Database Design and Programming

Peter Schneider-Kamp

DM 505, Spring 2012, 3rd Quarter

Course Organisation

- Literature
  - *Database Systems: The Complete Book*
- Evaluation
  - Project and 1-day take-home exam, 7 scale
- Project
  - Design and implementation of a database using PostgreSQL and JDBC
- Schedule
  - 4/2 lectures a week, 2/4 exercises a week

Course Organisation

- Literature
  - *Database Systems: The Complete Book*
  - Chapters 1 & 2 available online
  - Chapter 5.1 on Blackboard
  - book available from SDU book store

(Preliminary) Course Schedule

Week	Room	05	06	07	08	09	10	11
Tue 14-16	U151	L Fri 12-14	E	E	E	E	E	E
Wed 10-12	U151	L	L	L	L	L Fri 08-10	L	L
Thu 14-16	U151	L	E	L	E	L	E	L

- some exercises in terminal room
- 1st in Week 07

Where are Databases used?

It used to be about boring stuff:

- Corporate data
  - payrolls, inventory, sales, customers, accounting, documents, ...
- Banking systems
- Stock exchanges
- Airline systems
- ...

Where are Databases used?

Today, databases are used in all fields:

- Web backends:
  - Web search (Google, Live, Yahoo, ...)
  - Social networks (Facebook, ...)
  - Blogs, discussion forums
  - ...
- Integrating data (data warehouses)
- Scientific and medical databases
- ...

Why are Databases used?

- Easy to use
- Flexible searching
- Efficiency
- Centralized storage, multi-user access
- Scalability (large amounts of data)
- Security and consistency
- Abstraction (implementation hiding)
- Good data modeling

Why learn about Databases?

- Very widely used
- Part of most current software solutions
- DB expertise is a career asset
- Interesting:
  - Mix of different requirements
  - Mix of different methodologies
  - Integral part of data driven development
  - Interesting real world applications

Short History of Databases

- Early 60s: *Integrated Data Store*, General Electric, first DBMS, network data model
- Late 60s: *Information Management System*, IBM, hierarchical data model
- 1970: E. Codd: Relational data model, relational query languages, Turing prize
- Mid 70s: First relational DBMSs (IBM System R, UC Berkeley Ingres, ...)
- 80s: Relational model de facto standard

Short History of Databases

- 1986: SQL standardized
- 90s: Object-relational databases, object-oriented databases
- Late 90s: XML databases
- 1999: SQL incorporates some OO features
- 2003, 2006: SQL incorporates support for XML data
- …

Current Database Systems

- DBMS = Database Management System
- Many vendors (Oracle, IBM DB2, MS SQL Server, MySQL, PostgreSQL, . . .)
- All rather similar
- Very big systems, but easy to use
- Common features:
  - Relational model
  - SQL as the query language
  - Server-client architecture

Transactions

- Groups of statements that need to be executed together
- Example:
  - Transferring money between accounts
  - Need to subtract amount from 1st account
  - Need to add amount to 2nd account
  - Money must not be lost!
  - Money should not be created!

ACID

Required properties for transactions

- "A" for "atomicity" – all or nothing of transactions
- "C" for "consistency" – constraints hold before and after each transaction
- "I" for "isolation" – illusion of sequential execution of each transaction
- "D" for "durability" – effect of a completed transaction may not get lost

Database Develolpment

- Requirement specification (not here)
- Data modeling
- Database modeling
- Application programming
- Database tuning

Database Course Contents

- E/R-model for data modeling
- Relational data model
- SQL language
- Application programming (JDBC)
- Basic implementation principles
- DB tuning

