Database Systems
Spring 2013

Introduction
SL01

- Organization of the course
- The database field, basic definitions
- DB applications, functionality, users and languages
- Data models, schemas, instances, and redundancy
- Main characteristics of the database approach
- History, advantages and disadvantages of database systems

Organization of the Course

- Database curricula at ifi
- Literature
- Lectures
- Exercises
- Content

About me

- I have been a database system person since 20 years.
- My previous affiliations (and the first example of a database):

  Affiliations

Start	End	Institution	Country
1990	1994	ETH Zürich	CH
1994	1995	University of Arizona	USA
1995	2003	Aalborg University	DK
2003	2009	Free University of Bozen-Bolzano	IT
2009	now	University of Zürich	CH

- For the course it is important that you are **precise** and that you can apply your knowledge on relevant **examples**.

About the Database Systems Course

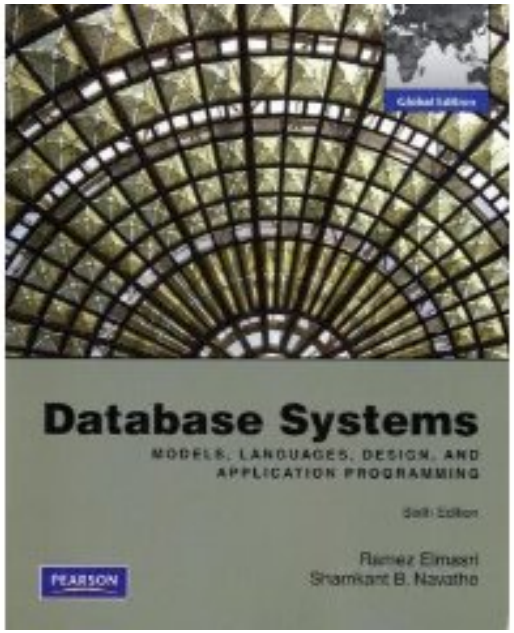
- Slides will be ready shortly before classes on the course web page:
  `http://www.ifi.uzh.ch/dbtg/teaching/courses/DBS.html`
- The textbook is Database Systems by Elmasri and Navathe. Use this book for preparation throughout the semester.
- **Doing the exercises is very important. It is the best preparation for the exam.**
- During the lecture we will solve illustrative examples on the board. Interaction during class is welcome.

- What is important
  - Being able to apply your knowledge to relevant examples.
  - Being able to be precise about the key concepts of database systems.

About Database Sytems @ifi

- Database Systems (DBS), Spring, 4th semester
- Praktikum Datenbanksysteme (PDBS), Fall, 5th semester
- Distributed Databases (DDBS), Fall, 5th semester
- XML Databases, (XMLDB), Spring, 6th ot 8th semester
- Database Systems Implementation (IDBS), Fall, 7th semester
- Seminar Database Systems (SDBS), Spring, 8th or 6th semester
- Data Warehousing, Spring (DW), Spring even years, 8th semester
- Nonstandard Databases (NDBS), Fall, 9th semester

Literature and Acknowledgments



Reading List for SL01:

- Database Systems, Chapters 1 and 2, Sixth Edition, Ramez Elmasri and Shamkant B. Navathe, Pearson Education, 2010.

These slides were developed by:

- Michael Böhlen, University of Zürich, Switzerland
- Johann Gamper, Free University of Bozen-Bolzano, Italy

The slides are based on the following text books and associated material:

- Database Systems, Sixth Edition, Ramez Elmasri and Shamkant B. Navathe, Pearson Education, 2010.
- A. Silberschatz, H. Korth, and S. Sudarshan: Database System Concepts, McGraw Hill, 2006.

The Course/1

- The final exam is written and takes place Tuesday, June 18, 10:15 - 12:00 (check official web pages for details).
- There is no re-exam.
- Office hours after appointment with TAs (after exercise hour or by email).
- The exercises take place Tuesday 12:00-13:45 (1 group) and Wednesday 14:00-15:45 (2 groups). Start is February 26. This week there is no exercise.
- TAs: Katerina Papaioannou (BIN-2.A.01, English), Anton Dignös (BIN-0.B.06, German), Francesco Cafagna (BIN-2.A.01, English).

The Course/2

- Please sign up for the exercise groups by the end of this week by filling the Doodle (cf. course web page). We will balance the load across groups.
- The weekly exercises are an important part of the course. The assessment consists of the completion of 9 out of 12 exercises and the participation at the final exam. Both parts have to be passed independently.
- Hand in of the exercises is Tuesday at 12:00 (in class) or before to TA directly.
- Exercises are only valid for the current year.

