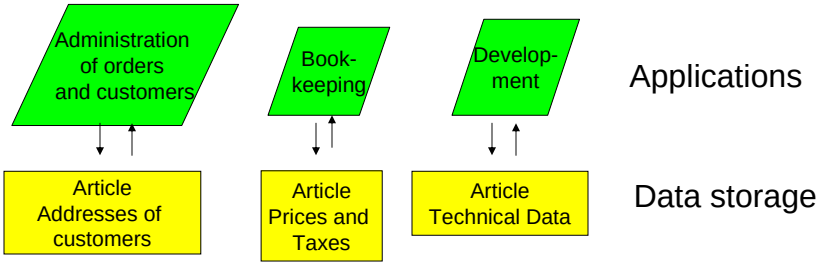
Foundations of Databases – 1: Introduction

Meike Klettke
Universität Greifswald, Fachbereich Mathematik und Informatik
Universität Rostock, Fakultät für Informatik und Elektrotechnik
meike.klettke@uni-greifswald.de oder
meike@informatik.uni-rostock.de

Motivation

- Databases can
  - store data,
  - administer data,
  - realise queries (efficiently),
  - realise updates,
  - protect data
- Fields of applications
- Advantages and disadvantages of database systems
- Data model:
  - How we can store data?
  - Operations on the data model /Query language
  - Design of databases
  - Normal forms of relational databases

Data storage without databases



Software stores the data in individual (file) formats

- Changes of an *article*: how to realise it?

Other Problems:

- Redundant storage of information
- Multi user processes cannot be realised
- Data protection and data security are difficult to realise

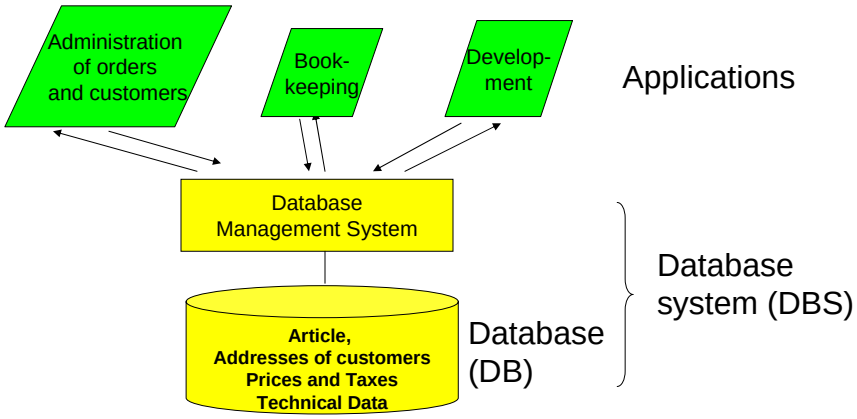
Data storage in databases

All applications work with the same data

- For example: Addresses and Articles are stored in a databases, several application can use it
- Consistency of data can be checked

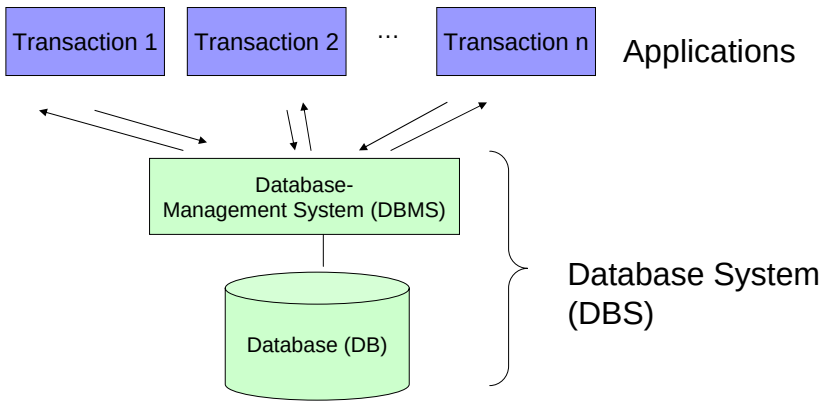
- Database system can efficiently administer large amounts of data (optimisation of query execution, indexing of data)
- Several user can work parallel on the database (Realising Transactions)
- Data Security
- Data Protection
- Database Recovery

Data storage in databases



- Changes of *article* data: how to realise it?

Generalised:



Basic architecture of a database system

Databases are not necessary ..

- if only a few data are available and
- data are only used over a short period of time and
- only one person or only some persons are using the data

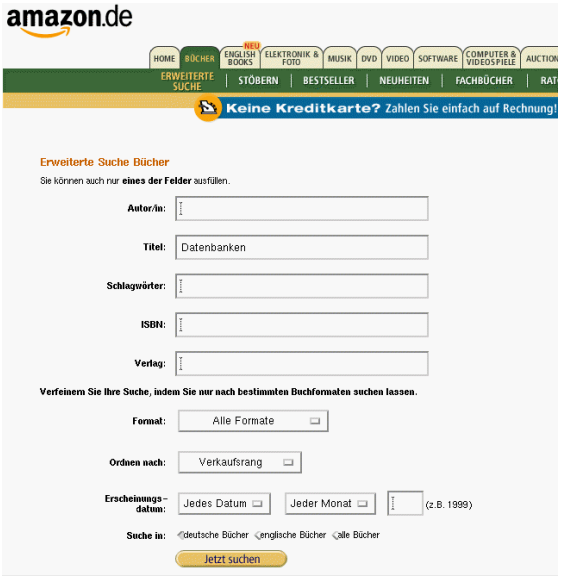
- In all other cases databases are necessary!

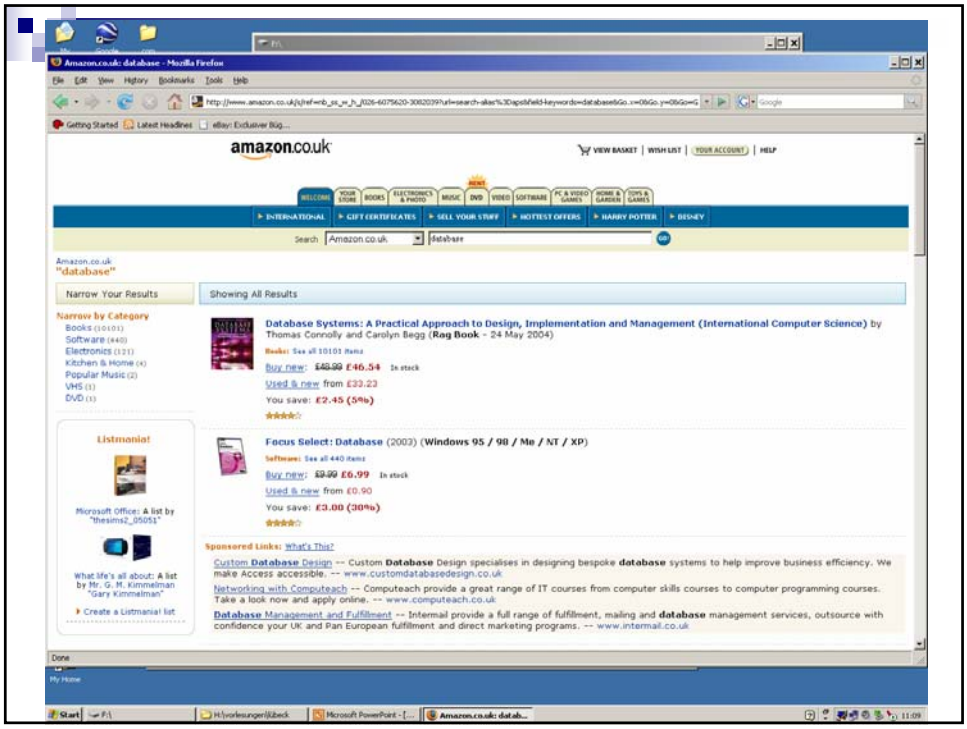
Databases are necessary ..

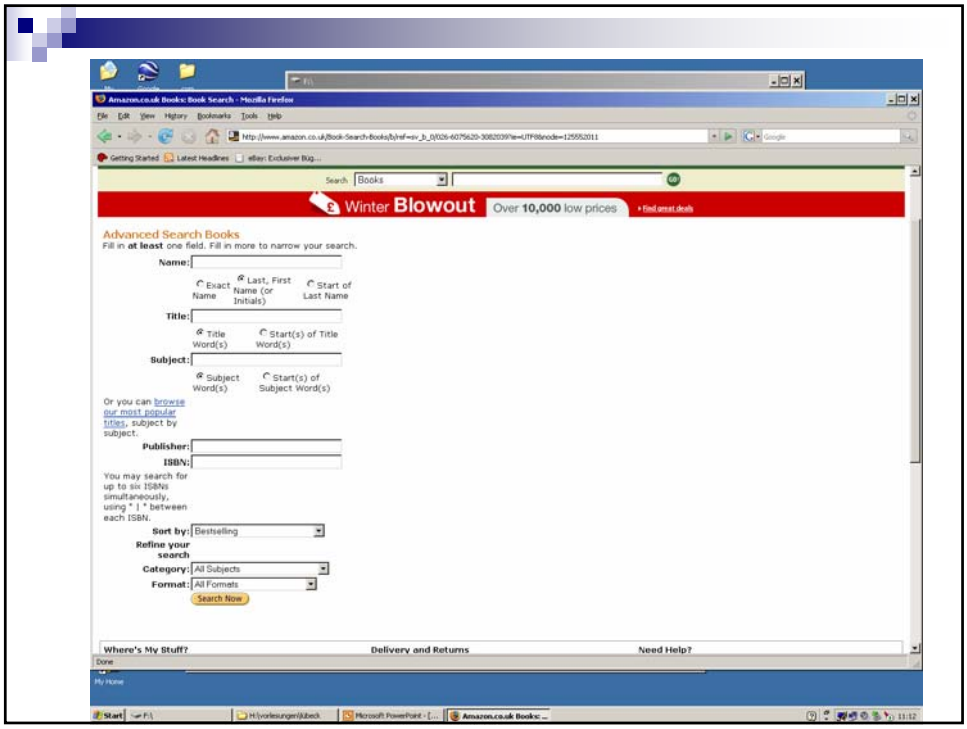
- Large amounts of data are available
- Efficient query realisation is necessary
- Many users shall read/ change the data
- Different users shall have different rights (only read/ read and update, delete)
- Different applications use the same data
- Data are used over longer periods of time
- Data consistency is important (correct data)
- Data security (privacy: not all user can read or change all information)
- Data protection (to prevent loss of data)

