DM505 Database Design and Programming
DM576 Database Systems

Panagiotis Tampakis

ptampakis@imada.sdu.dk

Course Organisation

- Evaluation (7 scale)
  - 24h take-home exam (everyone)
  - Project (DM576)
- Schedule
  - Find schedule on itslearning: https://sdu.itslearning.com/
- Course material
  - Course Book
  - Slides and other material on itslearning

Course Organisation

- Lectures Outline
  - Introduction
  - Relational Algebra
  - Introduction to basic SQL
  - Extended Relational Algebra and SQL
  - Functional dependencies and normal forms
  - Entity-relationship (E/R) diagrams
  - Constraints, triggers, assertions
  - Transactions, stored procedures, JDBC, psycopg2

Course Organisation

- Lectures Outline
  - Storage and Indexing (only DM576)
  - Introduction to NoSQL database models and databases such as key-value stores (only DM576)

Course Organisation

- 4 groups and 1 teaching assistant
  - Bonnie Liefting

Course Organisation

- Course Book

  Garcia-Molina, Ullman, Widom: *Database Systems: The Complete Book*. Prentice Hall, 2nd edition, 2008.

- Available as a hardcopy:

  Garcia-Molina, Ullman, Widom: *Database Systems: Pearson New International Edition: The Complete Book*. Pearson, 2013.

- Available as an e-book (google "database system the complete book")

And here we go…

- We offer you the following:
  1. I explain all needed concepts (as often as needed)
  2. We try to be available and always willing to help you
  3. We guide your learning by assigning exercises

And here we go…

- From you I/we expect the following:
  1. You ask questions, when something is unclear
  2. You contact a TA (or a classmate or me), when you need help
  3. *You practice early and often!*

Databases

What is a Database?

A Database is:

- A collection of information that exist over a long period of time
  - This implies some persistent storage

What is a Database?

- The DBMS is expected to:
  - Allow users create a DB
  - Enable users to query/modify the DB
  - Support storage of large volumes of data
    - Allow efficient retrieval of data and data modifications
  - Be durable
  - Support concurrency
  - Be consistent

Question

A straight-forward way to implement a database would be to store data in files and use a traditional programming language (e.g. Java) to maintain it (i.e. store, retrieve, durability, concurrency etc.)

- Discuss in groups and come up with a list of shortcomings with this approach.

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, Bing, ...)
  - Social networks (Facebook, Instagram, ...)
  - Blogs, discussion forums
  - ...
- Mobile apps
- Integration (data warehouses, data hubs)
- ...

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 opportunities for 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
- 2010s: increased focus on NOSQL DBs
- …

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

- Logical blocks/groups of statements that must 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 Development