The Course/3

- **Exercises**

26+27.2	Databases, relational algebra
05+06.3	Tuple relational calculus, domain relational calculus
12+13.3	PostgreSQL, SQL (metadata, DDL, simple DML)
19+20.3	SQL (advanced DML)
26+27.3	Stored procedures, triggers
09+10.4	Relational database design
16+17.4	Functional dependencies, multivalued dependencies
23+24.4	Entity relationship (ER) model
30.4	From the ER model to the relational model
07+08.5	Query trees and plans
14+15.5	Cost computation, empirical performance tests
21+22.5	Transaction processing
28+29.5	-

The Course/4

- **Course Content**
  - Database systems, chapter 1 and 2
    - The field, terminology, database system, schema, instance, functionality, architecture
  - Relational model, algebra, and calculus, chapter 3 and 6
    - The relational model, relational algebra, tuple relational calculus, domain relational calculus
  - SQL, chapter 4
    - Data definition language, data manipulation language
  - Constraints, triggers, views, DB programming, chapter 5 and 12
    - column constraints, table constraints, assertions, referential integrity, triggers, stored procedures
  - Relational database design, chapter 14 and 15
    - design goals, keys, functional dependencies, normal forms, lossless join decompositions, higher normal forms

The Course/5

- **Course Content**
  - Conceptual database design, chapter 7 and 8
    - The design process, the entity-relationship model, entity-relationship to relational model mapping
  - Physical database design, chapter 16 and 17
    - Physical Storage media, file and buffer manager, indices, B-trees, hashing
  - Query processing and optimization, chapter 18 and 19
    - Measures of query cost, selection and join operation, transformation of relational expressions, evaluation plans
  - Transactions, concurrency, recovery, chapter 20, 21, 22
    - ACID properties, SQL transactions, concurrency protocols, log-based recovery

The Database Field

- Professional Resources
- Products
- Activities of Database People
- Basic Terminology and Definitions

The Field/1

- Conference Publications
  - SIGMOD/PODS
  - VLDB
  - ICDE
  - EDBT/ICDT
- Journal Publications
  - ACM Transaction on Database System (TODS)
  - The VLDB Journal (VLDBJ)
  - IEEE Transactions on Knowledge and Data Engineering (TKDE)
  - Information Systems (IS)
- DBLP Bibliography (Michael Ley, Uni Trier, Germany)
  - http://dblp.uni-trier.de/db/
- DBWorld mailing list
  - http://www.cs.wisc.edu/dbworld/

The Field/2

dblp .uni-trier.de
Computer Science Bibliography

The DBLP Computer Science Bibliography

maintained by Michael Ley - Welcome - FAQ

DBLP is available from several hosts: Trier I - Trier II - ACM SIGMOD - SunSITE CE

Search

- Author
- Faceted search (L3S Research Center, U. Hannover)
- CompleteSearch (Holger Bast, Max Planck Institut f. Inf.)

Bibliographies

- **Conferences**: SIGMOD, VLDB, PODS, ER, EDBT, ICDE, POPL, ...
- **Journals**: CACM, TODS, TOIS, TOPLAS, DKE, VLDB J., Inf. Systems, TPLP, TCS, ...
- **Series**: LNCS/LNAI, IFIP
- **Books**: Reference - Collections - DB Textbooks
- **By Subject**: Database Systems, Logic Prog., IR, ...

Full Text: ACM SIGMOD Anthology

Products

- Commercial Products
  - Oracle
  - DB2 (IBM)
  - Microsoft SQL Server
  - Teradata
  - Sybase (SAP)
  - Informix (IBM)
  - PC "DBMSs": Access (Microsoft), Paradox, ...
  - ...
- Open Source Products
  - MySQL (Oracle)
  - PostgreSQL
  - MonetDB
  - ...

We will use PostgreSQL for this course.

