Relational Database Systems I

Wolf-Tilo Balke
Hermann Kroll

Institut für Informationssysteme
Technische Universität Braunschweig
www.ifis.cs.tu-bs.de
0. Organizational Issues

• Who is who?
  – Wolf-Tilo Balke
    • lecture, exams
  – Hermann Kroll
    • detours
  – Janus Wawrzinek
    • detours
  – Stephan Mennicke
    • tutorials
  – Regine Dalkiran
    • office

• In case of questions, feel free to ask us.
0. Organizational Issues

• Lecture
  – October 18, 2018 to January 31, 2019
  – 15:00 – 17:15 (including a break)
  – *integrated* lecture
    (theory and detours)
  – 5 credits
  – Written exam on March 8, 2019

• Homework
  – weekly assignments
    • … can be downloaded from our website / StudIP
    • … must (!) be completed in groups of two students
• **Tutorial groups**
  – led by our HiWis
  – homework discussion

• **In order to pass this module you need to**
  1) … achieve **50%** of homework points
      (Studienleistung, ungraded 1 CP)
  2) … pass the exam (Prüfungsleistung, graded 4 CP)
0 Homework

• **Weekly** homework assignments
  – can be **downloaded** from our **website**

• Homework has to be completed within groups of two students **(no larger groups, please!)**

• To be handed in **before the next lecture**
  – drop your homework into the mailbox at our institute (**Informatikzentrum, 2nd floor**)
  – or just give it to us right before the next lecture
  – no email submissions!
  – Mark each sheet of paper with
    • your **names** and **matriculation numbers**
    • And your **tutorial group number**
  – If you have multiple pages, **staple** them together
0 Exercises - Example

Name #1
Name #2
MatNr #1
MatNr #2
GroupNo (assigned to you via registration system)
What is this thing? (RDB1, 3rd Exercise)

UBUNGEN RDB 1 - B
1 - Inheritance: Inheritance is the concept of

Mailbox on IfIS floor (opposite of elevator)

Answers

Mailbox on IfIS floor (opposite of elevator)
Of course, you can discuss the homework assignment with other people, but do not copy it.

- Always answer in your own words
  - Also copying answers from the slides does not count

Homework is graded and corrected/commented by our hiwis and returned to you in your tutorial group.

- for any questions regarding the grading, contact your HiWi directly
0 Tutorial Groups

• The tutorial groups start in two weeks
  – but: **Registration is required!**
  – registration is done via StudIP
    • Subscribe yourself with your partner to a group
    • Subscribing will be possible starting at **8 pm today (18.10)**
  – registration possible until **October 22 (next Monday)**

• **Fixed pairs of two students**
  – no more, no less…
  – you may choose a preferred partner
    • if you do not, you get a random partner
0 Tutorial Groups

Starting at 8pm 18.10!

1. Dozentinnen
   - Nachname, Vorname
   - Balke, Wolf-Tilo, Prof. Dr.

2. Mitwirkende
   - Nachname, Vorname
   - Mennicke, Stephan, M. Sc.

3. Teilnehmende nach Gruppen
   - Gruppe 1 Montag 09:45 1 (0/2)
   - Gruppe 1 Montag 09:45 2 (0/2)
   - Gruppe 1 Montag 09:45 3 (0/2)
0. Why should you be here?

- It's mandatory in your course of study….
- Database systems are an integral part of most businesses, workflows, and software products.

- There is an abundance of jobs for people with good database skills.
  - Help yourself to put you into a good position within the job market.
  - Prepare for a sunny and wealthy future!
0. Why should you be here?

Job descriptions also exactly describe this course...
After successfully completing this course students should be able to

– explain the fundamental terms of

• databases in general
• the relational model
• theoretical and practical aspects of query languages
• conceptual and logical design of databases including normalization
• application programming
• further concepts like constraints, views, indexes, transactions and object databases
0. Instructional Objectives

• They should furthermore be able to
  – design and implement a database for any specified domain using **ER-Diagrams** or **UML-Diagrams**, the Relational Model and SQL-DDL
  – **normalize** a given relational database schema
  – **enhance** the database with views, indexes, constraints, and triggers
  – formulate data retrieval **queries** in **SQL**, Relational **Algebra**, and Relational **Calculi**
  – write programs **accessing databases** using JDBC
0. Courses at ifis