Note: DM 505 ≠ SQL course

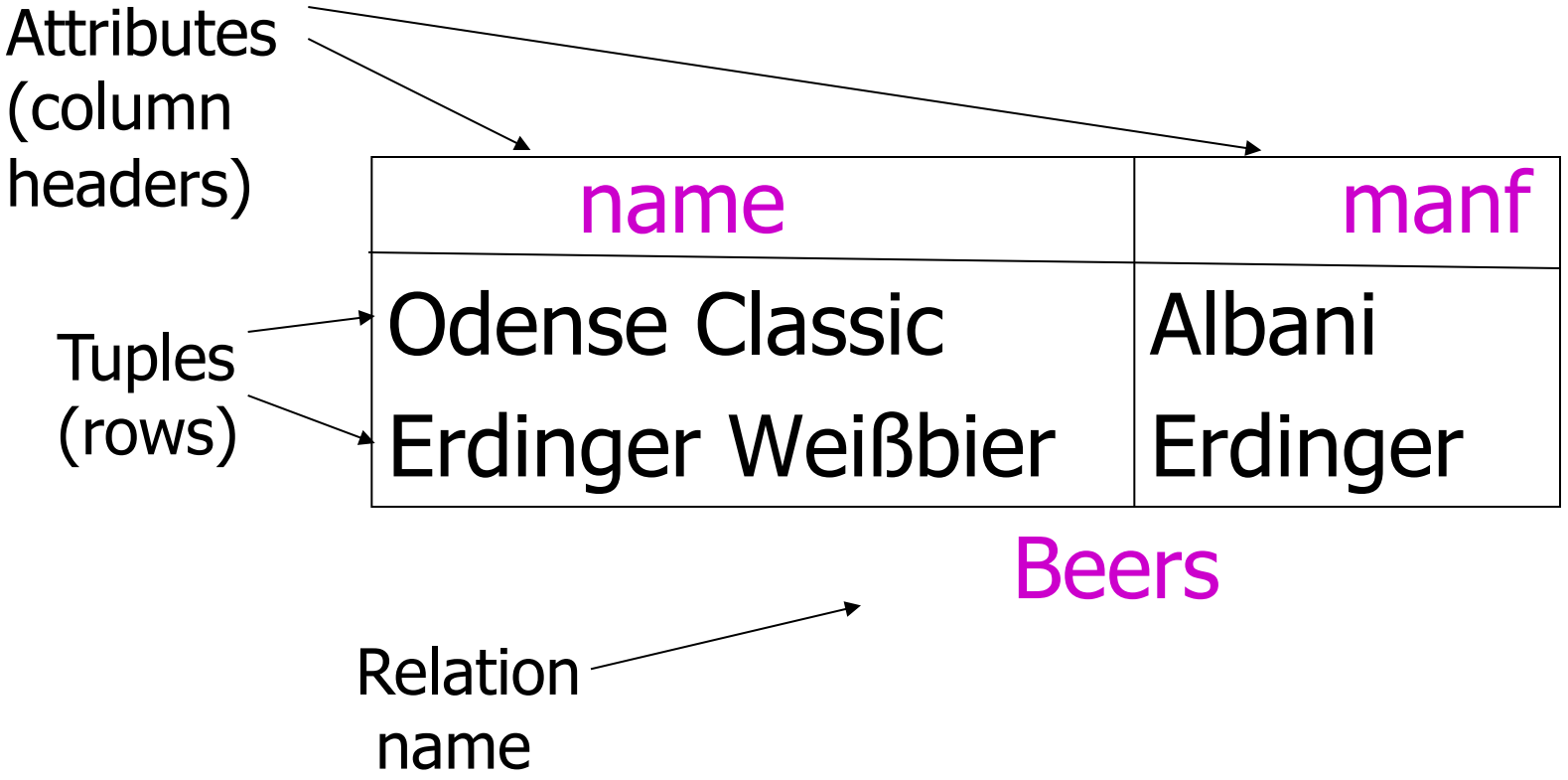
DM 505 ≠ PostgreSQL course

Data Model

What is a Data Model?

1. Mathematical representation of data
   - relational model = tables
   - semistructured model = trees/graphs
   - …
2. Operations on data
3. Constraints

A Relation is a Table



Note: Order of attributes and rows is irrelevant (sets / bags)

Schemas

- *Relation schema* = relation name and attribute list
  - Optionally: types of attributes
  - Example: Beers(name, manf) or Beers(name: string, manf: string)
- *Database* = collection of relations
- *Database schema* = set of all relation schemas in the database

Why Relations?

- Very simple model
- *Often* matches how we think about data
- Abstract model that underlies SQL, the most important database language today

Our Running Example

Beers(name, manf)

Bars(name, addr, license)

Drinkers(name, addr, phone)

Likes(drinker, beer)

Sells(bar, beer, price)

Frequents(drinker, bar)

- Underline = *key* (tuples cannot have the same value in all key attributes)
  - Excellent example of a constraint

Database Schemas in SQL

- SQL is primarily a query language, for getting information from a database
- But SQL also includes a *data-definition* component for describing database schemas

Creating (Declaring) a Relation

- Simplest form is:

```
CREATE TABLE <name> (
        <list of elements>
);
```

- To delete a relation:

```
DROP TABLE <name>;
```

Elements of Table Declarations

- Most basic element:
  an attribute and its type
- The most common types are:
  - INT or INTEGER (synonyms)
  - REAL or FLOAT (synonyms)
  - CHAR(*n*) = fixed-length string of *n* characters
  - VARCHAR(*n*) = variable-length string of up to *n* characters

Example: Create Table

```
CREATE TABLE Sells (
    bar CHAR(20),
    beer    VARCHAR(20),
    price   REAL
);
```

SQL Values

- Integers and reals are represented as you would expect
- Strings are too, except they require single quotes
  - Two single quotes = real quote, e.g., `'Trader Joe''s Hofbrau Bock'`
- Any value can be NULL
  - (like Objects in Java)

Dates and Times

- DATE and TIME are types in SQL
- The form of a date value is:

  DATE 'yyyy-mm-dd'

  - Example: `DATE '2009-02-04'` for February 4, 2009

Times as Values

- The form of a time value is:

  TIME ’hh:mm:ss’

  with an optional decimal point and fractions of a second following
  - Example: `TIME ’15:30:02.5’` = two and a half seconds after 15:30

Declaring Keys

- An attribute or list of attributes may be declared PRIMARY KEY or UNIQUE
- Either says that no two tuples of the relation may agree in all the attribute(s) on the list
- There are a few distinctions to be mentioned later

Declaring Single-Attribute Keys

- Place PRIMARY KEY or UNIQUE after the type in the declaration of the attribute
- Example:

```
CREATE TABLE Beers (
    name    CHAR(20) UNIQUE,
    manf    CHAR(20)
);
```

Declaring Multiattribute Keys

- A key declaration can also be another element in the list of elements of a CREATE TABLE statement
- This form is essential if the key consists of more than one attribute
  - May be used even for one-attribute keys

Example: Multiattribute Key

- The bar and beer together are the key for Sells:

```
CREATE TABLE Sells (
    bar     CHAR(20),
    beerVARCHAR(20),
    price   REAL,
    PRIMARY KEY (bar, beer)
);
```

PRIMARY KEY vs. UNIQUE

1. There can be only one PRIMARY KEY for a relation, but several UNIQUE attributes
2. No attribute of a PRIMARY KEY can ever be NULL in any tuple. But attributes declared UNIQUE may have NULL's, and there may be several tuples with NULL