Oracle's Solution Stack

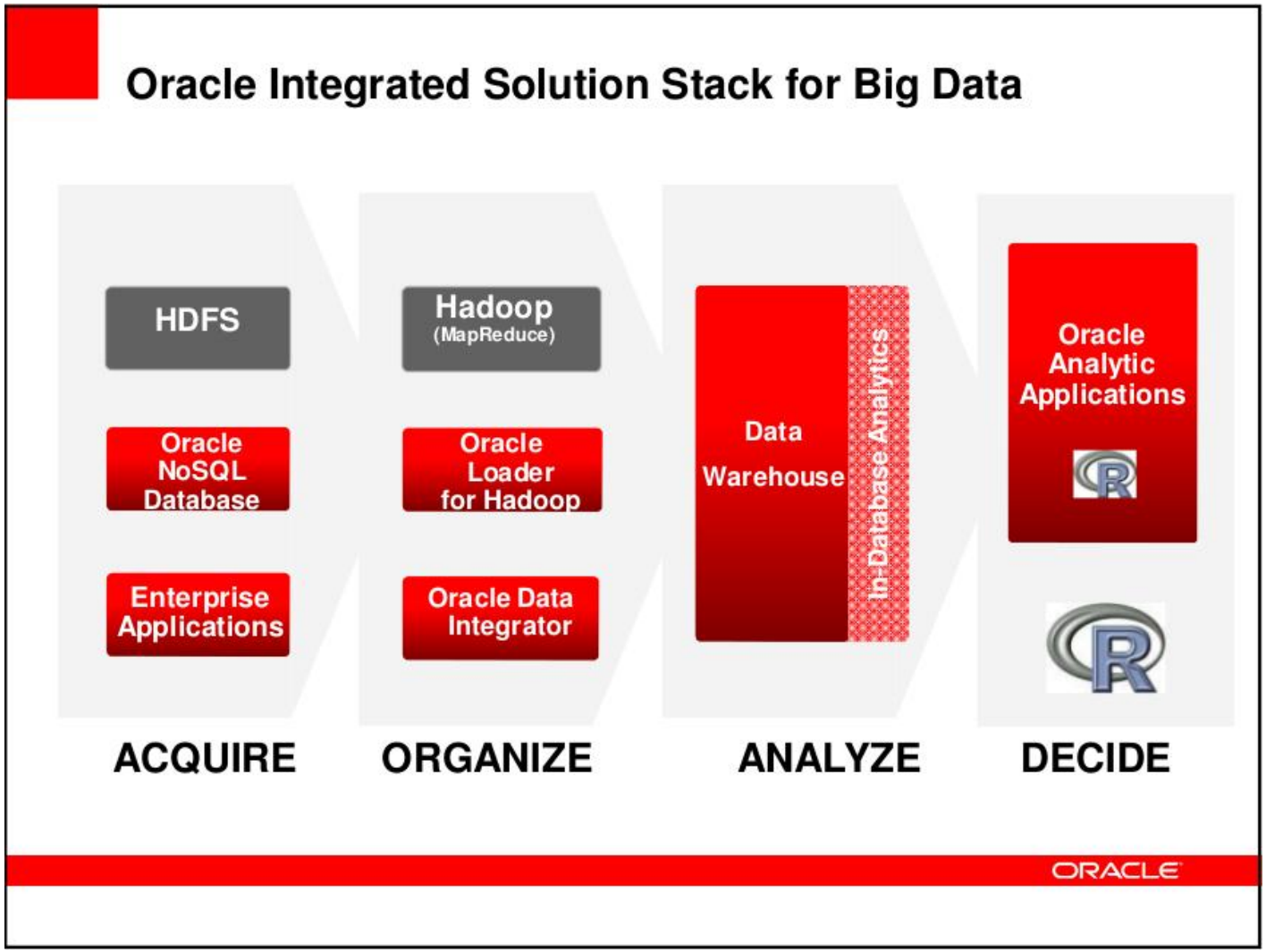


Image: Roger Wullschleger, Oracle @ DBTA Workshop on Big Data, Bern, 2012

Typical Activities/Jobs of Database People

- Data modeling
- Handling large volumes of complex data
- Distributed databases
- Design of migration strategies
- User interface design
- Development of algorithms
- Design of languages
- New data models and systems
  - XML/semi-structured databases
  - Stream data processing
  - Temporal and spatial databases
  - GIS systems
- etc.