• Basic course in databases
  – Relational Databases I (Bachelor)
    • What can we do with an DBMS?
    • Conceptual modeling, data retrieval, relational model, SQL, building applications, basic data models
  – Relational Databases II (Master)
    • How can we implement a DBMS?
    • Storage models, query optimization, transactions, concurrency control, recovery, data security
## 0. Contents this Course

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Data Modeling 1</td>
</tr>
<tr>
<td>3</td>
<td>Data Modeling 2</td>
</tr>
<tr>
<td>4</td>
<td>View Integration</td>
</tr>
<tr>
<td>5</td>
<td>Relational Model</td>
</tr>
<tr>
<td>6</td>
<td>Relational Algebra</td>
</tr>
<tr>
<td>7</td>
<td>Relational Calculus</td>
</tr>
<tr>
<td>8</td>
<td>SQL 1</td>
</tr>
<tr>
<td>9</td>
<td>SQL 2</td>
</tr>
<tr>
<td>10</td>
<td>Normalization</td>
</tr>
<tr>
<td>11</td>
<td>Application Programming 1</td>
</tr>
<tr>
<td>12</td>
<td>Application Programming 2</td>
</tr>
<tr>
<td>13</td>
<td>Object Persistence</td>
</tr>
<tr>
<td>14</td>
<td>Active Databases</td>
</tr>
</tbody>
</table>
0. Courses at ifis

• Advanced courses in Information Systems (Master)
  – Information Retrieval and Web Search Engines
  – Multimedia Databases
  – Distributed Data Management
  – Knowledge-Based Systems and Deductive Databases
  – Data Warehousing and Data Mining Techniques

If you still don’t have enough!
0. Recommended Literature

- **Fundamentals of Database Systems (EN)**
  - Elmasri and Navathe
  - Addison-Wesley

- **Database System Concepts (SKS)**
  - Silberschatz, Korth, and Sudarshan
  - McGraw Hill

- **Database Systems (GUW)**
  - Garcia-Molina, Ullman, and Widom
  - Prentice Hall

- **Datenbanksysteme (KE)**
  - Kemper, and Eickler
  - Oldenbourg
0. Recommended Literature

- **Database Modeling and Design: Logical Design**
  - Teorey, Lightstone, and Nadeau
  - Morgan Kaufmann

- **SQL Cookbook**
  - Molinaro
  - O’Reilly

- **Using the New DB2**
  - Chamberlin
  - AP Professional

- **W3Schools SQL**
  - [http://www.w3schools.com/sql/](http://www.w3schools.com/sql/)
1 Introduction

• What is a Database?
• Characteristics of a Database
• History of Databases
1.1 What is a Database?

- Managing large amounts of data is an integral part of most nowadays business and governmental activities
  - collecting taxes
  - bank account management
  - bookkeeping
  - airline reservations
  - human resource management
  - …
1.1 What is a Database?

- **Databases** are needed to manage that vast amount of data
- A database (DB) is a collection of related data
  - represents some aspects of the real world
    - universe of discourse
  - data is logically **coherent**
  - is provided for an intended group of **users** and **applications**
As for today, the database industry is one of the most successful branches of computer science:

- constantly growing since the 1960s
- more than $8 billion revenue per year
- DB systems found in nearly any application
- ranging from large commercial transaction-processing systems to small open-source systems for your Web site
1.1 What is a Database?

• Databases are maintained by using a collection of programs called a database management system (DBMS), that deals with
  – definition of data and structure
  – physical construction
  – manipulation
  – sharing/protecting
  – persistence/recovery
1.1 File Systems

• A file system is not a database!
• File management systems are **physical** interfaces
1.1 File Systems

**Advantages**
- fast and easy access

**Disadvantages**
- uncontrolled redundancy
- manual maintenance of consistency
- limited data sharing and access rights
- poor enforcement of standards
- excessive data and access paths maintenance
Databases are logical interfaces
– retrieval of data using data semantics
– controlled redundancy
– data consistency & integrity constraints
– effective and secure data sharing
– backup and recovery

However…
– more complex
– more expensive data access
1.1 Databases

• **DBMS** replaced previously dominant file-based systems in **banking** due to special requirements
  – **simultaneous** and quick access is necessary
  – failures and loss of data **cannot** be tolerated
  – data always has to remain in a **consistent** state
  – frequent queries and modifications
1 Introduction

• What is a Database?
• Characteristics of a Database
• History of Databases
• Organizational Issues
1.2 Characteristics of DBs