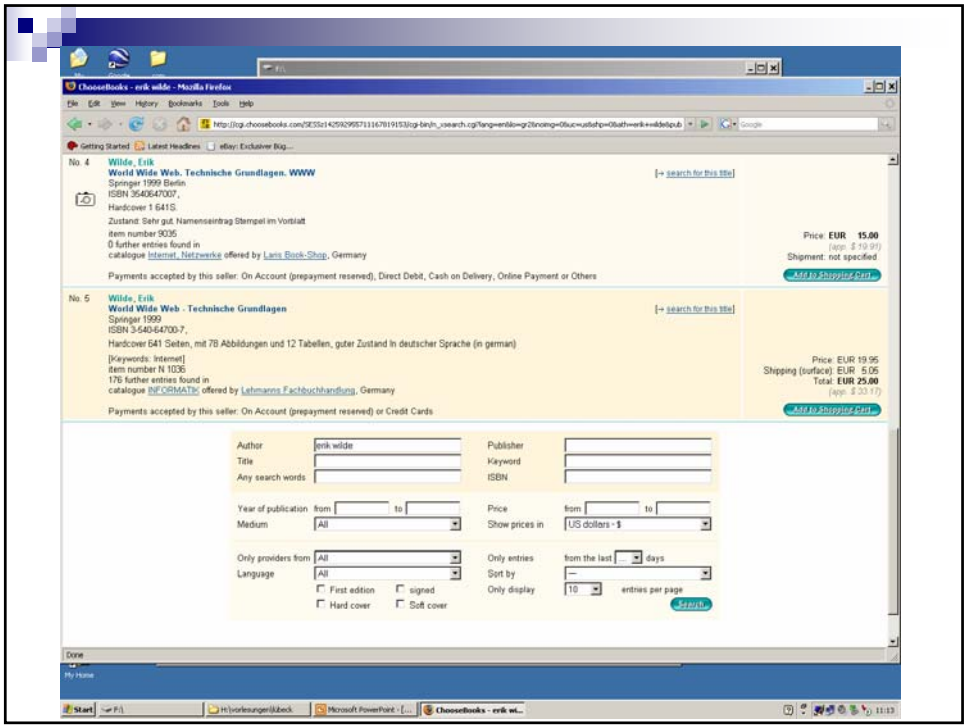
Did anybody ever work with databases

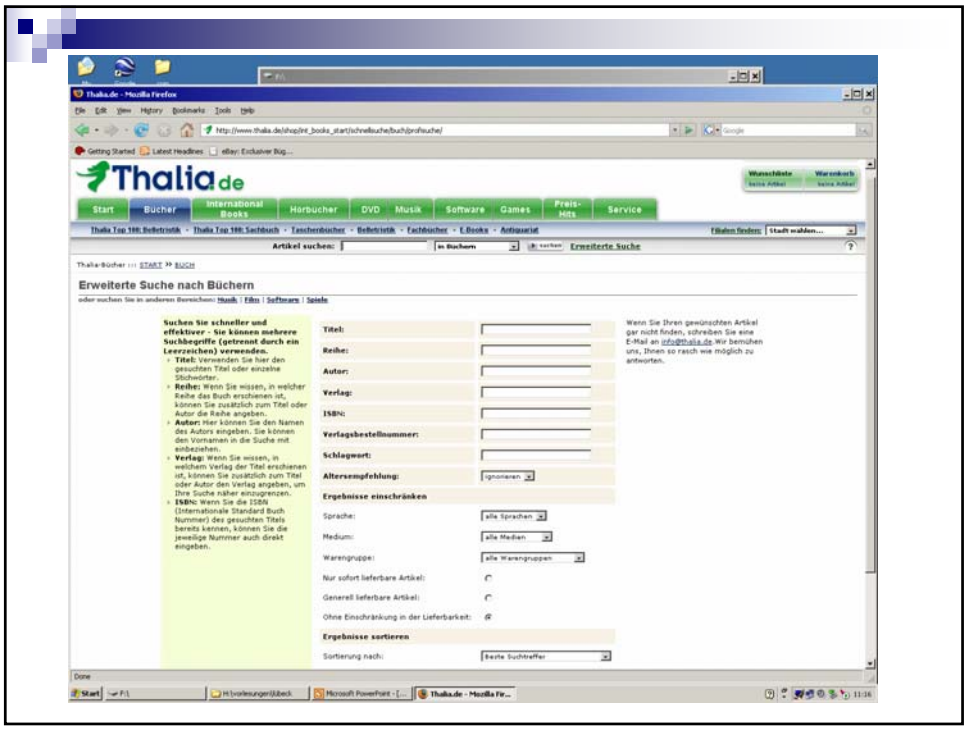
?




Amazon.co.uk: database - Mozilla Firefox
amazon.co.uk
Search Amazon.co.uk database
Amazon.co.uk "database"
Narrow Your Results
Showing All Results
Narrow by Category
Books (10101)
Software (440)
Electronics (121)
Kitchen & Home (4)
Popular Music (2)
VHS (1)
DVD (1)
Listmania!
Database Systems: A Practical Approach to Design, Implementation and Management (International Computer Science) by Thomas Connolly and Carolyn Begg (Rag Book - 24 May 2004)
Buy new: £48.99 £46.54 In stock
Used & new from £33.23
You save: £2.45 (5%)
Focus Select: Database (2003) (Windows 95 / 98 / Me / NT / XP)
Buy new: £9.99 £6.99 In stock
Used & new from £0.90
You save: £3.00 (30%)
Sponsored Links: What's This?
Custom Database Design -- Custom Database Design specialises in designing bespoke database systems to help improve business efficiency. We make Access accessible. -- www.customdatabasedesign.co.uk
Networking with Computeach -- Computeach provide a great range of IT courses from computer skills courses to computer programming courses. Take a look now and apply online. -- www.computeach.co.uk
Database Management and Fulfillment -- Intermail provide a full range of fulfillment, mailing and database management services, outsource with confidence your UK and Pan European fulfillment and direct marketing programs. -- www.intermail.co.uk


Amazon.co.uk Books: Book Search - Mozilla Firefox
Search Books
Winter Blowout Over 10,000 low prices
Advanced Search Books
Fill in at least one field. Fill in more to narrow your search.
Name:
Exact Name
Last, First Name (or Initials)
Start of Last Name
Title:
Title Word(s)
Start(s) of Title Word(s)
Subject:
Subject Word(s)
Start(s) of Subject Word(s)
Or you can browse our most popular titles, subject by subject.
Publisher:
ISBN:
You may search for up to six ISBNs simultaneously, using " | " between each ISBN.
Sort by: Bestselling
Refine your search
Category: All Subjects
Format: All Formats
Search Now
Where's My Stuff?
Delivery and Returns
Need Help?


ChooseBooks - erik wilde - Mozilla Firefox
No. 4
Wilde, Erik
World Wide Web. Technische Grundlagen. WWW
Springer 1999 Berlin
ISBN 3540647007,
Hardcover 1 641 S.
Zustand: Sehr gut. Namenseintrag Stempel im Vorblatt
item number 9035
0 further entries found in
catalogue Internet, Netzwerke offered by Lanis Book-Shop, Germany
Payments accepted by this seller: On Account (prepayment reserved), Direct Debit, Cash on Delivery, Online Payment or Others
Price: EUR 15.00
Shipment: not specified
No. 5
Wilde, Erik
World Wide Web - Technische Grundlagen
Springer 1999
ISBN 3-540-64700-7,
Hardcover 641 Seiten, mit 78 Abbildungen und 12 Tabellen, guter Zustand In deutscher Sprache (in german)
[Keywords: Internet]
item number N 1036
176 further entries found in
catalogue INFORMATIK offered by Lehmanns Fachbuchhandlung, Germany
Payments accepted by this seller: On Account (prepayment reserved) or Credit Cards
Price: EUR 19.95
Shipping (surface): EUR 5.05
Total: EUR 25.00
Author
erik wilde
Title
Any search words
Publisher
Keyword
ISBN
Year of publication
Medium
Price
Show prices in
US dollars - $
Only providers from
Language
Only entries
Sort by
Only display
entries per page
First edition
signed
Hard cover
Soft cover

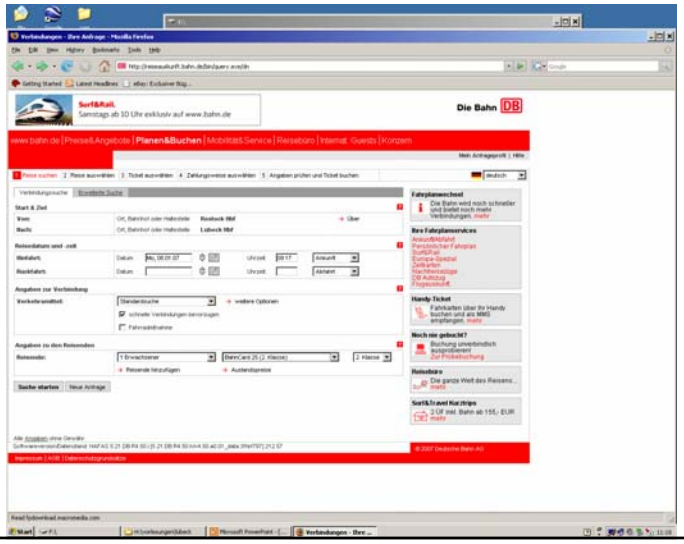

Thalia.de - Mozilla Firefox
Thalia de
Start
Bücher
International Books
Hörbücher
DVD
Musik
Software
Games
Preis-Hits
Service
Erweiterte Suche nach Büchern
Titel:
Reihe:
Autor:
Verlag:
ISBN:
Verlagsbestellnummer:
Schlagwort:
Altersempfehlung:
Ergebnisse einschränken
Sprache:
Medium:
Warengruppe:
Nur sofort lieferbare Artikel:
Generell lieferbare Artikel:
Ohne Einschränkung in der Lieferbarkeit:
Ergebnisse sortieren
Sortierung nach:

In all cases

- Underlying (relational) database,
  - similar organised
  - similar kinds of queries can be realised

```
select *
from book
where Title like ´%database%´
order by price;
```

Another example



Relational Model

- A database is a set of relations

lending

signature	name
2937	Meyer
2393-2	Schulz
2928	Schmidt
9736	Lehmann

books

signature	title	ISBN	author
2937	Datenbanken – 1	2-198-43948-3	Heuer/Saake
2928	Datenbanken - 2	9-845-49375-1	Saake/Heuer
9736	Database Systems	9-346-24657-2	Date
2393-2	ER Modellierung	4-637-44929-1	Thalheim
6430	Informationssysteme	2-394-49483-3	Biskup
4957	Datenbank-Handbuch	4-298-49387-3	Lockemann

- Names of relations and attribute names belong to the **schema** of the relation
- A **relation** is also called **table**
- A **row** of the table or a **tuple**
- An **attribute** of the relation is also called **column**

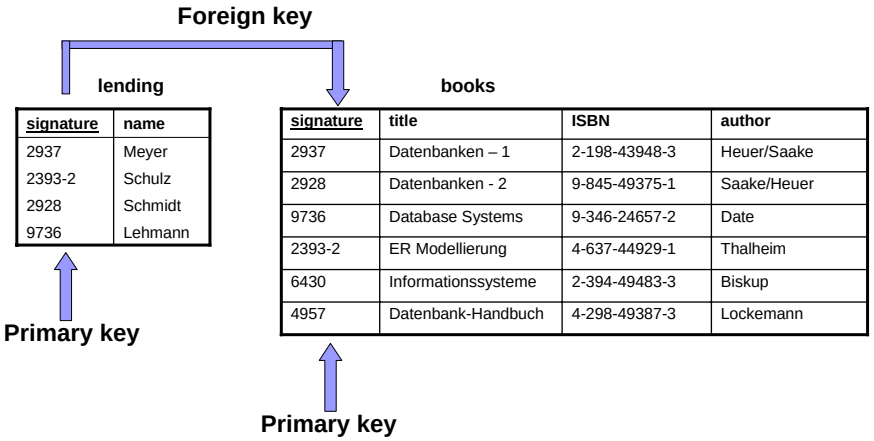
...