Basic Definitions/1

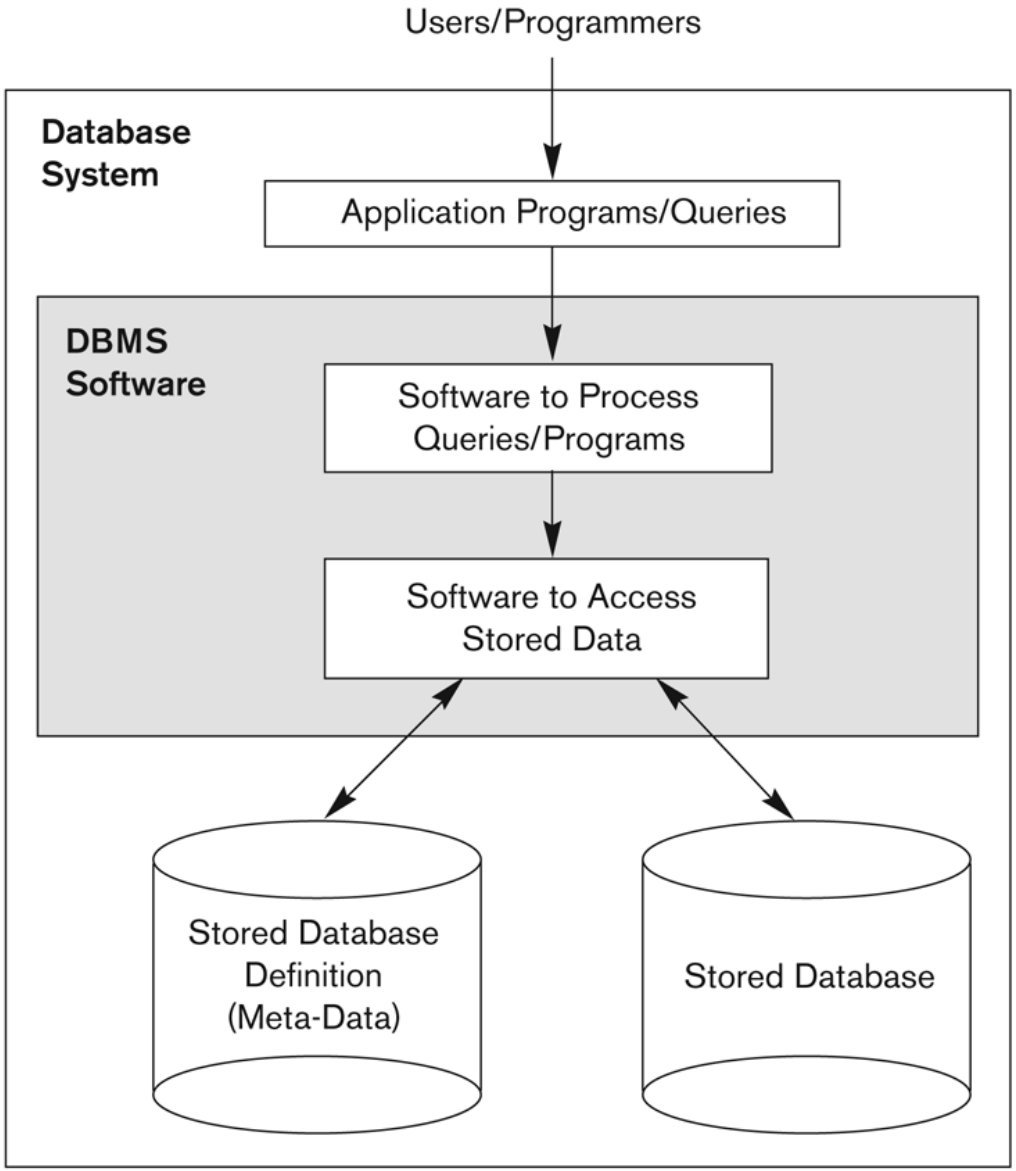
About, data, information, and knowledge:

- **Data** are facts that can be recorded:
  - book(Lord of the Rings, 3, 10)
- **Information** = data + meaning
  - book:
  - title = Lord of the rings,
  - volume nr = 3,
  - price in USD = 10
- **Knowledge** = information + application

Basic Definitions/2

- **Mini-world**: The part of the real world we are interested in
- **Data**: Known facts about the mini-world that can be recorded
- **Database (DB)**: A collection of related data
- **Database Management System (DBMS)**: A software package to facilitate the creation/maintenance/querying of databases
- **Database System (DBS)**: DB + DBMS
- **Meta Data**: Information about the structure of the DB.
  - Meta data is organized as a DB itself.

Basic Definitions/3



DBMS Languages/1

- A DBMS offers two types of languages:
  - data definition language (DDL) to create and drop tables, etc
  - data manipulation language (DML) to select, insert, delete, and update data
- The standard language for database systems is SQL
  - SQL stands for Structured Query Language
  - Example SQL query: `select * from r`
  - the original name was SEQUEL
  - "Intergalactic data speak" [Michael Stonebraker].
- SQL offers a DDL and a DML.

DBMS Languages/2

- We distinguish between
  - High level or declarative (non-procedural) languages
  - Low level or procedural languages
- High level or non-procedural language:
  - For example, the SQL relational language
  - Set-oriented (retrieve multiple results)
  - Specify **what** data to retrieve and not how to retrieve it
  - Also called declarative languages.
- Low level or procedural language:
  - Retrieve data one record at a time
  - Specify **how** to retrieve data
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

Review 1.1

1. Give examples of declarative and procedural approaches from the real world.

Applications, Functionality, Users and Interfaces

- Application Areas of Database Systems
- Functionality of Database Systems
- Users of Database Systems
- DBMS Interfaces

Applications of Database Systems

- Traditional Applications
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications
- Examples:
  - Bank (accounts)
  - Insurances
  - Stores (inventory, sales)
  - Reservation systems
  - University (students, courses, rooms)
  - online sales (amazon.com)
  - online newspapers (nzz.ch)

Functionality of Database Systems/1

Typical DBMS functionality:

- **Define** a particular database in terms of its data types, structures, and constraints
- **Construct** or **load** the initial database contents on a secondary storage medium
- **Manipulating** the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- **Sharing** by a set of concurrent users and application programs while, at the same time, keeping all data valid and consistent

Functionality of Database Systems/2

Additional DBMS functionality:

- Other features of DBMSs:
  - Protection or security measures to prevent unauthorized access
  - Active processing to take internal actions on data
  - Presentation and visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application (called database, software, and system maintenance)

Users of Database Systems/1

Database users have very different tasks. There are those who use and control the database content, and those who design, develop and maintain database applications.