• Databases control redundancy
  – same data used by different applications or tasks is stored only once
  – access via a single interface provided by DBMS
  – redundancy only purposefully used to speed up data access (e.g. materialized views)

• Problems of uncontrolled redundancy
  – updating data may result in inconsistent data
1.2 Characteristics of DBs

- Databases are well-structured (e.g. ER model)
  - simple banking system

- Relational Databases provide
  - catalog (data dictionary) contains all meta data
  - defines the structure of the data in the database
1.2 Characteristics of DBs

• Databases support **declarative querying**
  – just specify what you want, not how and from where to get it
  – queries are separated and abstracted from the actual physical organization and storage of data

• Get the firstname of all customers with lastname *Smith*
  – file system: trouble with physical organization of data
    • Load file c:\datasets\customerData.csv.
    • Build a regular expression and iterate over lines: If 2\textsuperscript{nd} word in line equals *Smith*, then return 1\textsuperscript{st} word.
    • Stop when end-of-file marker is reached.
  – database system: simply query
    • SELECT firstname FROM data WHERE lastname='Smith'
1.2 Characteristics of DBs

- Databases aim at **efficient** manipulation of data
  - physical tuning allows for good data allocation
  - indexes speed up search and access
  - query plans are optimized to improve performance

- Example: Simple Index

<table>
<thead>
<tr>
<th>Index File (checking accounts)</th>
<th>Data File</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>number</strong></td>
<td><strong>type</strong></td>
</tr>
<tr>
<td>4543032</td>
<td>saving</td>
</tr>
<tr>
<td>7809849</td>
<td>saving</td>
</tr>
<tr>
<td>4543032</td>
<td>checking</td>
</tr>
<tr>
<td>5539783</td>
<td>saving</td>
</tr>
<tr>
<td>7809849</td>
<td>checking</td>
</tr>
<tr>
<td>8942214</td>
<td>checking</td>
</tr>
<tr>
<td>9134354</td>
<td>saving</td>
</tr>
<tr>
<td>9543252</td>
<td>saving</td>
</tr>
</tbody>
</table>
1.2 Characteristics of DBs

- **Isolation** between applications and data
  - database employs **data abstraction** by providing **data models**
  - applications work only on the **conceptual representation** of data
    - Data is strictly typed (Integer, Float, Timestamp, Varchar, …)
    - Details on where data is actually stored and how it is accessed are **hidden** by the DBMS
    - Applications can access and manipulate data by invoking **abstract operations** (e.g. SQL statements)
  - DBMS-controlled parts of the file system are **protected** against external manipulations (tablespaces)
1.2 Characteristics of DBs

**Example:** Schema can be changed and tablespace moved without adjusting the Application’s SELECT

```sql
SELECT number FROM account WHERE balance > 0
```
1.2 Characteristics of DBs

- **Example:** Schema can be changed and tablespace moved without adjusting the Application’s SELECT

```sql
SELECT number FROM account WHERE balance>0
```

![Diagram showing schema changes](image-url)
1.2 Characteristics of DBs

• Supports multiple **views** of the data
  – views provide a different perspective of the DB
    • a user’s conceptual understanding or task-based excerpt of the data (e.g. aggregations)
    • security considerations and access control (e.g. projections)
  – for applications, a view does not differ from a table
  – views may contain **subsets** of a DB and/or contain **virtual data**
    • virtual data is **derived** from the DB (mostly by simple SQL statements, e.g. joins over several tables)
    • can either be computed at query time or **materialized** upfront
1.2 Characteristics of DBs

- Example views: **Projection**
  - *saving* account clerk vs. *checking* account clerk

**Original Table**

<table>
<thead>
<tr>
<th>number</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1278945</td>
<td>saving</td>
<td>€ 312.10</td>
</tr>
<tr>
<td>2437954</td>
<td>saving</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td>4543032</td>
<td>checking</td>
<td>€ -43.03</td>
</tr>
<tr>
<td>5539783</td>
<td>saving</td>
<td>€ 12.54</td>
</tr>
<tr>
<td>7809849</td>
<td>checking</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td>8942214</td>
<td>checking</td>
<td>€ -345.17</td>
</tr>
<tr>
<td>9134354</td>
<td>saving</td>
<td>€ 2.22</td>
</tr>
<tr>
<td>9543252</td>
<td>saving</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>

**Saving View**

<table>
<thead>
<tr>
<th>number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1278945</td>
<td>€ 312.10</td>
</tr>
<tr>
<td>2437954</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td>5539783</td>
<td>€ 12.54</td>
</tr>
<tr>
<td>9134354</td>
<td>€ 2.22</td>
</tr>
<tr>
<td>9543252</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>

