemas in viele kleine Relationen
- Anwendung muss den semantischen Zusammenhang der Daten wieder herstellen

Lösung: Definition von Sichten (Views)

- Sicht = scheinbare (virtuelle) Relation
- Realisierung nicht-normalisierter Relationen des semantischen Schemas als Sichten
- dadurch Realisierung der externen Ebene
- Datenschutz: z.B Ausblenden von Attributen
- Fixierung häufig formulierter und komplexer Anfragen
- wichtiges Mittel zur Erhöhung der Datenunabhängigkeit von Anwendungen
- allerdings nur bedingt Änderungsoperationen möglich

Beispiel

Gebäude

Gebäude	Straße
B	Am Schwimmbad
C	Blechhammer
D	Blechhammer

Veranstaltungsort

LV.-Nr	Gebäude	Raum
1238	B	431
1153	D	123
1543	B	23
1352	C	231
1421	C	250

zwei real existierende Relationen

Sicht Veranstaltungsort

LV.-Nr	Gebäude	Straße	Raum
1238	B	Am Schwimmbad	431
1153	D	Blechhammer	123
1543	B	Am Schwimmbad	23
1352	C	Blechhammer	231
1421	C	Blechhammer	250

nicht-normalisierte Relation als Sicht

- Sprachkonzepte der SQL-DDL -

Inhalt

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL

Überblick

- Konzeptuelle Ebene
- Externe Ebene (virtuelle Tabellen)

- Konzeptuelle Ebene -

Inhalt

- Einführung
- Datenbanksprachen
- Datendefinitionssprache (DDL) von SQL
- Sprachkonzepte der SQL-DDL
  - Konzeptuelle Ebene
  - Externe Ebene (virtuelle Tabellen)

Überblick

- Konzepte
- Datentypen
- Wertebereiche (SQL-92)
- Tabellenschema
- Integrität (SQL-92)
  Integritätsbedingungen
- Sonstige Bedingungen

- Konzepte -

Typen

- CHAR, INT, REAL, DATE, NUMERIC, . . .

Wertebereiche

- DOMAIN

Tabellenschema

- TABLE

Primärschlüssel, Schlüssel

- PRIMARY KEY
- UNIQUE KEY

Fremdschlüssel

- FOREIGN KEY

Tupelintegrität

- **Schlüsselbedingung**
  - PRIMARY KEY CONSTRAINT
  - UNIQUE KEY CONSTRAINT

Referentielle Integrität

- **Kardinalität und Fremdschlüsselbedingung**
  - FOREIGN KEY CONSTRAINT
  - CHECK CONSTRAINT

Werte- bzw. Anwendungsspezifische Integrität

- **Wertebedingung**
  - CHECK CONSTRAINT

- Datentypen -

Auswahl an Typen

```
Wertebeschreibung :=
  {<Datentyp>|<Domäne>}

Datentyp :=                                {
  CHAR[(<n>)]                              |
  VARCHAR[(<n>)]                           |
  {INT | INTEGER}                          |
  {FLOAT | REAL}                           |
  NUMERIC[<precision>[, <scale>]]          |
  DATE                                     }

precision :=
  <positive ganze Zahl>
scale :=
  <ganze Zahl>
```

```
CHAR          := 'a', '5', ' '
              :≠ 'fd-5', 5

CHAR(4)       :=  'a', 'fd-5', ' '
              :≠  'abcdef', 12345

VARCHAR(4)    := 'ab d', '7   '
              :≠  '7     ', 123.4

INT*          :=   7, -5437, 0, 32767
              :≠  '7', 32768

REAL          :=  -45, 45.25467, 0
              :≠  '1234.7'

NUMERIC       :=  45, 0, -5437
              :≠  45.25, '45'

NUMERIC(2)    :=  45, 0, -45
              :≠  0.45, 45.25, 452, '45'

NUMERIC(4,2)  :=  45, 45.25
              :≠  -4525.46, 452546

DATE          :=  '22-jun-1961',
                  '22.06.1961'
              :≠  22061961,
                  22.06.1961
```