- A database is a **set** of relations (tables)
- A relation is a **set** of attributes (columns)
- A relation contains a **set** of tuples (rows)

Integrity Constraints

- Local integrity constraints
  - **signature** (in the sample relation **books**) is *primary key* for books
    - values are unique,
    - there are no different tuples in the table book that have the same values for signature
    - Keys are used for identifying rows
- Global integrity constraints
  - signature in books is *foreign key* and references books
  - That means:
    - each value for signature in the relation **lending** has to occur in the table **books**, attribute **signature**
    - in the relation **books** the attribute **signature** is a *primary key*

Relational Model with Keys and Foreign Keys



Query Operations / 1

- Relational data model also integrates the facility for realising query operations
- **Selection** of rows
  - All books that the author ‚Heuer' wrote

signature	title	ISBN	author
2937	Datenbanken - 1	2-198-43948-3	Heuer/Saake
2928	Datenbanken - 2	9-845-49375-1	Saake/Heuer

  - All books that the library user with the name ‚Schulz' borrowed

signature	name
2392-2	Schulz

Query Operations / 2

- Selection of columns (attributes) (Projection)
- title and authors of books

title	author
Datenbanken - 1	Heuer/Saake
Datenbanken - 2	Saake/Heuer
Database Systems	Date
ER Modellierung	Thalheim
Informationssysteme	Biskup
Datenbank-Handbuch	Lockemann

Query Operations / 3

- Join (prerequisite for queries over several tables):
- Tables are connected over attributes with identical values

signature	name
2937	Meyer
2393-2	Schulz
2928	Schmidt
9736	Lehmann

signature	title	ISBN	author
2937	Datenbanken – 1	2-198-43948-3	Heuer/Saake
2928	Datenbanken - 2	9-845-49375-1	Saake/Heuer
9736	Database Systems	9-346-24657-2	Date
2393-2	ER Modellierung	4-637-44929-1	Thalheim
6430	Informationssysteme	2-394-49483-3	Biskup
4957	Datenbank-Handbuch	4-298-49387-3	Lockemann

signature	name	title	ISBN	author
2937	Meyer	Datenbanken – 1	2-198-43948-3	Heuer/Saake
2393-2	Schmidt	Datenbanken – 2	9-845-49375-1	Saake/Heuer
2928	Lehmann	Database Systems	9-346-24657-2	Date
9736	Schulz	ER Modellierung	4-637-44929-1	Thalheim

Query Operations / 4

- All operations that are shown till now can be combined
- For example:
  - We are looking for all the book titles of all books that the user with the name „Schulz“ borrowed

name	title
Schulz	ER Modellierung

Which operations are necessary for deriving this result?

There exist further operations:

- union
- intersect
- renaming

Views

Different users are interested in different information

Different „Views“ can be offered

For example:

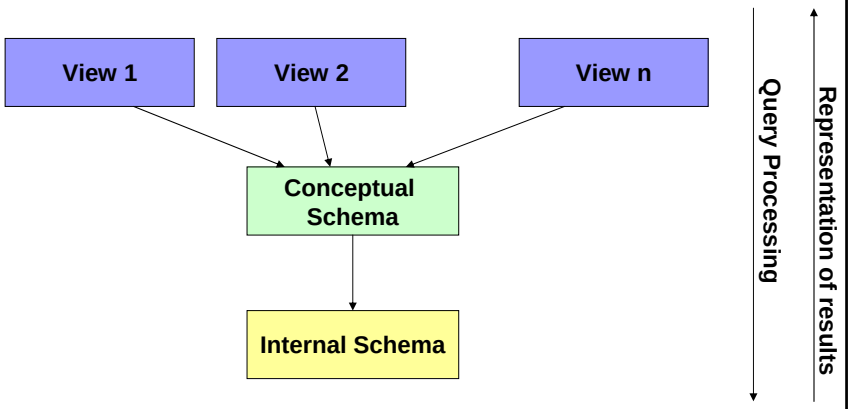
For a library user:

signature	title	author
2937	Datenbanken - 1	Heuer/Saake
2928	Datenbanken - 2	Saake/Heuer
9736	Database Systems	Date
2393-2	ER Modellierung	Thalheim

For the librarian:

signature	ISBN
2937	2-198-43948-3
2928	9-845-49375-1
9736	9-346-24657-2
2393-2	4-637-44929-1

Schema Architecture



Database Design Process

- Stepwise process
- Data independency:
  - Design with a conceptual model is independent from the logical design
  - Design on the logical level (relational model) ins independent from the physical storage

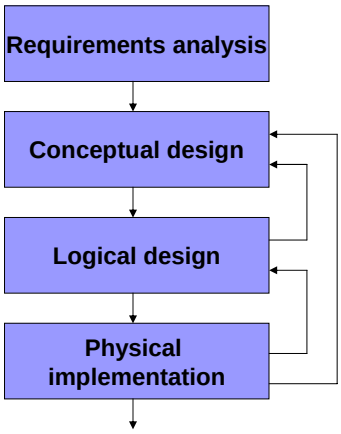


Figure similar in Mannila/Räihä

Standardisation

- Standards for database queries:
- SQL: Structured Query Language,
  - Available Standard SQL2003
  - Actual work on SQL 200x,
  - Systems have implemented (SQL-99, SQL-92)
- DDL: Data Definition Language

Conclusion of the First Part

- Advantages and disadvantages of database systems
- Fields of applications
- Foundation of the relational model
  - Database, relation, attribute, tuple
  - Operations on the relational model
- Views

- .. In the following:
  - Conceptual design: Entity-Relationship Model
  - Translation into relational databases

Literature

- Heuer/Saake: Datenbanken – Konzepte und Sprachen mitp press, 2000
- Georg Lausen: Datenbanken. Grundlagen und XML – Technologien, Spektrum Akademischer Verlag, 2005
- Gottfried Vossen: Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme, Oldenbourg Verlag, 2000
- Kifer, Bernstein, Lewis: Database Systems, Pearson International Edition, 2005
- Elmasri, Navathe: Fundamaentals of Database Systems, Benjamin/Cummings, 1994

2) Databases: Conceptual Design

Meike Klettke

meike.klettke@uni-greifswald.de or
meike@informatik.uni-rostock.de

Table of Contents

- Modelling
  - Characteristics of a model
  - Modelling process
- Entity-Relationship-Model (ERM)
  - Building blocks of the ERM
  - Design with the Entity-Relationship-Model
- Translation into the relational model
- Creation of a database schema

Motivation

- Why conceptual modelling ?
- Large applications/ databases are difficult to define
- First of all:
  - Determine which information shall be stored
  - Which connections exist between the information
- Discuss a design with a domain expert
- Aim:
  - early recognition of errors and incompleteness
  - documentation
  - Changes (during the use of a database) become easier

Models

- Models are used in each science field
- Herbert Stachowiak suggested 1973 a widely accepted model definition
  - *Mapping: A model is always an image of something, a representation of natural or artificial originals, which can be even again models.*
  - *Reduction: A model do not contain all characteristics of the original, but only those, which appear relevant to the designer.*
  - *Pragmatics: A model orients on its purpose. The questions for whom?, Why? and for what? are considered.*
- This definition don‘t depend on an application domain

Advantages of a Model

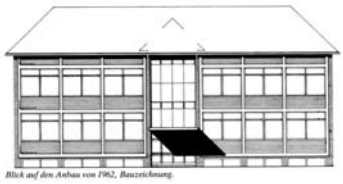
- a model contains abstraction, concentrates on some aspects only.
- Don‘t contain technical details that means is suited for communication between database designer and application experts.

Example for a Model

- Model of the Stephansdom in Vienna to a scale of 1:100-
- Model for blinds
- With lots of details
- For children to understand the dimensions of the church

Further Examples

Grundriß des Anbaues von 1962 und des Altbaues, 2. Stock.



Blick auf den Anbau von 1962, Bauzeichnung.

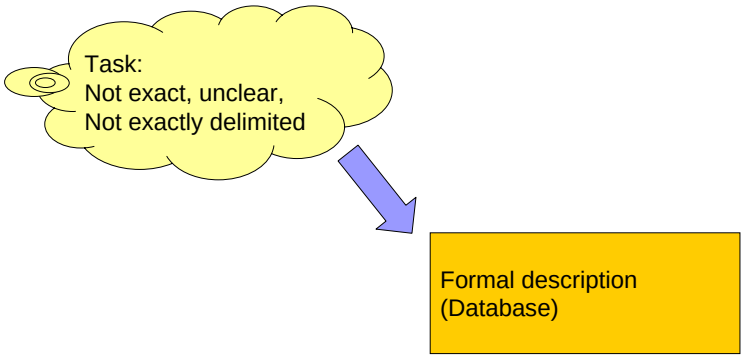
Another Example

...

- The model should be
  - as simple as possible
  - as complex as necessary

→ Contradiction between these two demands

Design task



Similar in Biskup: „Grundlagen von Informationssystemen“

Entity-Relationship-Model

suggested from Peter Chen (1976)

Basis building blocks:

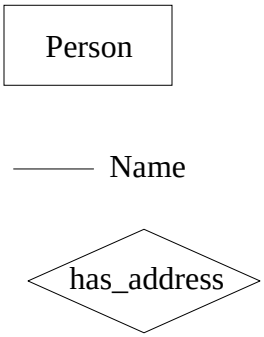
- Entities
- Relationships
- Attribute

Entity-Relationship-Model

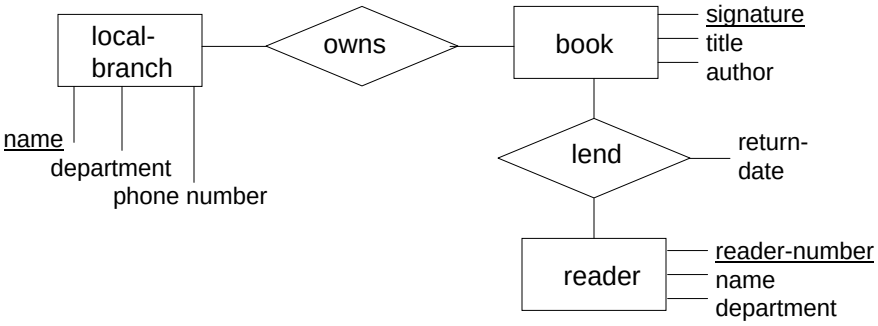
- Entities
  - Objects of the real world, we want tot store information about the objects
  - Other definitions/declarations: unit, term
  - Example: book, container, address, person, supplier, ...
  - Often nouns are used for naming the entities
- Relationships
  - Relationships between Entities
  - Example: Author writes book, company orders a Container, Person has an address
  - Often verbs are used for naming the relationships
- Attribute
  - Characteristic of an entity or relationship
  - examples: title of a book, name of an author, volume, city, zip code, company name

Graphical Representations

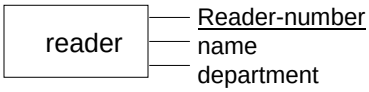
- Entity
- Attribute
- Relationships



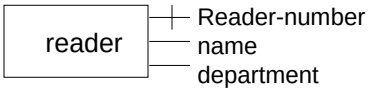
Simple Example



Defining Keys

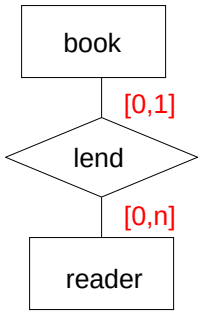


or



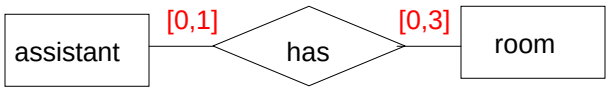
- Graphical Notation: key attributes are underlined or the connection between the attribute and the entity or relationship is marked

Cardinality Constraints/ 1



- Defines, how often an instance of an entity takes part in the relationship
- In brackets the minimum and maximum values are defined
- Default value: [0,n], (that means no restriction)
- Other typical values: Minimum 0 or 1
- Maximum: 1 or n (auch *)