- **Database administrators**:
  - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
- **Database Designers**:
  - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Users of Database Systems

- **End-users**: They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
  - **Casual**: access database occasionally when needed
  - **Naïve**: they make up a large section of the end-user population.
    - They use previously well-defined functions in the form of "canned transactions" against the database.
    - Examples are bank-tellers or reservation clerks.
  - **Sophisticated**:
    - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
    - Many use tools in the form of software packages that work closely with the stored database.
  - **Stand-alone**:
    - Mostly maintain personal databases using ready-to-use packaged applications.
    - An example is a tax program user that creates its own internal database or a user that maintains an address book

DBMS Interfaces/1

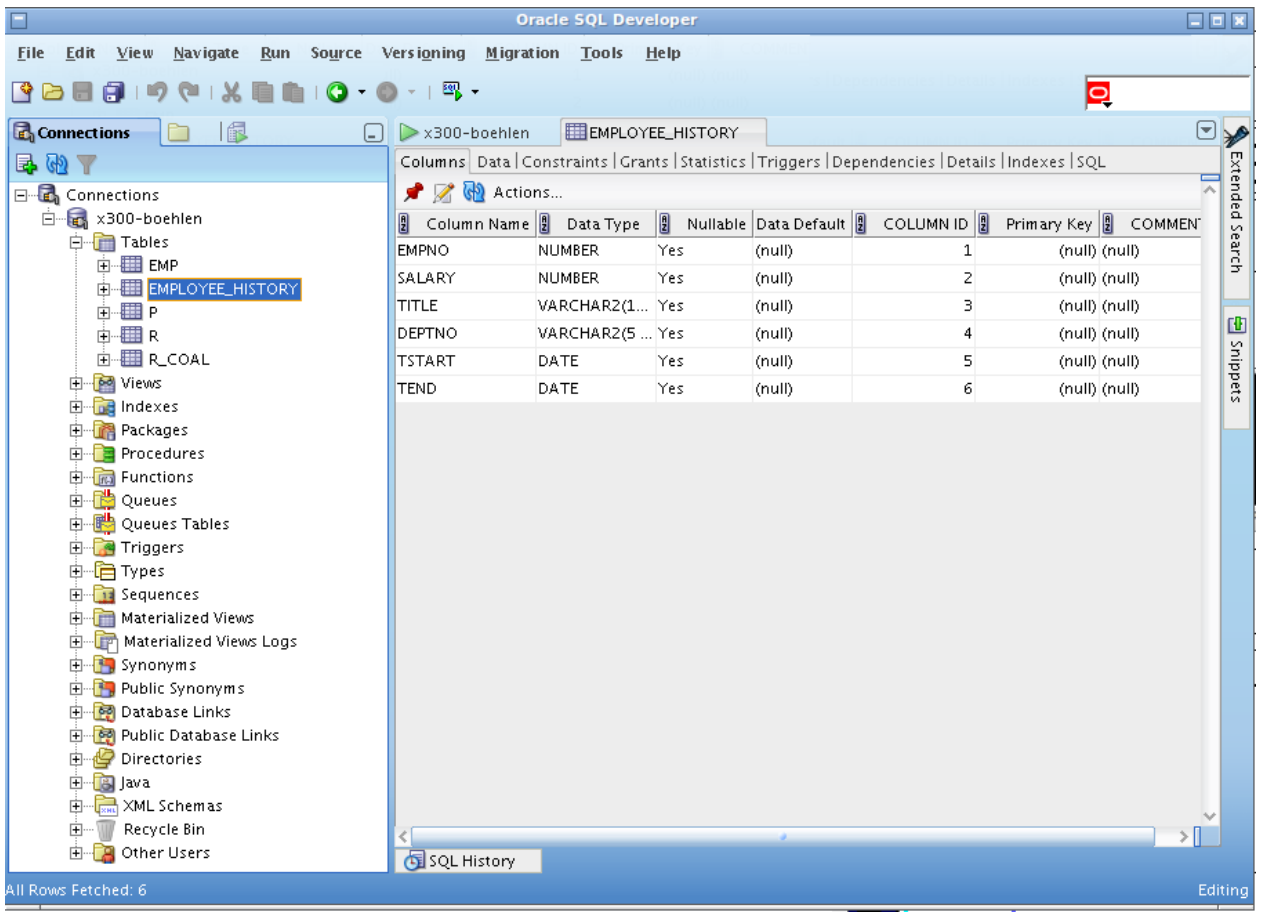
- **User-friendly** interfaces
  - Menu-based, forms-based, graphics-based, etc.
- **Stand-alone** query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. psql in PostgreSQL, sqlplus in Oracle)
- **Program interfaces** for embedding DML in programming languages
- **Web Browser** as an interface
- **Speech** as Input and Output
- **Parametric interfaces**, e.g., bank tellers using function keys.
- **Interfaces for the DBA**:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