* 16 Bit, signed

- Wertebereiche (SQL-92) -

Definition von Wertebereichen

```
CREATE DOMAIN <Name>
  AS <Wertebeschreibung>
```

```
CREATE DOMAIN TypeName
  AS VARCHAR(30)

CREATE DOMAIN TypeAdresse
  AS VARCHAR(50)

CREATE DOMAIN TypeAlter
  AS NUMERIC(3)

CREATE DOMAIN TypePreis
  AS NUMERIC(6,2)

CREATE DOMAIN TypeVorname
  AS TypeName
```

Löschen von Wertebereichen

```
DROP DOMAIN <Name>
```

```
DROP DOMAIN TypeName

DROP DOMAIN TypeAdresse

DROP DOMAIN TypeAlter

DROP DOMAIN TypePreis

DROP DOMAIN TypeVorname
```

- Tabellenschema -

```
CREATE TABLE <Tabelle>
  <Spaltenbeschreibung>

Spaltenbschreibung :=
  ( <Spalte> <Wertebereich>
  [, Spaltenbeschreibung] )
```

```
CREATE TABLE    TabStudent
  ( Matr_Nr     NUMERIC(10)
  , Name        TypeName
  );

CREATE TABLE    TabLehrveranstaltung
  ( LV_Nr       NUMERIC(4)
  , LV_Name     TypeName
  , Gebaeude    CHAR(1)
  , Raum        CHAR(5)
  );

CREATE TABLE    TabBelegungsplan
  ( Matr_Nr     NUMERIC(10)
  , LV_Nr       NUMERIC(4)
  );
```

Ändern von Tabellendefinitionen

```
ALTER TABLE <Tabelle>
  ADD <Spaltenbschreibung>

Spaltenbschreibung :=
  <Spalte> <Wertebereich>
  [ADD Spaltenbeschreibung]
```

```
ALTER TABLE     TabStudent
  ADD Vorname   TypeVorname
  ADD Geb_Datum DATE ;
```

Löschen von Tabellendefinitionen

```
DROP TABLE      <Tabelle>
```

```
DROP TABLE      TabStudent

DROP TABLE      TabLehrveranstaltung
```

Relationen der Folie BCNF Normalform (II) verwendet, vgl. Foliensatz „Kapitel 4 Teil 1“.

- Integritätsbedingungen (SQL-92) -

Constraints (Einschränkungen)

```
ALTER TABLE <Tabellenname>
  ADD CONSTRAINT <Name>
  <Integrität>

Integrität :=
  {<Tupelintegrität>                |
   <referentielle Inegrität>        |
   <anwend.spez. Integrität>        }

Tupelintegrität :=
  {UNIQUE | PRIMARY KEY} (<Spalten>)

referentielle Integrität :=
  FOREIGN KEY (<Spalten>)
  REFERENCES <Tabelle> [(<Spalten>)]

Spalten :=
  <Spalte> [, <Spalten>]
```

Primär- und Fremdschlüssel

```
Alter TABLE       TabLehrveranstaltung
  ADD CONSTRAINT  PKLehrveranstaltung
  PRIMARY KEY     (LV_Nr) ;

ALTER TABLE       TabLehrveranstaltung
  ADD CONSTRAINT  UKLehrveranstaltung
  UNIQUE KEY      (LV_Name) ;

ALTER TABLE       TabBelegungsplan
  ADD CONSTRAINT  PKBelegungsplan
  PRIMARY KEY     (Matr_Nr, LV_NR) ;

ALTER TABLE       TabBelegungsplan
  ADD CONSTRAINT  FKStudent
  FOREIGN KEY     (Matr_Nr)
  REFERENCES      Student ;

ALTER TABLE       TabBelegungsplan
  ADD CONSTRAINT  FKLehrveranstaltung
  FOREIGN KEY     (LV_NR)
  REFERENCES      TabLehrveranstaltung ;
```

Relationen der Folie BCNF Normalform (II) verwendet, vgl. Foliensatz „Kapitel 4 Teil 1“.