**Checking View**

<table>
<thead>
<tr>
<th>number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>4543032</td>
<td>€ -43.03</td>
</tr>
<tr>
<td>7809849</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td>8942214</td>
<td>€ -345.17</td>
</tr>
</tbody>
</table>
• **Sharing** of data and support for atomic multi-user transactions

  – transactions are a **series of database operations** executed as **one logical operation**

  – **concurrency control** is necessary for maintaining consistency
    • multiple users and applications may access the DB at the same time

  – transactions need to be **atomic** and **isolated** from each other
1.2 Characteristics of DBs

• Example: Atomic transactions

  – Program:
    Transfer $x$ Euros from Account 1 to Account 2
    1. Debit amount $x$ from Account 1
    2. Credit amount $x$ to Account 2
1.2 Characteristics of DBs

• **Example:** Atomic transactions

  – **Program:**
    Transfer \( x \) Euros from Account 1 to Account 2
    1. Debit amount \( x \) from Account 1
    2. Credit amount \( x \) to Account 2

  – **But what happens if the system fails after performing the first step?**
1.2 Characteristics of DBs

- **Example:** multi-user transactions
  - **Program:**
    Withdraw $x$ Euros from Account 1
    1. Read *old balance* from DB
    2. Set *new balance* to *old balance* – $x$
    3. Write *new balance* back to the DB
  - **Problem:** Dirty Read
    - Account 1 has €500
    - User 1 deduces €20
    - User 2 deduces €80 at the same time
  - Without multi-user transactions, Account 1 will have either €480 or €420, but not the correct €400
    - Both users read old value of €500 simultaneously, both deduce either €20 or €80, both write back new value (in random order)
1.2 Characteristics of DBs

- **Persistence** of data and disaster recovery
  - data needs to be persistent and accessible at all times
  - **quick** recovery from system crashes **without data loss**
  - recovery from natural disasters (fire, earthquake, …)
1.2 Database Users

• Usually **several groups of persons** are involved in the daily usage of a large DBMS (many job opportunities for smart DB people…)

• Persons directly involved on DB level
  – **Database Administrators**
    • responsible for tuning and maintaining the DBMS
    • management of storage space, security, hardware, software, etc.
  – **Database Designers**
    • identify the data that needs to be stored and chooses appropriate data structures and representations
    • integrate the needs of all users into the design
1.2 Database Users

- **Application Developers**
  - identify the requirements of the end-users
  - develop the software that is used by (naïve) end-users to interact with the DB
  - cooperate closely with DB designers

- **Data Analyst**
  - Analyzes trends in data to uncover valuable feedback for business operations
  - i.e., discover sales trends, upcoming supply shortages, etc.

- **Persons working behind the scenes**
  - **DBMS Designers and Implementers**
    - implement the DBMS software
  - **Tool Developers**
    - develop generic tools that extend the DBMS’ functionalities
  - **Operators and maintenance personnel**
    - responsible for actually running and maintaining the DBMS hardware
1.2 Database Users

• Persons using the database (End Users)
  – All people who use the DB to do their job

• End Users can be split into
  – Naïve End Users
    • make up most DB users
    • usually repeat similar tasks over and over
    • are supported by predesigned interfaces for their tasks
    • e.g. bank tellers, reservation clerks, …
### 1.2 Database Users

**– Sophisticated End Users**
- require *complex* non-standard operations and views from the DB
- are familiar with the facilities of the DBMS
- can solve their problems themselves, but require complex tools
- e.g. engineers, scientists, business analysts, …

**– Casual End Users**
- use DB only from time to time, but need to perform different tasks
- are familiar with query languages
- e.g. people in middle or senior management
I Introduction

• What is a Database?
• Characteristics of a Database
• History of Databases
Databases have an exceptional history of development

- many synergies between academic, governmental and industrial research
- much to be learned from it
- most popular concepts used today have been invented decades ago
1.3 History of DBs

• The beginnings
  – 1880: U.S. Bureau of Census instructs Herman Hollerith to develop a machine for storing census data
  – result: **Punch card** tabulating machine
    • the evaluation of 1880’s census took 8 years
    • 1890’s has been finished after only 6 years
  – leads to the foundation of **IBM**
    • International Business Machines
  – data processing machines soon established in accounting
1.3 History of DBs

• One of Hollerith’s punch cards:
1.3 History of DBs

- Tabulating machines
  - operations or “programs” directed by a plug board
  - up to 150 cards per minute
  - results were printed or punched for input to other processing steps
1.3 History of DBs

• In 1951 IBM develops the electric **UNIVAC 1**
  – first commercial computer produced in the USA
    • programmable (turing complete)
    • input (programs and data) with tape or punched cards

• In 1959, USA dominated the (still highly active) punch card machine market
  – within the USA, the Pentagon alone used more than 200 data processing computers, costing $70 million per year
In 1964, the term *data base* appeared for the first time in military computing using *time sharing systems*.