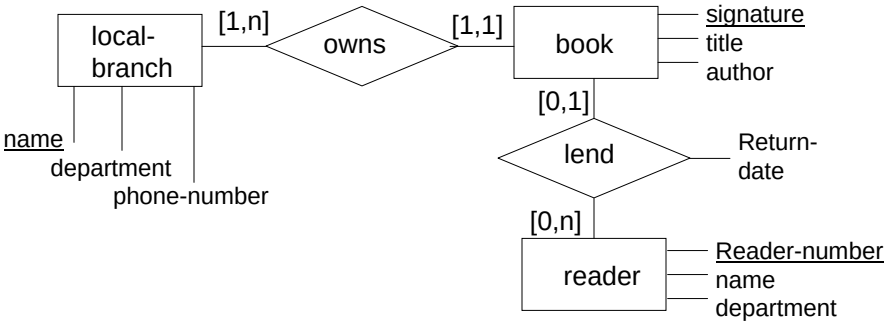
Cardinality Constraints / 2



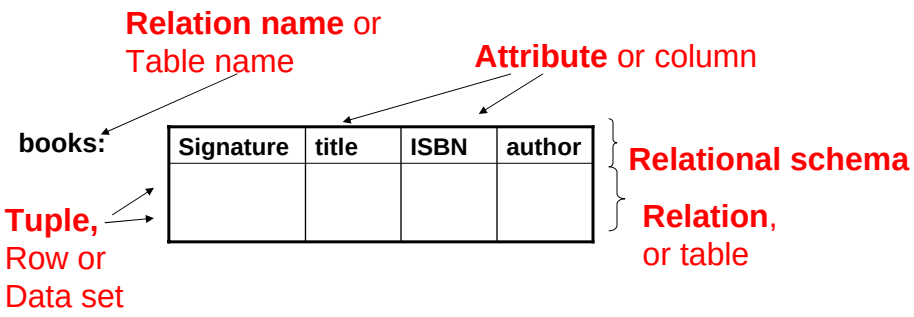
- an assistant normally has a room, there exist some assistants without a room (external assistants)
- In a room there are no more than 3 assistants

- each assistant can supervise several diploma theses, each diploma thesis is supervised by one assistant
- (Discussion of the examples)

Cardinality Constraints / 3

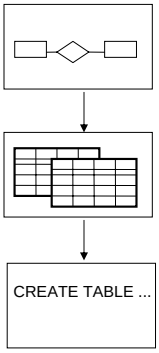


Repetition: Relational Model



Database Design Process

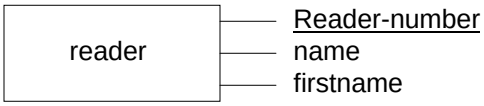
- Conceptual model (ERM)
- Translation of the ERM into the relational model
- Data definition



Maybe a stepwise process – (like each software development)

Translation into the Relational Model

- Entity is translated into a relation (table)
- Attributes of the entity become Attributes of the table

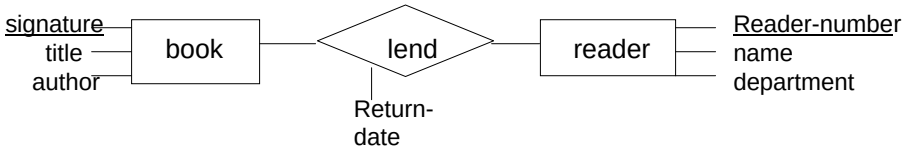


reader:

Reader-number	**Name**	**Firstname**
...		

- Primary key becomes the key of the relation

Translation into the relational model

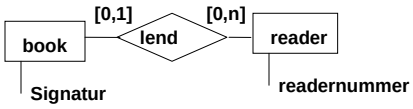


- Relationship is translated into a relation (table)
- The attributes of the relationship becomes attributes of the table
- Key attributes of the associated entities are added in the relation, too
- Foreign key references are defined

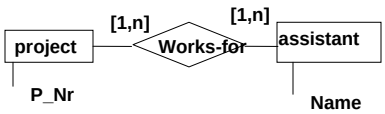
lend:

Signature	**readernummer**	**Return-date**
...		

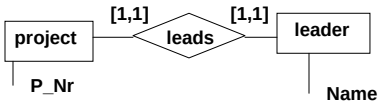
Keys of the Generated Relations



- Primary key: Signatur



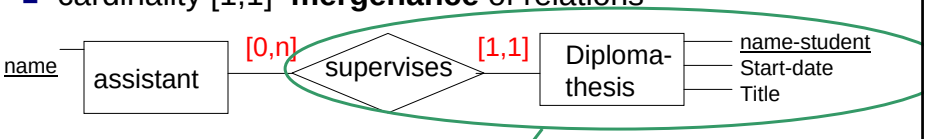
- Primary key: P_Nr AND Name



- Primary key: P_Nr OR Name

Optimising the Translation

- cardinality [1,1] **mergenance** of relations



- The relations **Diploma-thesis** und **supervises** are summerised

Diploma-thesis

name-student	**start-date**	**title**	**name** (foreign key to assistent.name)

Defining a Database

- SQL-DDL (Structured Query Language, Data Definition Language)

- CREATE TABLE
  - Creates a new table
- ALTER TABLE
  - Changes an existing table
- DROP TABLE
  - Deletes an existing table

Syntax of the CREATE TABLE Statement

```
CREATE TABLE  relation-name
(  attributename_1 domain_1 [NOT NULL] [DEFAULT defaultvalue]
      [PRIMARY KEY]
   ..
   attributename_n domain_n [NOT NULL]

PRIMARY KEY (attributename_i)
FOREIGN KEY (attributename_j) REFERENCES
                        relation-name_x (attribute_x)
);
```

[]- optional,
There are two possibilities for defining primary keys, immediately after the attribute declaration or at the end.
If a primary key contains two or more attributes than only the second method can be chosen.

Create Table Statement

reader:

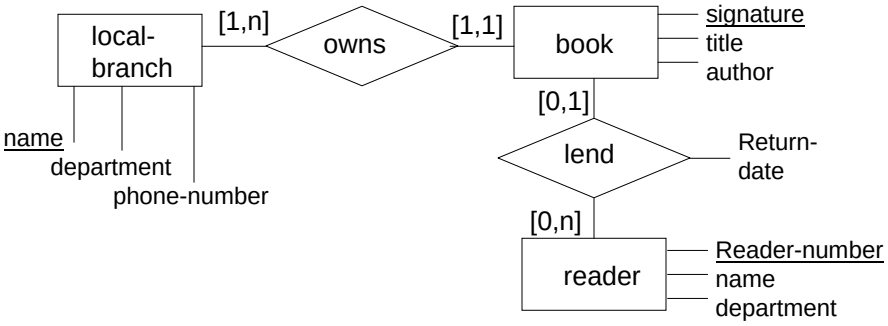
Reader-number	**name**	**firstname**
...		

```
CREATE TABLE reader
(
    Reader-number     VARCHAR(10) NOT NULL PRIMARY KEY,
    name              VARCHAR(30) NOT NULL,
    firstname         VARCHAR(30) NOT NULL
);
```

Domains

- We can associate a domain to the attributes
- for instance
  - INTEGER
  - FLOAT
  - CHARACTER (string with fixed length)
  - VARCHAR (strings with variable length)
  - DATE
  - TIME
  - ...
  - (more than 40 predefined domains)

The Complete Example: once again



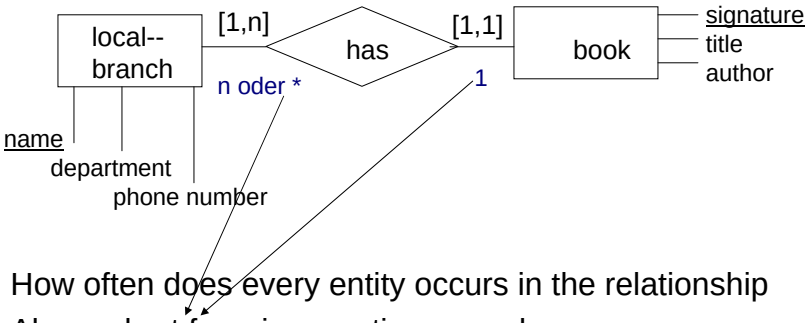
DDL (Data Definition Language) for the Complete Example

```
CREATE TABLE local-branch (
   name VARCHAR(10)
        NOT NULL PRIMARY KEY,
   department VARCHAR(10),
   phone-number VARCHAR(17));

CREATE TABLE book (
   signature VARCHAR(10) NOT
    NULL PRIMARY KEY,
   title VARCHAR(30),
   author VARCHAR(40),
   branch-name VARCHAR(10),
   FOREIGN KEY (branch-name)
    REFERENCES local-
    branch(name));
```

```
CREATE TABLE reader (
   reader-number  VARCHAR(10)
        NOT NULL PRIMARY KEY,
   name VARCHAR(30),
   department VARCHAR(4));

CREATE TABLE lend (
   signature VARCHAR(10)
       NOT NULL PRIMARY KEY,
   reader-number  VARCHAR(10),
   return-date DATE,
   name VARCHAR(30),
   FOREIGN KEY (reader-number)
       REFERENCES reader);
```

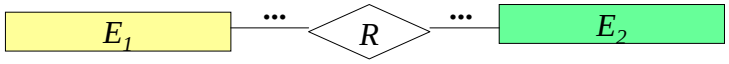
Unfortunately,

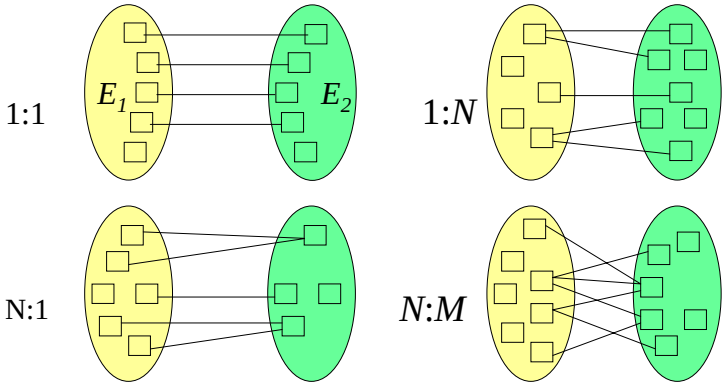
- there exist several syntaxes for specifying cardinalities
- Till now:



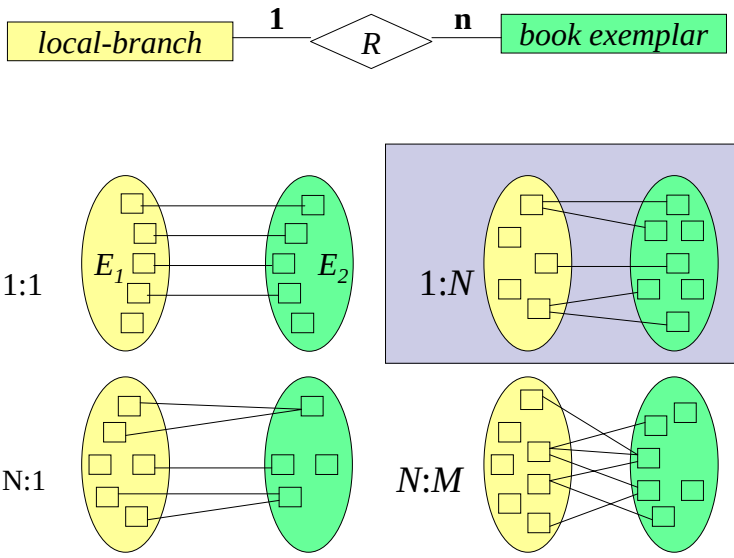
- How often does every entity occurs in the relationship
- Also a short form is sometimes used

Another Possibility for Defining Cardinalities





Cardinalities for the Example



Conclusion of this Part

- Conceptual design
  - Modelling databases with the Entity-Relationship-Model
  - Translation into the relational model
  - Defining a database schema with the Data Definition language
- There exist several extension of the (very simple) Entity-Relationship-Model
- Next tasks:
  - Normal forms of relational databases
  - Database queries

Literature

- Heuer/Saake: Datenbanken – Konzepte und Sprachen mitp press, 2000
- Kifer, Bernstein, Lewis: Database Systems, Pearson International Edition, 2005
- Foliensatz: Alfons Kemper, TU München