- Sonstige Bedingungen (SQL-92) -

```
anwend.spez. Integrität :=
  CHECK <Bedingungen>

Bedingungen :=
  {(<Bedingung>)                          |
   NOT <Bedingungen>                      |
   <Bedingung> AND <Bedingungen>          |
   <Bedingung> OR  <Bedingungen>          }

Bedingung :=
  { <Spaltenname> IS [NOT] NULL           |
    EXISTS <Datenbankanfrage>             }

Datenbankabfrage :=
  SELECT *
  FROM     <Tabelle>, <Tabelle>
  WHERE    <Prädikat>

Prädikat :=
  [<Tabelle>.]<Spaltenname>
  θ [<Tabelle>.]<Spaltenname>

θ := { < | > | = | <= | >= }
```

Kardinalität von Attributen

```
ALTER TABLE        TabStudent
  ADD CONSTRAINT   NNStudentName
  CHECK ( Name IS NOT NULL );

ALTER TABLE        TabLehrveranstaltung
  ADD CONSTRAINT   NNLehrveranstaltungName
  CHECK ( LV_Name IS NOT NULL );
```

Simulation von Muss-Beziehungen

```
ALTER TABLE        TabLehrveranstaltung
  ADD CONSTRAINT   EXOrt
  CHECK ( Gebaeude IS NOT NULL
          AND
          Raum IS NOT NULL );

ALTER TABLE        TabLehrveranstaltung
  ADD CONSTRAINT   EXBelegungsplan
  CHECK (  EXISTS SELECT  *
           FROM   TabLehrveranstaltung l,
                  TabBelegungsplan b
           WHERE  l.LV_NR = b.LV_NR );
```

Relationen der Folie BCNF Normalform (II) verwendet, vgl. Foliensatz „Kapitel 4 Teil 1“.

- DOMAIN mit Bedingungen -

```
CREATE DOMAIN <DomainTypeName>
         AS <Wertebeschreibung>

ALTER DOMAIN <DomainTypeName>
  ADD <domain_constraint>

domain_constraint :=
      CONSTRAINT <Name>
          NOT NULL |
          NULL     |
          <anwend.spez. Integrität>

ALTER DOMAIN <Name>
    VALIDATE CONSTRAINT constraint_name


anwend.spez. Integrität :=
  CHECK <Bedingungen>

Syntax für <Bedingungen> siehe Folie 26
```

Beispiel Erzeugung eines eigenen Datentyps für Wochentage.

```
CREATE DOMAIN MeineWochentage
         AS CHAR(2)

ALTER DOMAIN MeineWochentage
  ADD CONSTRAINT dcWochentage
  CHECK (VALUE IN ('Mo',
                   'Di',
                   'Mi',
                   'Do',
                   'Fr',
                   'Sa',
                   'So'))
```

Keine automatische Identifizierung von verletzenden Zeilen durch Standard-SQL.

Manuelle Ursachen-Abfrage notwendig bei Fehlschlag der Änderung durch Existenz von verletzenden Daten.

- Externe Ebene (virtuelle Tabellen) -

Definieren von externen Sichten

```
CREATE VIEW  <Sicht>
  AS         <Anfrage>

Anfrage :=
  SELECT     <Tupel>
  FROM       <Tabellen>            [
  WHERE      <Prädikat>            ]

Anfragesyntax siehe Kapitel 6
```

```
CREATE VIEW  VStudent
  AS         SELECT  Vorname, Name
             FROM    TabStudent

CREATE VIEW  VBelegungsplan
  AS         SELECT  S.Name, L.LV_Name
             FROM    TabStudent S,
                     TabLehrveranstaltung L,
                     TabBelegungsplan B
             WHERE   S.Matr_Nr=B.Matr_Nr
             AND     L.LV_Nr=B.LV_Nr
```

Ändern von externen Sichten

```
CREATE OR REPLACE VIEW
             <Sicht>
  AS         <Anfrage>
```

```
CREATE OR REPLACE VIEW
             VMitarbeiter
  AS         SELECT Vorname, Name
             FROM   TabMitarbeiter
```

Löschen von Sichtdefinitionen

```
DROP VIEW  <Sicht>
```

```
DROP VIEW  VMitarbeiter

DROP VIEW  VBelegungsplan
```