- Data could be shared among users.
- But data was still bound to one specific application.
  - Similar data needed by multiple applications had to be duplicated.
  - Consistency problems when updating data.

- Data structure highly-dependent on the hardware and (low-level) programming language used.
  - Inspired by punch cards and optimized for magnetic tapes.
  - Usually, no *relationships* between different records have been stored, just plain data.
To turn stored data into a proper **database**, the following goals had to be achieved (McGee, 1981):

- **Data Consolidation**
  - data must be stored in a central place, accessible to all applications
  - knowledge about relationships between records must be represented

- **Data Independence**
  - data must be independent of the specific quirks of the particular low level programming language used
  - provide high-level interfaces to physical data storage

- **Data Protection**
  - data must be protected against loss and abuse
1.3 History of DBs

• **Data Consolidation** motivated the development of data models
  – Hierarchical Data Model
  – Network Data Model
  – **Relational Data Model**
  – Object-oriented Data Model
  – Semantic Data Model

• **Data Independence** inspired the development of query models and high-level languages
  – **Relational Algebra, SQL**

• **Data Protection** led to development of transactions, backup schemes, and security protocols
1.3 History of DBs

• Hierarchical Data Model
  – first appearance in IBM’s IMS database system, designed for the Apollo Program in 1966
    • still, as of 2006, 95% of all Fortune 1000 companies use IBM IMS in their data backbone…
  – benefits from advances in hardware design
    • random access main memory and tape media available
Hierarchical data model
- each type of record has some defined structured data
- hierarchical **one-to-many** relationships

**Advantages**
- 1:n relationships can be expressed
- can easily be stored on tape media

**Disadvantages**
- no n:m relationships
- no Data Independence
1.3 History of DBs

- **Network Data Model**
  - in the mid-1960th, direct access storage devices (DASD) gained momentum
    - primarily hard disks
    - more complex storage schemes possible
  - Hierarchical Data Model failed, e.g. for bill-of-material-processing (BOMP)
    - many-to-many relationships needed
    - development of the IBM DBOMP system (1960)
  - result: Network Data Model
    - two types of files: master files, chain files
    - chain file entries could chain master file entry to one another
1.3 History of DBs

- **Network Data Model**
  - The model was standardized by Charles W. Bachman for the **CODASYL** Consortium in 1969
    - CODASYL = Conference of Data Systems Languages
    - Thus, also called the CODASYL model
  - Allowed for more natural modeling of **associations**

- **Advantages**
  - Many-to-many-relationships

- **Disadvantages**
  - No declarative queries
  - Queries must state the data access path
The relational data model
Published by Edgar F. “Ted” Codd in 1970, after several years of work
– A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, 1970
– employee of IBM Research
  • IBM ignored his idea for a long time as not being “practical” while pushing its hierarchical IMS database system
  • other researchers in the field also rejected his theories
  • finally, he received the Turing Award in 1981
1.3 History of DBs

• **Idea underlying the relational model:**
  – database is seen as a collection of **predicates** over a finite set of **predicate variables**
    • example
      – dislikes(x, y)
      – dislikes(‘Ted Codd’, ‘hierarchical IMS database system’) (TRUE)
      – dislikes(‘IBM’, ‘hierarchical IMS database system’) (FALSE)
    • the set of all true assignments is called a **relation**
    • relations are stored in **tables**
  – contents of the DB are a **collection of relations**
  – queries are also **predicates**
    • queries and data are very similar
    • Allows for **declarative querying**
• It’s really like a collection of index cards – more details during the next weeks…

R

<table>
<thead>
<tr>
<th>A₁</th>
<th>...</th>
<th>Aₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relation variable (Table name)

Attribute (Column) {unordered}

Heading

Body

Relation (Table)