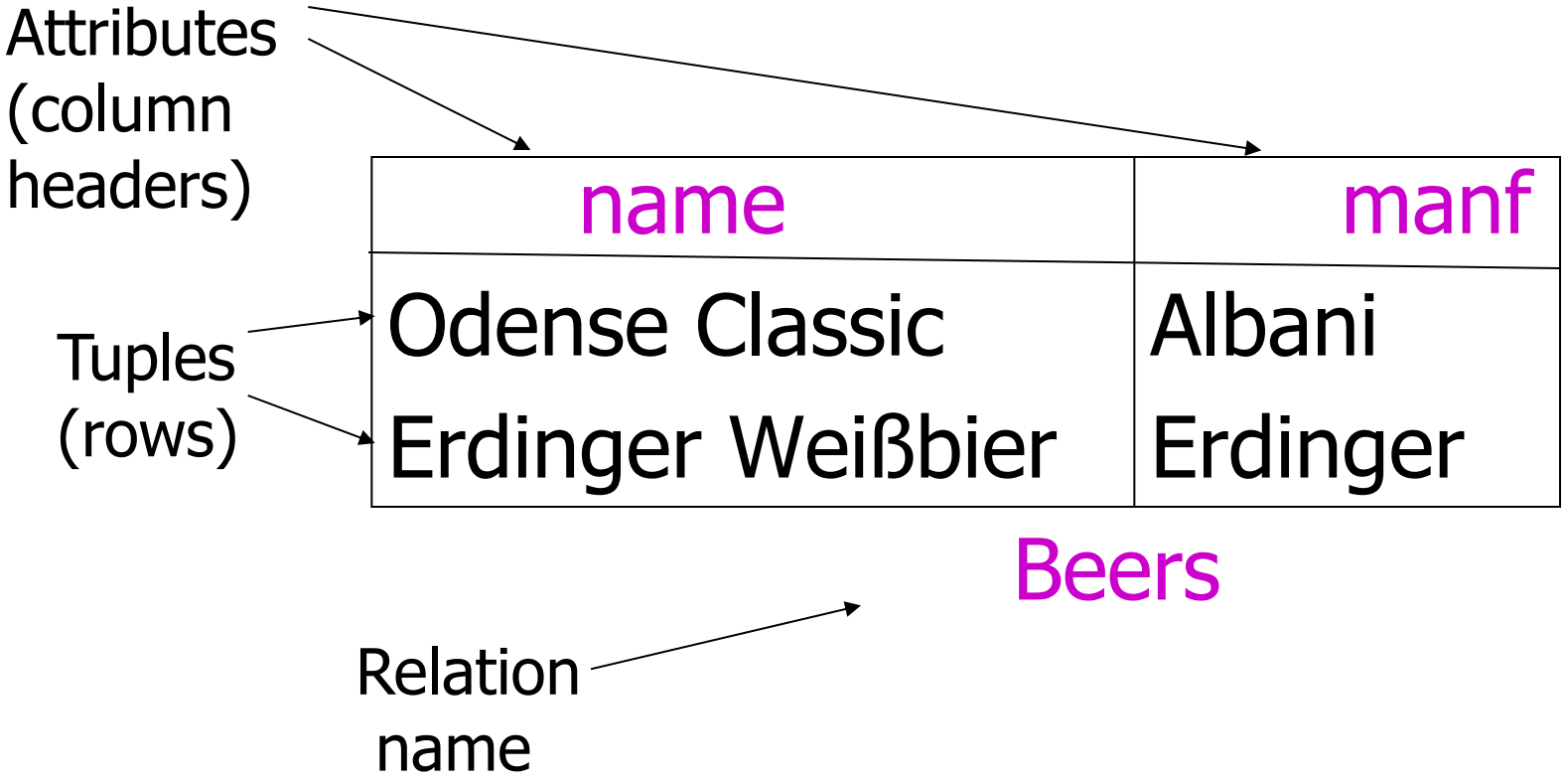
- Requirement specification (not here)
- Conceptual modeling
  - E/R-model for data modeling
- Logical modeling
  - Relational data model
- Query Language
  - SQL
- Physical Design (only DM576)
  - Storage and Indexing

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 null in Java and None in Python)

Dates and Times

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

  DATE ’yyyy-mm-dd’
  - Example: `DATE ’2021-02-01’` for 1 February 2020

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 ’10:48:02.5’` = two and a half seconds after 10:48

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),
    beer    VARCHAR(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

Question

- Course(CourseID, Name, Semester)
- Student(StudentID, Name, Address)
- StudentCourse(StudentID, CourseID, Year)

Changing a Relation Schema

- To delete an attribute:

  ALTER TABLE <name> DROP <attribute>;

- To add an attribute:

  ALTER TABLE <name> ADD <element>;

- Examples:

  ALTER TABLE Beers ADD prize CHAR(10);

  ALTER TABLE Drinkers DROP phone;

Summary 1

Things you should know now:

- Basic ideas about databases and DBMSs
- What is a data model?
- Idea and Details of the relational model
- SQL as a data definition language

Things given as background:

- History of database systems