DBMS Interfaces/2

- Programmer interfaces for embedding DML in programming languages:
  - Embedded Approach:
    embedded SQL (for C, C++, etc.)
    SQLJ (for Java)
  - Procedure Call Approach:
    JDBC for Java
    ODBC for other programming languages
  - Database Programming Language Approach:
    e.g., ORACLE has PL/SQL, a programming language based on SQL;
    language incorporates SQL and its data types as integral components

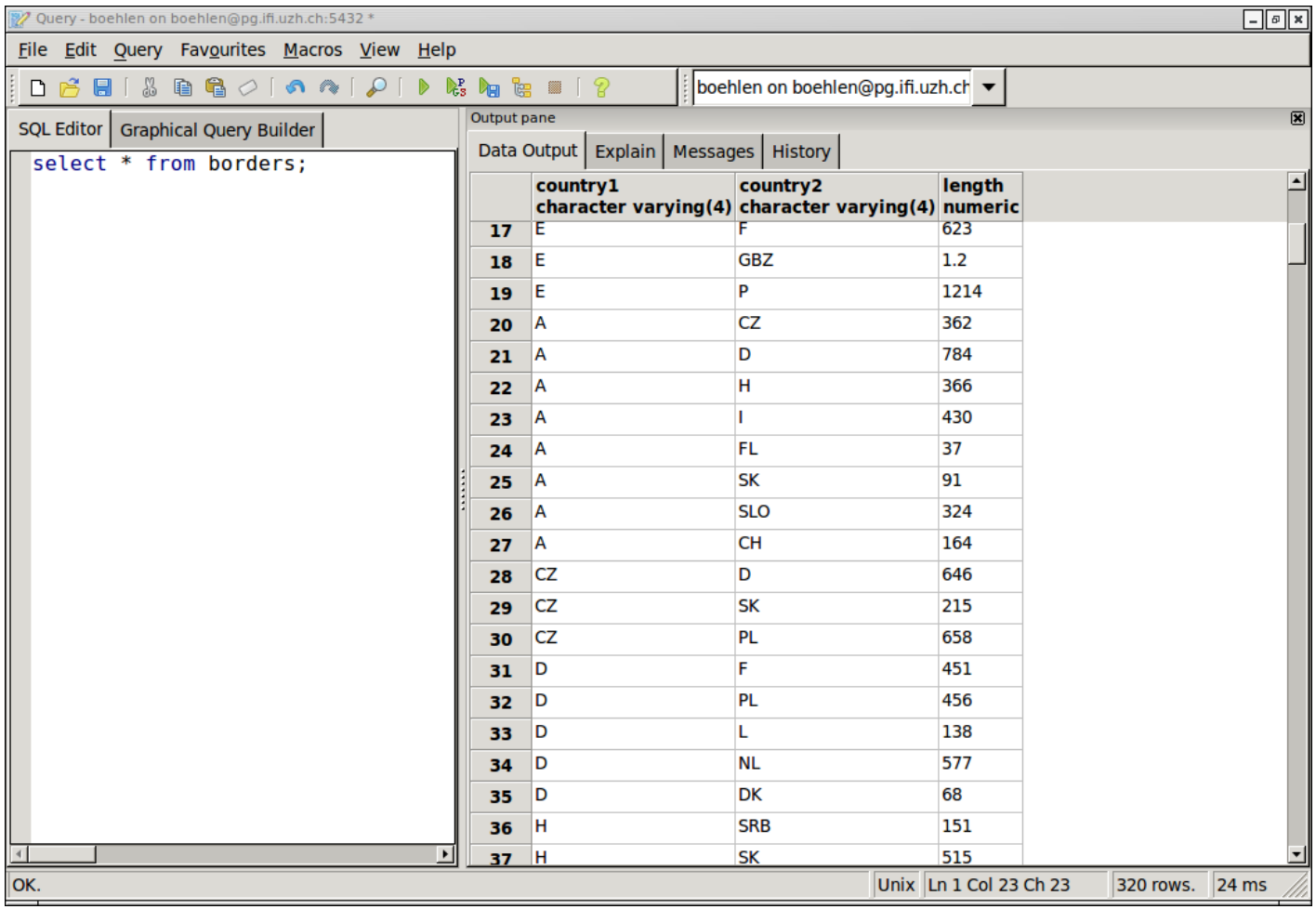
DBMS Interfaces/3

- **Oracle SQL Developer** is a graphical tool for DB development.
- With SQL Developer you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements.