Tuple (Row) {unordered}
Beginning 1977, Lawrence J. Ellison picked up the idea and created Oracle DB (currently in version 12c) – and became insanely rich – long time in the Top 10 of the richest people – in 2015 Oracle ranked second on the list of largest software companies in the world, right after Microsoft.
1.3 History of DBs

• Oracle also sells a suite of business applications
  – Oracle eBusiness Suite
  – includes software to perform financial- and manufacturing-related operations, customer relationship management, enterprise resource planning, and human resource management

• Basically gained from high-value acquisitions
  – JD Edwards, PeopleSoft, Siebel Systems, BEA, Sun Microsystems (Java), …
During the 1970s, IBM had also decided to develop a relational database system

- **System R** with the first implementation of the **SQL** declarative query language (SEQUEL)
- at first, mostly a research prototype, later became the base for **IBM DB2**

Ingres Database was developed at UC Berekeley as a research project

- It became PostgreSQL in 1996
### 1.3 History of DBs

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1887</td>
<td>Hollerith census machine</td>
</tr>
<tr>
<td>1951</td>
<td>Univac 1 electrical data machine</td>
</tr>
<tr>
<td>1959</td>
<td>First CODASYL Conference</td>
</tr>
<tr>
<td>1960</td>
<td>Flight reservation system SABRE</td>
</tr>
<tr>
<td>1966</td>
<td>IMS hierarchical database</td>
</tr>
<tr>
<td>1969</td>
<td>Network model</td>
</tr>
<tr>
<td>1971</td>
<td>CODASYL Recommendation for DDL and 3-Layer-Architecture</td>
</tr>
<tr>
<td>1975</td>
<td>System R introduces SEQUEL query language</td>
</tr>
<tr>
<td>1976</td>
<td>System R introduces transaction concepts</td>
</tr>
<tr>
<td>1976</td>
<td>Peter Chen proposes entity relationship modeling</td>
</tr>
<tr>
<td>1980</td>
<td>Oracle, Informix and others start selling DBMS with SQL support</td>
</tr>
<tr>
<td>1983</td>
<td>Work on ACID transactions published by Theo Haerder and Andreas Reuter</td>
</tr>
</tbody>
</table>
1.3 History of DBs

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1986</td>
<td>SQL standardized as SQL-1 ANSI/SQL</td>
</tr>
<tr>
<td>1987</td>
<td>SQL internationally standardized as ISO 9075</td>
</tr>
<tr>
<td>1989</td>
<td>SQL 2 standard supports referential integrity</td>
</tr>
<tr>
<td>1991</td>
<td>SQL 2 supports domains and key definitions</td>
</tr>
<tr>
<td>1993</td>
<td>Object-oriented data model</td>
</tr>
<tr>
<td>1995</td>
<td>Preliminary SQL 3 supporting sub-tables, recursion, procedures, and triggers</td>
</tr>
<tr>
<td>1996</td>
<td>First object-oriented databases</td>
</tr>
<tr>
<td>1999</td>
<td>First part of the SQL 3 standard finalized</td>
</tr>
<tr>
<td>2003</td>
<td>SQL 2003 finalized with support for object-relational extensions</td>
</tr>
<tr>
<td>2006</td>
<td>SQL 2006 introduces new XML capabilities</td>
</tr>
<tr>
<td>2008</td>
<td>SQL 2008 ORDER BY outside cursors, adds INSTEAD OF triggers and the TRUNCATE statement</td>
</tr>
<tr>
<td>2011</td>
<td>SQL 2011 supports temporal databases</td>
</tr>
<tr>
<td>2016</td>
<td>SQL 2016 row pattern matching, polymorphic table functions, JSON</td>
</tr>
</tbody>
</table>

... To be continued ...
Lately, the so-called NoSQL databases have become popular

- NoSQL systems are mostly non-relational database systems
  - They might support SQL as a query language
- Its development was driven by IT companies like Amazon, Google and Facebook
  - Databases have to deal with huge amounts of data and massive numbers of users
- NoSQL comprise: key-value stores, document stores, column stores, distributed databases, graph databases, triple stores
  - For details, visit our advanced lectures
I Summary

• Databases
  – are logical interfaces
  – support declarative querying
  – are well-structured
  – aim at efficient manipulation of data
  – support control redundancy
  – support multiple views of the data
  – support atomic multi-user transactions
  – support persistence and recovery of data
• Next Lecture
  – Phases of DB Design
  – Data Models
  – Basic ER Modeling
    • Chen Notation
    • Mathematical Model