3) Normal Forms of Relational Databases

Meike Klettke
meike.klettke@uni-greifswald.de or
meike@informatik.uni-rostock.de

Normalising a Database Schema

- Aim of a normalisation process:
  - No redundant information
  - No Anomalies in the database
  - Simple structured relations
- Normalisation is a design task
  - After conceptual modelling and translation into relational databases a database is normalised

Sample relation

tenant-number	*tenant-name*	*object-number*	*object-adresse*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*owner-name*
c001	Holger Meyer	w01	Rostock, Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina Lange
		w05	Rostock, Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone Neumann
c002	Karsta Lehmann	w01	Rostock, Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina Lange
c003	Eva Schmidt	w06	Greifswald, Feldstr.4	2003-12-01	2006-08-01	270	E001	Tina Lange

Anomalies

= error in the database, incorrect states in the relations

- Insert anomaly
  - For example:
    - a new object is introduces, but till now not rented,
    - tenantnumber is part of the primary key and containts a null value
    - (not allowed in databases)
- Update anomaly:
  - A tenant has a new (family)name, changes are only made at one object
  - the others are forgotten
- Delete anomaly:
  - a tuple is deleted, if it was the last tupel for an object then
  - we loose the values for the address and the rent of this object

1. Normal Form

tenant-number	*tenant-name*	*object-number*	*object-adresse*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*owner-name*
c001	Holger Meyer	w01	Rostock, Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina Lange
		w05	Rostock, Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone Neumann
c002	Karsta Lehmann	w01	Rostock, Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina Lange
c003	Eva Schmidt	w06	Greifswald, Feldstr.4	2003-12-01	2006-08-01	270	E001	Tina Lange

1. Normal form: all attributes are atomar, no complex attributes, no repreating groups

In the example these attribute are atomar:
- tenantnumber,
- objectnumber,
- begin-rent,
- end-rent,
- rent, and
- ownernumber

Complex attributes are:
- tenantname
- objectadresse
- ownername

Repreating groups are:
- objectnumber, objectadresse, begin-rent, end-rent, rent, ownernumber, nnd ownername

1. Normal Form

- Several database system enable complex attributes, and sets or lists of attributes
- For example:

```
create row type address_t
( zip      INTEGER,
  city      VARCHAR(25),
  street  VARCHAR(20),
  no       INTEGER
);
```

```
create table hotel
( HotelID  INTEGER  NOT NULL
           PRIMARY KEY,
  name     VARCHAR(20) NOT NULL,
  address  address_t,
  phone    SET(INTEGER  NOT NULL)
);
```

- That means first normal form is not relevant any longer

2. Normal Form

tenant-number	*tenant-firstname*	*tenant-name*	*object-number*	*objectort*	*object-street*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*Owner-firstname*	*Owner-name*
c001	Holger	Meyer	w01	Rostock	Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina	Lange
c001	Holger	Meyer	w05	Rostock	Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone	Neumann
c002	Karsta	Lehmann	w01	Rostock	Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina	Lange
c003	Eva	Schmidt	w06	Greifswald	Feld-str.4	2003-12-01	2006-08-01	270	E001	Tina	Lange

2. normal form: each non-key attribute is full functional depending from the primary key

Primary key of the relation:
• tenantnumber, objectnumber

Functional Dependencies

- Belong to the local integrity constraints (like keys)
  - *A key specifies which attributes of a relation always have unique values and (because of that characteristic) can identify the tuples.*
  - with *functional dependencies* associations between attributes are determined. They are used for insuring the database integrity

Definition of Functional Dependencies

- But first the explanation:
  - A functional dependency between two attribute sets X and Y specifies, that the values of X determine the values of the attribute Y of a relation
- Functional dependencies are defined as follows
  - t_1 und t_2 are two tuple of a relation r,
  - the functional dependency $X \rightarrow Y$ is valid in r, if

$$\forall t_1, t_2 \in r : t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

Keys, formal

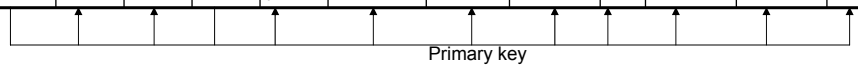
- A key in a relational schemata is defined as follows:
  - t_1 and t_2 are two tuples of a relation r
  - A key X is fulfilled in a relation r, if the projection onto the attribute set X delivers for different tuples different results:
  - That means:

$$\forall t_1, t_2 \in r, t_1 \neq t_2 \text{ gilt} : t_1[X] \neq t_2[X]$$

- That also means:
  - If X is a key of the relation, for all non-key-attributes Y: $X \rightarrow Y$

Functional Dependencies

tenant-number	*tenant-firstname*	*tenant-name*	*object-number*	*objectort*	*object-street*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*Owner-firstname*	*Owner-name*
c001	Holger	Meyer	w01	Rostock	Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina	Lange
c001	Holger	Meyer	w05	Rostock	Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone	Neumann
c002	Karsta	Lehmann	w01	Rostock	Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina	Lange
c003	Eva	Schmidt	w06	Greifswald	Feld-str.4	2003-12-01	2006-08-01	270	E001	Tina	Lange

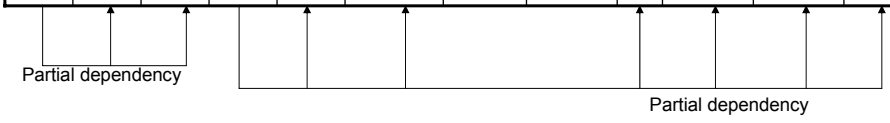


Primary keys of the relation:
- tenantnumber, objectnumber

Additional candidates for keys:
- tenantnumber, begin-rent
- objectnumber, begin-rent

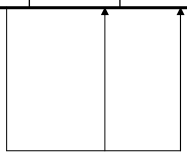
Functional Dependencies

tenant-number	*tenant-firstname*	*tenant-name*	*object-number*	*objectort*	*object-street*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*Owner-firstname*	*Owner-name*
c001	Holger	Meyer	w01	Rostock	Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina	Lange
c001	Holger	Meyer	w05	Rostock	Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone	Neumann
c002	Karsta	Lehmann	w01	Rostock	Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina	Lange
c003	Eva	Schmidt	w06	Greifswald	Feld-str.4	2003-12-01	2006-08-01	270	E001	Tina	Lange



Functional Dependencies

tenant-number	*tenant-firstname*	*tenant-name*	*object-number*	*objectort*	*object-street*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*Owner-firstname*	*Owner-name*
c001	Holger	Meyer	w01	Rostock	Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina	Lange
c001	Holger	Meyer	w05	Rostock	Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone	Neumann
c002	Karsta	Lehmann	w01	Rostock	Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina	Lange
c003	Eva	Schmidt	w06	Greifswald	Feld-str.4	2003-12-01	2006-08-01	270	E001	Tina	Lange



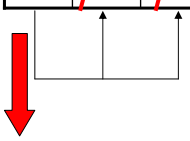
Transitive dependency

Process for Achieving the 2. Normal Form

- Input information: Partial functional dependencies
- For all partial functional dependencies:
  - A *new relation* is created that contains the attributes of the left and right side of the partial functional dependency
  - The *non-key-attributes* are deleted from the original relation

Example for the 2. Normal Form

tenant-number	*tenant-firstname*	*tenant-name*	*object-number*	*objectort*	*object-street*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*Owner-firstname*	*Owner-name*
c001	Holger	Meyer	w01	Rostock	Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina	Lange
c001	Holger	Meyer	w05	Rostock	Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone	Neumann
c002	Karsta	Lehmann	w01	Rostock	Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina	Lange
c003	Eva	Schmidt	w06	Greifswald	Feld-str.4	2003-12-01	2006-08-01	270	E001	Tina	Lange



(partial dependency)

tenantnumber	*tenantfirstname*	*tenantname*
c001	Holger	Meyer
c002	Karsta	Lehmann
c003	Eva	Schmidt

- Non-key-attribute are deleted in the original relation
- Key-attributes are kept in the original relation

Example for the 2. Normal Form

tenantnumber	*object-number*	*begin-rent*	*end-rent*
c001	w01	2004-01-01	2006-01-01
c001	w05	2005-12-01	2006-08-01
c002	w01	2006-01-01	2006-09-01
c003	w06	2003-12-01	2006-08-01

tenantnumber	*tenantfirstname*	*tenantname*
c001	Holger	Meyer
c002	Karsta	Lehmann
c003	Eva	Schmidt

objectnumber	*objectort*	*objectstreet*	*rent*	*ownernumber*	*owner-firstname*	*Owner-name*
w01	Rostock	Lange Str. 12	350	E001	Tina	Lange
w05	Rostock	Kurze Str. 12	450	E002	Simone	Neumann
w06	Greifswald	Feldstr.4	270	E001	Tina	Lange

Which anomalies are avoided by the second normal form?

3. Normal Form

tenant-number	*tenant-firstname*	*tenant-name*	*object-number*	*objectort*	*object-street*	*begin-rent*	*end-rent*	*rent*	*owner-number*	*Owner-firstname*	*Owner-name*
c001	Holger	Meyer	w01	Rostock	Lange Str. 12	2004-01-01	2006-01-01	350	E001	Tina	Lange
c001	Holger	Meyer	w05	Rostock	Kurze Str. 12	2005-12-01	2006-08-01	450	E002	Simone	Neumann
c002	Karsta	Lehmann	w01	Rostock	Lange Str. 12	2006-01-01	2006-09-01	350	E001	Tina	Lange
c003	Eva	Schmidt	w06	Greifswald	Feld-str.4	2003-12-01	2006-08-01	270	E001	Tina	Lange

(transitive depencency of the original relation)

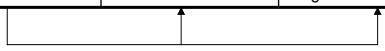
Transitive dependency:
If A→B and B→C then A→C, C is *transitive dependent on* A

The 3. normal form is fulfilled, if a relation is in 2. normal form and no non-key-attribute is dependent on the attributes of a primary key.

In relations that are not in 3. normal form, update anomalies can occur

3. Normal Form

objectnumber	*objectort*	*objectstreet*	*rent*	*ownernumber*	*Owner-firstname*	*Owner-name*
w01	Rostock	Lange Str. 12	350	E001	Tina	Lange
w05	Rostock	Kurze Str. 12	450	E002	Simone	Neumann
w01	Rostock	Lange Str. 12	350	E001	Tina	Lange
w06	Greifswald	Feldstr.4	270	E001	Tina	Lange



Transitive dependency:

If (objectnumber → ownernumber) and
(ownernumber→ owner-firstname, owner-name
then (objectnumber → owner-firstname, owner-name)

is transitive dependent

Process for Achieving the 2. Normal Form

objectnumber	*objectort*	*objectstreet*	*rent*	*ownernumber*	*Owner-firstname*	*Owner-name*
w01	Rostock	Lange Str. 12	350	E001	Tina	Lange
w05	Rostock	Kurze Str. 12	450	E002	Simone	Neumann
w01	Rostock	Lange Str. 12	350	E001	Tina	Lange
w06	Greifswald	Feldstr.4	270	E001	Tina	Lange

ownernumber	*Owner-firstname*	*Owner-name*
E001	Tina	Lange
E002	Simone	Neumann

Attributes of the left and the right side of a transitive dependency create a new relation,
The attributes of the right side of the transitive dependency are deleted from the original relation

Conclusion of this Part

- Normal forms avoid anomalies in databases,
- Help to ensure that data sets are correct
- Normalisation leads to a very fragmented storage of information
- That means: query formulation is more difficult and query execution is more time-consuming
- Both characteristics: during design it has to be decided between
  - Normalisation (correct database states) and
  - efficiency of query processing
- Sometimes relation are not completely normalised because of query processing efficiency

Literatur:

- Thomas Connolly, Carolyn Begg: Database Systems: A Practical Approach to Design, Implementation, and Management, Addison Wesley, 2002
- Meike Klettke: Akquisition von Integritätsbedingungen in Datenbanken, Dissertation, Universität Rostock, 1997