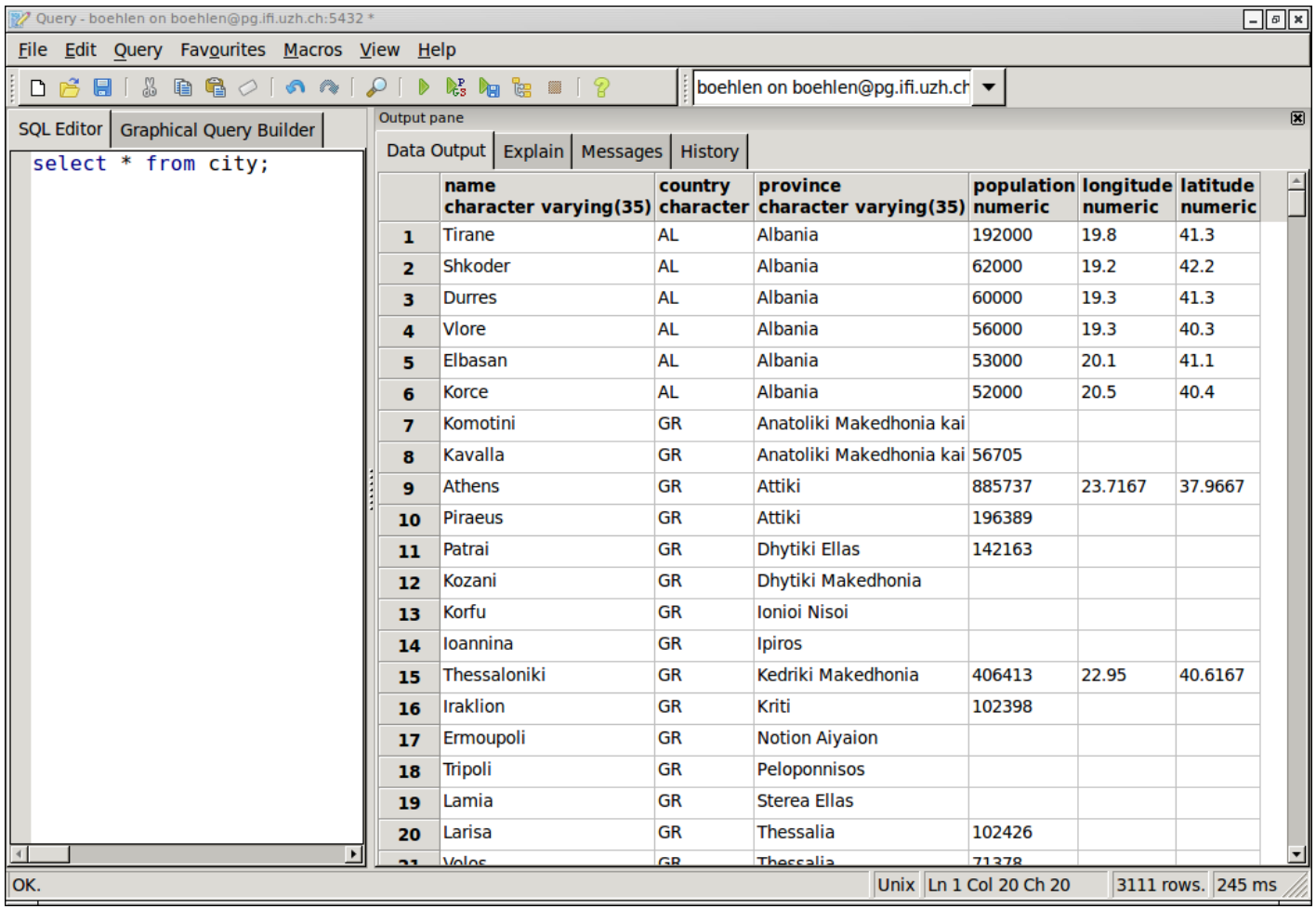
DBMS Interfaces/4

- **pgadmin** is the administration and development platform for PostgreSQL.



DBMS Interfaces/5

- The graphical interface supports all PostgreSQL features, from writing simple SQL queries to developing complex databases.