4) Structured Query Language (SQL)

Meike Klettke
meike.klettke@uni-greifswald.de or
meike@informatik.uni-rostock.de

Table of Contents

- SQL: Introduction
- Demands on database query languages
- Building blocks of SQL
- Simple SQL queries
- Complex SQL queries
- Conclusion

SQL

- Standardise
  - Data definition (DDL)-
  - Data manipulation (DML) and
  - Query language
- In this section:
  - query language
  - Bases on the algebra (operations on the relational model: selection, projection, join, renaming, union, intersection)
- Available standards: SQL 99 and SQL 2003

Queries - Introduction

- Query = List of operations that deliver a *relation* as a result
- Queries
  - can be formulated interactively or
  - can be written in a program
- Queries are necessary for using the stored data

Demands for Query Languages

- ad-hoc formulation
  - A user shall write a query without programming anything
- A user shall describe which information he wants to get (and not how the information is stored and how the result is generated)
- The result of a query (a table) can be the input information for another query
- the language has to be optimised (only a few operators, optimisation rules are available)
- Furthermore: a query must not run into an infinite loop

Repetition: Operations on the Relations

- Select columns (Projection)
- Select rows (tuples) (Selection)
- Cartesian product
- Join: combining relations over attributes with same attribute names and identical values
- Union of Tables
- Renaming of rows: important for Join and Union

SQL-Syntax – bird's eye view

select - which attributes shall be in the result

from - from which relation

Where - which conditions have to be valid

Sample Relations / 1

books

signature	title	author	local-branch
2937	Datenbanken - 1	Heuer/Saake	FBIN
2928	Datenbanken - 2	Saake/Heuer	FBIN
9736	Database Systems	Date	ING
2393-2	ER Modellierung	Thalheim	FBIN
6430	Informationssysteme	Biskup	FBIN
4957	Datenbank-Handbooks	Lockemann	ING

local-branch

name	department	phone-number
FBIN	FBIN	498 3292
ING	FBET	498 3872

Sample Relations / 2

reader

reader-number	name	department
1239848	Meyer	FBIN
2123304	Lehmann	FBET
3912817	Schulz	WIWI
3293948	Meyer	FBET

lean

signature	reader-number	return-date
2937	1239848	2006-01-20
2928	1239848	2006-01-20
2393-2	1239848	2006-01-27
6430	2123304	2006-01-02
4957	2123304	2006-01-20

The first SQL Query

- **Query:**

select title, author
from books
where signature = ´4957´;

- **Result:**

```
title                                author
------------------------------------ ----------------
Datenbank-Handbooks                  Lockemann
```

Further SQL Examples / 1

- **Query:**

select * **(← all attributes)**
from books;

- **Result:**

signature	title	author	local-branch
2937	Datenbanken - 1	Heuer/Saake	FBIN
2928	Datenbanken - 2	Saake/Heuer	FBIN
9736	Database Systems	Date	ING
...			

Further SQL Examples / 2

- **Query:**

select *
from books, lean;

- **Result:**
30 tuples, all attributes from both relations,
all combinations of values

from - clause

- Enumerates the used relations
- Renaming can take place

select
from
where

- Example:

from reader, lean
from reader r, lean l

select - clause

- Enumeration of attributes or
- * for all attributes

select
from
where

- List of projection attributes of a query
- Example:
  - **select** title, author
  - **select** books.title, lean.return-date

select - clause

- Additionally possible:
- So called aggregate functions
  - Sum, min, max, avg (mean value), count (number)
- Arithmetic expressions
  - + - * / (only on numeric domains)
  - **substring, current_date, current_time**

- Examples:
  - **select current_date** - lean.return-date
  - **select** price * 1.19
  - **select max** (reader-number)

select
from
where

where – clause / 1

Condition that has to be fulfilled

- comparison attribute – constant value
- comparison attribute – attribute
- **between**: attribute between two constant values

- Examples:

where name = ´Meyer´
where return-date = **current_date**
where return-date **between** ´2006-10-20´ **and** ´2006-10-25´
where lean.signature = books.signature;

select
from
where

where – clause / 2

- Examples:
  where author = ´Date´
  where cost-price > market-price
  where price **between** 20 **and** 30
  where (title **like** ´%Datenbank%´) **or**
  (title **like** ´%database%´)
  where author **like** ´%SAAKE%´

select
from
where

- Some possibilities summarised:
  = , > , < , <=, >=, **like** , + , - , * , / , **substring, current_date, current_time, count, max, min, avg, sum**

Aggregate Functions

- **count, sum, min, max, avg**
- We are looking for the oldest book, the newest one and the average year of publication

select **min** (year), **max** (year), **avg** (year)
from book

select avg (Semester)
from Studenten;

Aggregate Functions /2

- Number of employees
-

```
select    count (*)
from      employee
```

- Number of employees in the research department

```
select    count (*)
from      employee, department
where     DNO=DNUMBER and DNAME='Research'
```

Sample database to the query from slide 20

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	1986-04-05	DAUGHTER
333445555	Theodore	M	1983-10-25	SON
333445555	Joy	F	1958-05-03	SPOUSE
987654321	Abner	M	1942-02-28	SPOUSE
123456789	Michael	M	1988-01-04	SON
123456789	Alice	F	1988-12-30	DAUGHTER
123456789	Elizabeth	F	1967-05-05	SPOUSE

Some Complete Queries /1

- **„Inventory“**
- Query:

  select count(*)
  from books
  where local-branch =´FBIN´;

- Result:

  1

  4

Some Complete Queries /2

- Which books are borrowed by the reader with the name „Meyer“?
- Query:

  select books.title, books.author
  from books, lean, reader
  where (books.signature=lean.signature) **and**
  (lean.reader-number = reader.reader-number) **and**
  (reader.name='Meyer');

- Result:

title	author
ER Modellierung	Thalheim
Datenbanken - 2	Saake/Heuer
Datenbanken - 1	Heuer/Saake

Some complete queries /3

- Which books has the reader „Meyer“ to give back now?
- Query:

```
select books.title, books.author
from books, lean, reader
where (books.signature=lean.signature) and
        (lean.reader-number = reader.reader-number) and
        (reader.name='Meyer') and
        (lean.return-date <= current_date);
```

- Result:
  0

Comparing Strings

wildcard "%" ; "_"

- "%" for a list of characters (also empty set)
- "_" for exactly one character

```
select *
from reader
where name like ´Me_er´;

select distinct name
from reader
where name like ´Schul%´;
```

Some Complete Queries / 4

- Search for database- or SQL-books
- Query:

  select title, author, local-branch

  from books

  where title **like** ´%Datenbank%´ **or** title **like** ´%SQL%´
- Result:

title	author	local-branch
Datenbanken - 1	Heuer/Saake	FBIN
Datenbanken - 2	Saake/Heuer	FBIN
Datenbank - Handbooks	Lockemann	ING

Removal of Duplicate Values

select distinct department

from reader

department
FBIN
FBET
WIWI

Union of Relations

```
select name, department
from reader
union
select name, ´computercentre´
from employee_computercentre;
```

Result:

```
1                    2
-------------------- -------------
Lehmann              FBET
Meyer                FBET
Meyer                FBIN
Mueller              Rechenzentrum
Bayer                Rechenzentrum
...
```

Prerequisite: compatible datatypes

Sorting Relations

- Sorting of tuples with order by clause

select * from books **order by** signature;

- Result:

```
signature   title                    author             local-branch
----------  ------------------------ ------------------ ----------------
2393-2      ER Modellierung          Thalheim           FBIN
2928        Datenbanken - 2          Saake/Heuer        FBIN
...
```

select * from books **order by** author;

- Result:

```
signature    title                         author     local-branch
-----------  ----------------------------- ---------- ------------------
6430         Informationssysteme           Biskup     FBIN
...
```

Sorting Relations

- Sorting of tuples with order by clause

select * from books **order by** local-branch **asc**, signature **asc**;

- Result:

.signature	title	author	local-branch
2393-2	ER Modellierung	Thalheim	FBIN
2928	Datenbanken - 2	Saake/Heuer	FBIN
2937	Datenbanken - 1	Heuer/Saake	FBIN
6430	Informationssysteme	Biskup	FBIN
4957	Datenbank-Handbooks	Lockemann	ING
9736	Database Systems	Date	ING

desc: also exists, reverse order, beginning with the largest value

Null Values

- Unknown value
- Different meanings (not existent, not relevant, unknown, will be added later)
- Null values can be created from queries that contains outer joins

- Sometimes „suprising“ results if null values exists
- For example:

select count (*)
from students
where semester < 13 **or** semester > =13

  - students with *Semester* attribute = **null** are not counted
  - reason: next slide

Querying Tuples with Null Values

1. In arithmetic expressions: if an operand is **null**, the the result becomes **null**. For example:
   - **null** + 1 = **null** but also
   - **null** * 0 = **null**
2. SQL uses a logic with three values: **true**, **false, and unknown**. If two values are compared and at least one is unknown then the result **unknown is generated**
3. Logical expressions are calculated accordingly to the tables (see next slide)

and	true	false	unknown
true	true	false	unknow
false	false	false	false
unknown	unknown	false	unknown

or	true	false	unknown
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

not	
true	false
false	true
unknown	unknown

Null Values in Queries

- Because of the special meaning of null values, these can occur in queries
- For example:

```
select    firstname, lastname
from      employee
where     superssn  is null
```

The query execution delivers **true** or **false** for the **where** Clause, tuples for which the result is true are added into the result relation

More Complex Queries

- SQL queries can be nested
- The result of each SQL query is a relation, other queries can be executed on this relation

- Most times nesting is realised in the **where**-clause,
  - Also possible in the **from**-clause
  - Subqueries, that deliver an atomic value as result can occur everywhere

„Pattern“ for nesting queries

```
select select_list
from table1 alias1
where expr operator (select column_list
                     from table2 alias2
                     where alias1.column operator alias2.column);
```

Nested Queries / 1

- Combination with: **in, not in (subset characteristic)** or **exists** or **not exists (check if results are empty or not)**
- // all readers, that actually borrowed a book or some books

```
select * from reader
where reader-number in
   (select reader-number from lean);
```

- Result:

reader-number	name	department
1239848	Meyer	FBIN
2123304	Lehmann	FBET

Nested Queries / 2

- // all books, that are available (not leaned)

```
select *
from books
where not exists
    (select *
     from lean
     where lean.signature= books.signature);
```

- Result:

signature	title	author	local-branch
9736	Database Systems	Date	ING

Nested Queries /3

- subquery in the **where**-clause
- Which examination results are better than the average result

```
select *
 from examination
 where result < ( select avg (result)
                  from examination);
```

Joins in SQL

- There exist several possibilities to join relations
- Till now we have seen the cartesian product
- Other methods are
  - Join on
  - Left | right | full outer join on

select *
from R_1, R_2
where $R_1.A = R_2.B$;

select *
from R_1 **join** R_2 **on** $R_1.A = R_2.B$;

Joins in SQL

```
select    Fname, Lname, address
from      (employee join department on
          DNUMBER=DNO)
where     Dname='Research'
```

Other Joins

• right outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟖

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
-	-	c_3	d_2	e_2

• left outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟕

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-

Other Joins

• full outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟗

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-
-	-	c_3	d_2	e_2

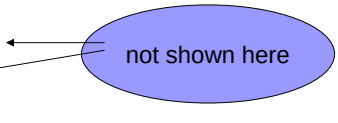
Example for Outer Joins

select title, author, local-branch, reader-number
from (book **left outer join** lean **on**
book.signature = lean.signature)
where title like '$ Datenbank $'

SQL: Conclusion

- An SQL statement consists of up to clauses, the first two are mandatory, the others optional

- **select** **<attribute list>**
  from **<table list>**
  [where **<condition>]**
  [group by **<grouping attribute(s)>]**
  [having **<group condition>]**
  [order by **<attribute list>]**



Summary of SQL Queries (cont.)

- The **select**-clause lists the attributes or functions to be retrieved
- The **from**-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The **where**-clause specifies the conditions for selection and join of tuples from the relations specified in the from-clause
- **group by** specifies grouping attributes
- **having** specifies a condition for selection of groups
- **order by** specifies an order for displaying the result of a query

- A query is evaluated by first applying the **where**-clause, then **group by** and **having**, and finally the **select**-clause

Literature

- Slides from Alfons Kemper, TU München
- Several SQL tutorials exists, for instance:
  - http://www.w3schools.com/sql/
  - http://sql.1keydata.com/de/

- Oracle:SQL:Documentation http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96540/toc.htm

5) Storing XML Documents with Databases

Meike Klettke
meike.klettke@uni-greifswald.de or
meike@informatik.uni-rostock.de

Motivation

- Current Situation
  - Increasing number of XML applications
  - Numerous XML documents available
- Storage of XML documents
  - For several applications necessary
  - Efficient storage, realisation of queries (all advantages of database systems)

Table of Contents

- Classification of XML documents
- Overview on methods for storing XML documents
  - Storage as files / clobs
  - Native storage of XML documents
  - Structured storage in databases
- Summary, References

Classification of XML Documents

- Data-centric documents

  (structured, regular, typed
  examples: product catalog, order, invoice)

- Document-centric documents

  (unstructured, irregular, untyped
  examples: article, book, Email, webpage)

- Mixed approaches

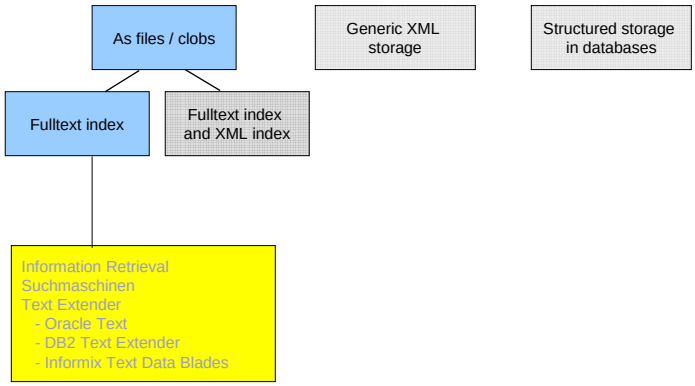
  (data-centric and document-centric portions
  examples: publications, online bookstore)

```
<order>
  <customer>Meyer</customer>
  <position>
    <isbn>1-234-56789-0</isbn>
    <number>2</number>
    <price currency=„Euro“>30.00</price>
  </position>
</order>
```

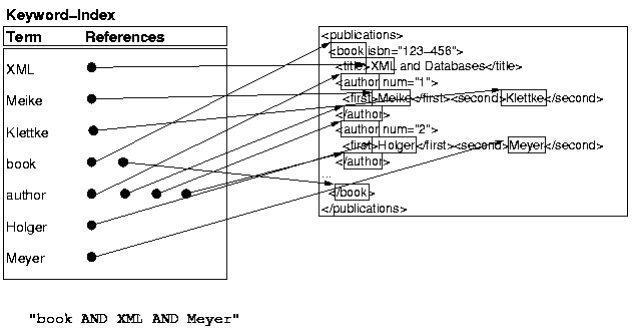
```
<content>
XML builds on the principles of two existing
 languages, <emph>HTML</emph> and
<emph>SGML</emph> to create a simple
mechanism ..
The generalized markup concept ..
</content>
```

```
<book>
  <author>Neil Bradley</author>
  <title>XML companion</title>
  <isbn>1-234-56789-0</isbn>
  <content>
    XML builds on the principles of two existing
    languages, <emph>HTML</emph> and ..
  </content>
</book>
```

Overview on Methods for Storing XML Documents

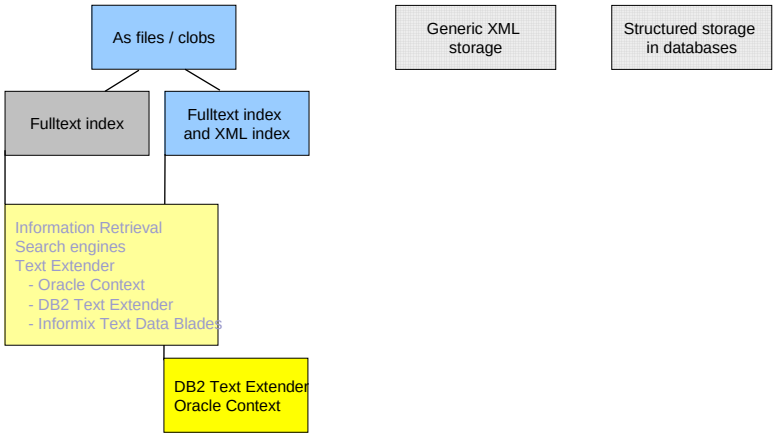


Fulltext Index

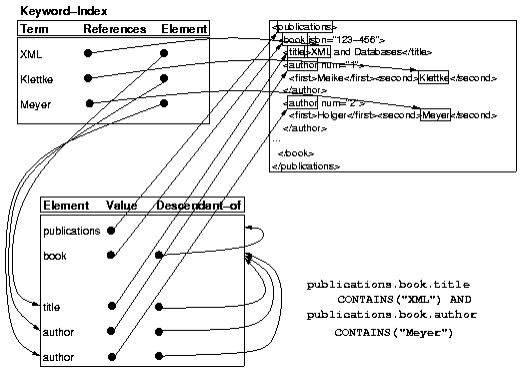


- well-known method
- Boolean retrieval (AND, OR, NOT)

Methods for Storing XML Documents

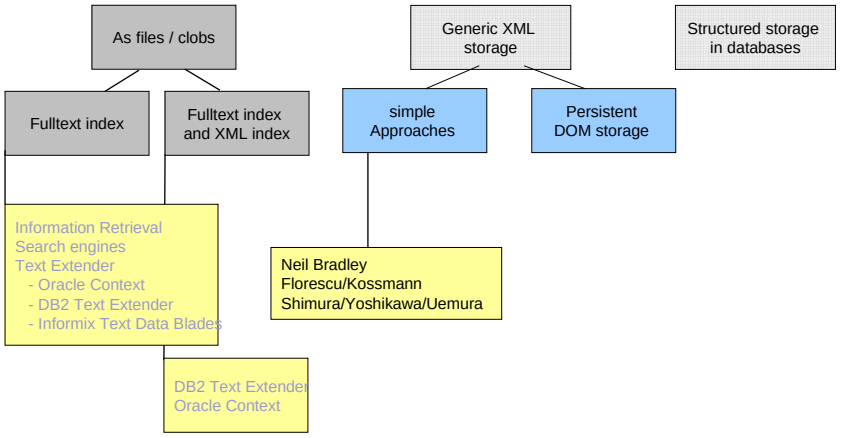


Fulltext Index and XML Index

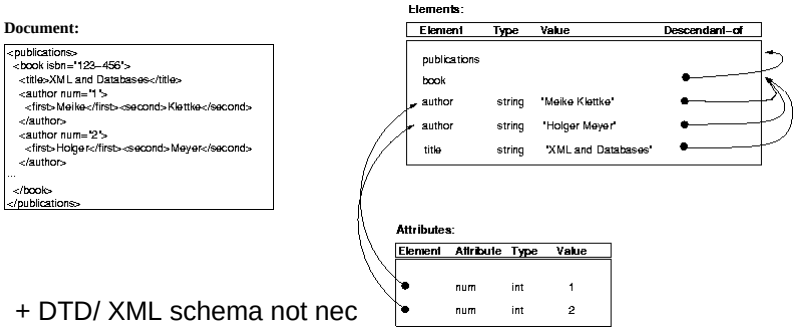


+ XML structure can be considered in queries

Methods for Storing XML Documents

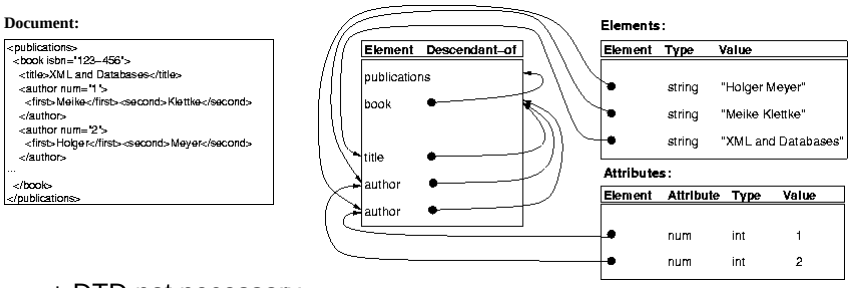


Generic tree approach / 1



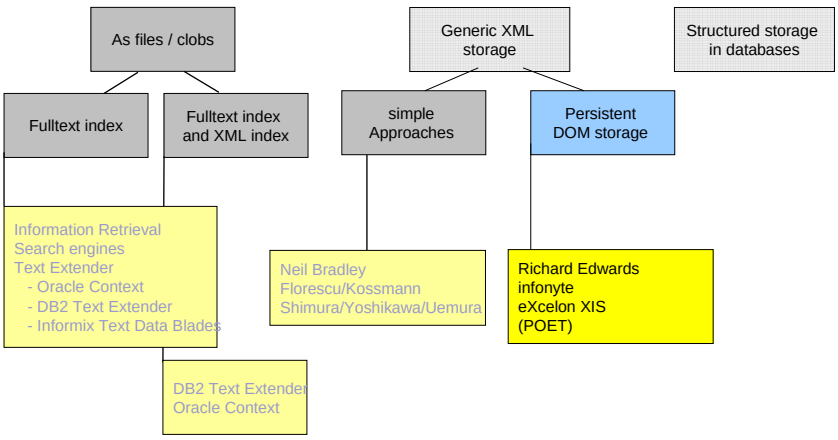
+ DTD/ XML schema not nec
+ Usable for simple queries
+ Datatypes
- Queries on two or more elements/attributes not efficient

Generic tree approach / 2



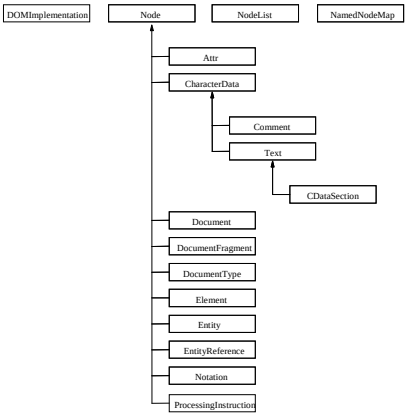
+ DTD not necessary
+ Usable for simple queries
+ Datatypes
- Queries on two or more elements/attributes not efficient

Methods for Storing XML Documents



Storage based on the Document Object Model

- Information of the Document Object Models represent the database schema
- Usage of relational or object orientied databases or
- Development of new storage components



Storage based on the Document Object Model

Methods of the class Node:
- getChildren()
- getFirstChild()
- getNextSibling()
- getNodeType()
- getParentNode()
- getPreviousSibling()
- hasChildren()

NodeID	NodeType	DocID	ParentNode

PreviousSibling	NextSibling	FirstChild

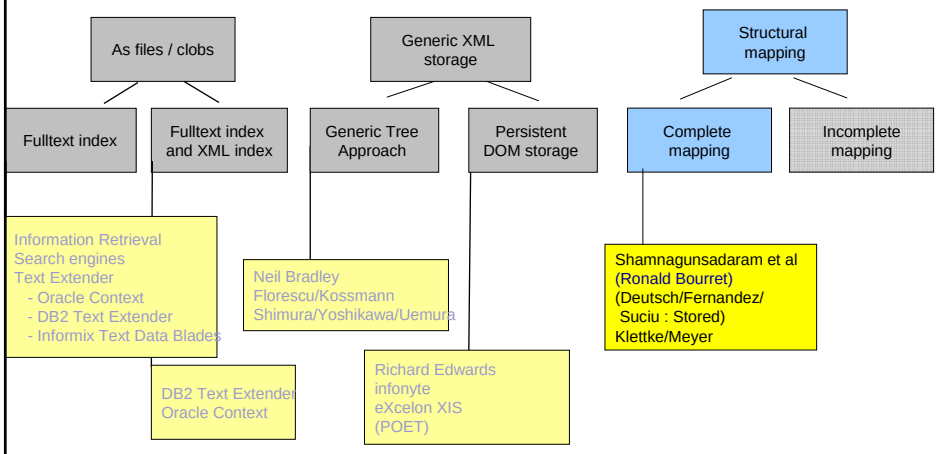
Methods of class Element:
- getAttributes()
- getElementsByTagName(String)
- getTagName()

NodeID	TagName

Methoden of the class Attribute:
- getName()
- getValue()

NodeID	ElementID	AttributName	AttributValue

Methods for Storing XML Documents



Structural mapping to relational databases

Document

```
<publications>
 <book isbn="123–456">
  <title>XML and Databases</title>
  <author num="1">
   <first>Meike</first><second>Klettke</second>
  </author>
  <author num="2">
   <first>Holger</first><second>Meyer</second>
  </author>
...
 </book>
</publications>
```

Database

Publications

ID	Book	...
0001	1001	...

Book

ID	ISBN	Title	Author	
1001	123–456	XML and Databases	2001	2002

Author

ID	First	Second
2001	Meike	Klettke
2002	Holger	Meyer

- \- DTD is necessary
- \+ Queries use SQL functionality
- \+ Datatypes

Structural mapping to object relational Databases

Document

```
<publications>
 <book isbn="123-456">
  <title>XML and Databases</title>
  <author num="1">
   <first>Meike</first><second>Klettke</second>
  </author>
  <author num="2">
   <first>Holger</first><second>Meyer</second>
  </author>
 ...
 </book>
</publications>
```

Database

Publications:

Book				
ISBN	Title	Author		...
		First	Second	
123–456	XML and Databases	Meike	Klettke	...
		Holger	Meyer	...

- \- DTD is necessary
- \+ Queries use SQL functionality
- \+ Datatypes
- \- Sparsely populated databases

Mapping of a DTD to object relational Databases (straight-forward)

- Sequence of Elements - Attributes of a relation
- Optional element - Nullable attribute
- Attribute - Database attribute (NOT NULL, DEFAULT VALUE)
- Elements with Quantifier +,* - Set or list of attributes (SET OF, LIST OF)
- Nested elements - TUPLE OF

Example for straight-forward Mapping / 1

```
<!ELEMENT book (front, body, references)>
<!ATTLIST   book isbn CDATA #REQUIRED>
<!ELEMENT front  (title, author+, edition?, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (first, second, e-mail?)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT second (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
```

book:

isbn	front						body	references
	title	author			edition	publisher		
		first	second	e-mail				

Example for straight-forward Mapping / 2

XML document:

```
<!ELEMENT book (front, body, references)>
<!ATTLIST   book isbn CDATA #REQUIRED>
<!ELEMENT front  (title, author+,
                  edition?, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (first, second, e-mail?)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT second (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
```

Informix syntax:

```
CREATE TABLE book
( isbn VARCHAR(20) NOT NULL,
  front ROW
    ( title VARCHAR(100) NOT NULL,
      author LIST
        (ROW
          ( first VARCHAR(30) NOT NULL,
            second VARCHAR(50) NOT NULL,
            email VARCHAR(30)))  NOT NULL,
      edition VARCHAR(20),
      publisher VARCHAR(40) NOT NULL),
    body ...
    references ...
);
```

Mapping Problems: Alternatives

Example: <!ELEMENT publications (book | article | conference)*>

- all alternatives in one relation

id	book		article	conference
	author	title		
0001	Bradley	XML comp.		
0002			Stored	
0003				DevCon

- split into separate relations

id	book	
	author	title
0001	Bradley	XML comp.

id	article
0002	Stored

id	conference
0003	DevCon

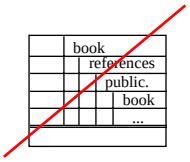
- attribute of type XML

publications
<book> <author>Bradley</author> <title>XML companion</title> </book>
<article>Stored</article>
<conference>DevCon</conference>

Mapping Problems: Recursion and Mixed Content

- Recursion (in DTD) - mark nodes
  - split into separate relations
  - using references (primary/foreign key)

  Example:
  <!ELEMENT publications (book | article | conference)*>
  <!ELEMENT book (front, body, references)>
  <!ELEMENT references (publications+)>



- Mixed content type - attribute of type XML

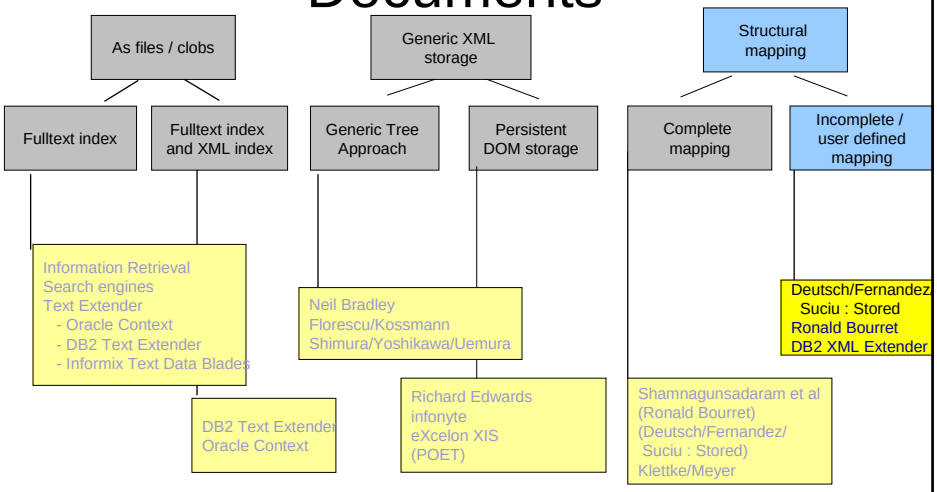
  Example:

...	content
...	XML builds on the principles of two existing languages, <emph>HTML</emph> and <emph>SGML</emph> to create a simple mechanism ..

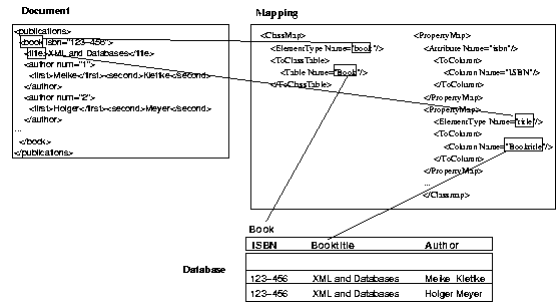
Pro/Cons of using Object-Relational Databases

- Pro: structured data
  - Queries, datatypes, aggregations, views
  - Integration with other databases, federation
- Cons: semi- and un-structured data
  - Large Schema, sparsely populated databases, lots of null values
  - Flexible types, alternatives (= characteristics of semistructured data)
  - Information retrieval, fulltext operations
- Conclusion:
  - usable for data-centric documents,
  - not suited for document-centric parts

Methods for Storing XML Documents



User defined Mappings



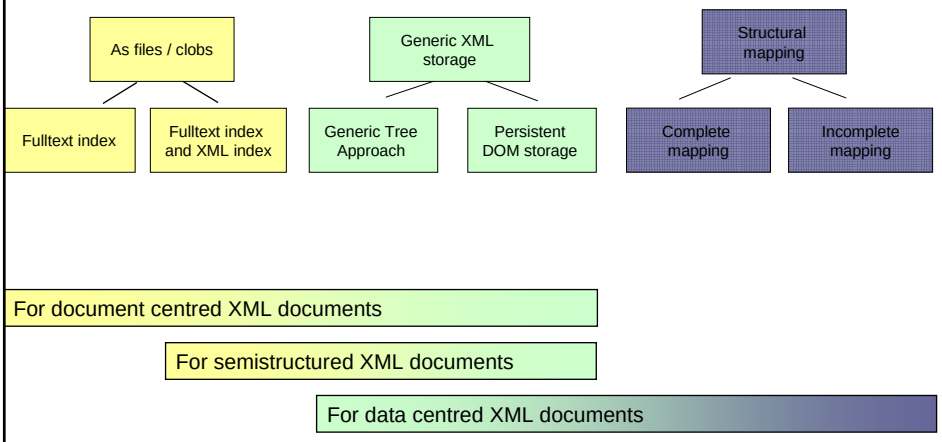
+ Variable method
+ Integration of XML documents in existing databases
- User has to specify the mapping

User defined Mappings

Oracle: XML schema extension

```
<xs:schema … >
 <xs:element name="PurchaseOrder" type="PurchaseOrderType"
       xdb:defaultTable="PURCHASEORDER"/>
   <xs:complexType name="PurchaseOrderType"
       xdb:SQLType="PURCHASEORDER_T"> <xs:sequence>
 <xs:element name="Reference" type="ReferenceType"
       minOccurs="1" xdb:SQLName="REFERENCE"/>
 <xs:element name="Actions" type="ActionsType"
       xdb:SQLName="ACTIONS"/>
 <xs:element name="Reject" type="RejectionType" minOccurs="0"
       xdb:SQLName="REJECTION"/>
 <xs:element name="Requestor" type="RequestorType"
       xdb:SQLName="REQUESTOR"/>
 <xs:element name="User" type="UserType" minOccurs="1"
       xdb:SQLName="USERID"/>
…
```

Application of XML storage methods



Summary

- Variety of techniques for storing XML documents available
- Best algorithm depend on the document characteristics (data-centric, document-centric, mixed)
- Commercial database (Oracle, db2, MS-SQLServer) systems support several methods:
  - Based o text indexing
  - Native storage and
  - Structured storage in databases

References / 1

- Neil Bradley, *The XML companion*, Addison Wesley, 1998
- Takeyuki Shimura, Masatoshi Yoshikawa, Shunsuke Uemura, *Storage and Retrieval of XML Documents Using Object-Relational Databases*, DEXA 1999, pp. 206-217
- Daniela Florescu, Donald Kossmann, *Storing and Querying XML Data Using an RDBMS*, Bulletin of the Technical Committee on Data Engineering, volume 22, number 3, September 1999, pp. 27-34
- Jayaval Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, Jeffrey Naughton, *Relational Databases for Querying XML Documents - Limitations and Opportunities*, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp. 302-314
- Meike Klettke, Holger Meyer, *XML and Object-Relational Databases - Enhancing Structural Mappings Based on Statistics*, WebDB 2000, Mai 2000

References / 2

- Ronald Bourret, C. Bornhövd, A.P. Buchmann, *A Generic Load/ Extract Utility for Data Transfer between XML Documents and Relational Databases*, Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, WECWIS'00
- Ronald Bourret, *XML-DBMS - Middleware for Transferring Data between XML Documents and Relational Databases*, http://www.rpbourret.com/xmldbms/index.htm
- Alin Deutsch, Mary F. Fernandez, Dan Suciu, *Storing Semistructured Data with STORED*, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1999, Philadephia, Pennsylvania, USA, ACM Press, 1999, pp. 431-442
- Meike Klettke, Holger Meyer, *XML and databases*, http://www.xml-und-datenbanken.de