	name character varying(35)	country character	province character varying(35)	population numeric	longitude numeric	latitude numeric
1	Tirane	AL	Albania	192000	19.8	41.3
2	Shkoder	AL	Albania	62000	19.2	42.2
3	Durres	AL	Albania	60000	19.3	41.3
4	Vlore	AL	Albania	56000	19.3	40.3
5	Elbasan	AL	Albania	53000	20.1	41.1
6	Korce	AL	Albania	52000	20.5	40.4
7	Komotini	GR	Anatoliki Makedhonia kai			
8	Kavalla	GR	Anatoliki Makedhonia kai	56705		
9	Athens	GR	Attiki	885737	23.7167	37.9667
10	Piraeus	GR	Attiki	196389		
11	Patrai	GR	Dhytiki Ellas	142163		
12	Kozani	GR	Dhytiki Makedhonia			
13	Korfu	GR	Ionioi Nisoi			
14	Ioannina	GR	Ipiros			
15	Thessaloniki	GR	Kedriki Makedhonia	406413	22.95	40.6167
16	Iraklion	GR	Kriti	102398		
17	Ermoupoli	GR	Notion Aiyaion			
18	Tripoli	GR	Peloponnisos			
19	Lamia	GR	Sterea Ellas			
20	Larisa	GR	Thessalia	102426		
21	Volos	GR	Thessalia	71378		

DBMS Interfaces/6

- There are various database system utilities to perform certain functions such as:
  - Loading data stored in files into a database. Includes data conversion tools.
  - Backing up the database periodically on tape.
  - Reorganizing database file structures.
  - Report generation utilities.
  - Performance monitoring utilities.
  - Other functions, such as sorting, user monitoring, data compression, etc.

Models, Schemas, Instances and Redundancy

- Data Models
- Database Schema
- Database Instance
- Redundancy

Data Models

- Data Model:
  - A set of concepts to describe the **structure** of a database, the **operations** for manipulating these structures, and certain **constraints** that the database should obey.
- Structure and Constraints:
  - Different constructs are used to define the database structure
  - Constructs typically include **elements** (and their **data types**) as well as **groups** of elements (e.g. **entity**, **record**, **table**), and **relationships** among such groups
  - **Constraints** specify some restrictions on valid data; these constraints must be enforced at all times
- Operations
  - **Operations** are used for specifying database retrievals and updates by referring to the constructs of the data model.
  - Operations on the data model may include basic model operations (e.g. generic insert, delete, update) and user-defined operations (e.g. compute_student_gpa, update_inventory)

Categories of Data Models

- Conceptual (high-level, semantic) data models:
  - Provide concepts that are close to the way many users perceive data. (Also called entity-based or object-based data models.)
- Physical (low-level, internal) data models:
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- Implementation (representational) data models:
  - Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Database Schema

- **Database Schema**:
  - The description of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- **Schema Diagram**:
  - An illustrative display of (most aspects of) a database schema.
- **Schema Construct**:
  - A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- The database schema changes very infrequently.
- Schema is also called **intension**.

Database Instance

- **Database Instance**:
  - The actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database.
  - Also called database state (or occurrence or snapshot).
  - The term instance is also applied to individual database components, e.g., record instance, table instance, entity instance
- **Initial Database Instance**: Refers to the database instance that is initially loaded into the system.
- **Valid Database Instance**: An instance that satisfies the structure and constraints of the database.
- The database instance changes every time the database is updated.
- Instance is also called **extension**.

Example of a Database Description

- Mini-world for the example:
  - Part of a UNIVERSITY environment.
- Some mini-world *entities* (an entity is a specific thing in the mini-world):
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - DEPARTMENTs
  - INSTRUCTORs
- Some mini-world *relationships* (a relationship relates things of the mini-world):
  - SECTIONs are of specific COURSEs
  - STUDENTs take SECTIONs
  - COURSEs have prerequisite
  - COURSE INSTRUCTORs teach SECTIONs
  - COURSEs are offered by DEPARTMENTs
  - STUDENTs major in DEPARTMENTs

Example of a Database Schema

STUDENT

Name	StudNr	Class	Major

COURSE

CourseName	CourseNr	CreditHours	Department

PREREQUISITE

CourseNr	PrerequisiteNr

SECTION

SectionID	CourseNr	Semester	Year	Instructor

GRADE_REPORT

StudNr	SectionId	Grade

Example of a Database Instance

COURSE

CourseName	CourseNr	CreditHours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Databases	CS3360	3	CS

SECTION

SectionID	CourseNr	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

PREREQUISITE

CourseNr	PrerequisiteNr
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

GRADE_REPORT

StudNr	SectionId	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

Redundancy

- During the design of a database the number of tables and their schemas must be determined.
- A key goal of database design is to avoid redundancy.

- **Redundancy** is present if information is stored multiple times.
- Example of redundancy: storing the same address multiple times
- Redundancy leads to update anomalies and inconsistent data (e.g., a person has multiple and partially invalid addresses)
- The goal of database design, and specifically of database normalization, is to eliminate redundancy.
- The term **controlled redundancy** is used if duplication of information is allowed and if the duplication is controlled by the DBMS.

Review 1.2

Consider the university database instance shown above.

1. Explain why this schema contains redundancy.
2. Give an example of a change that leads to update anomalies.
3. Propose a modified schema that eliminates the redundancy.

Review 1.2

Main Characteristics of Database Systems

- Three Schema Architecture
- Data Independence
- Main Characteristics

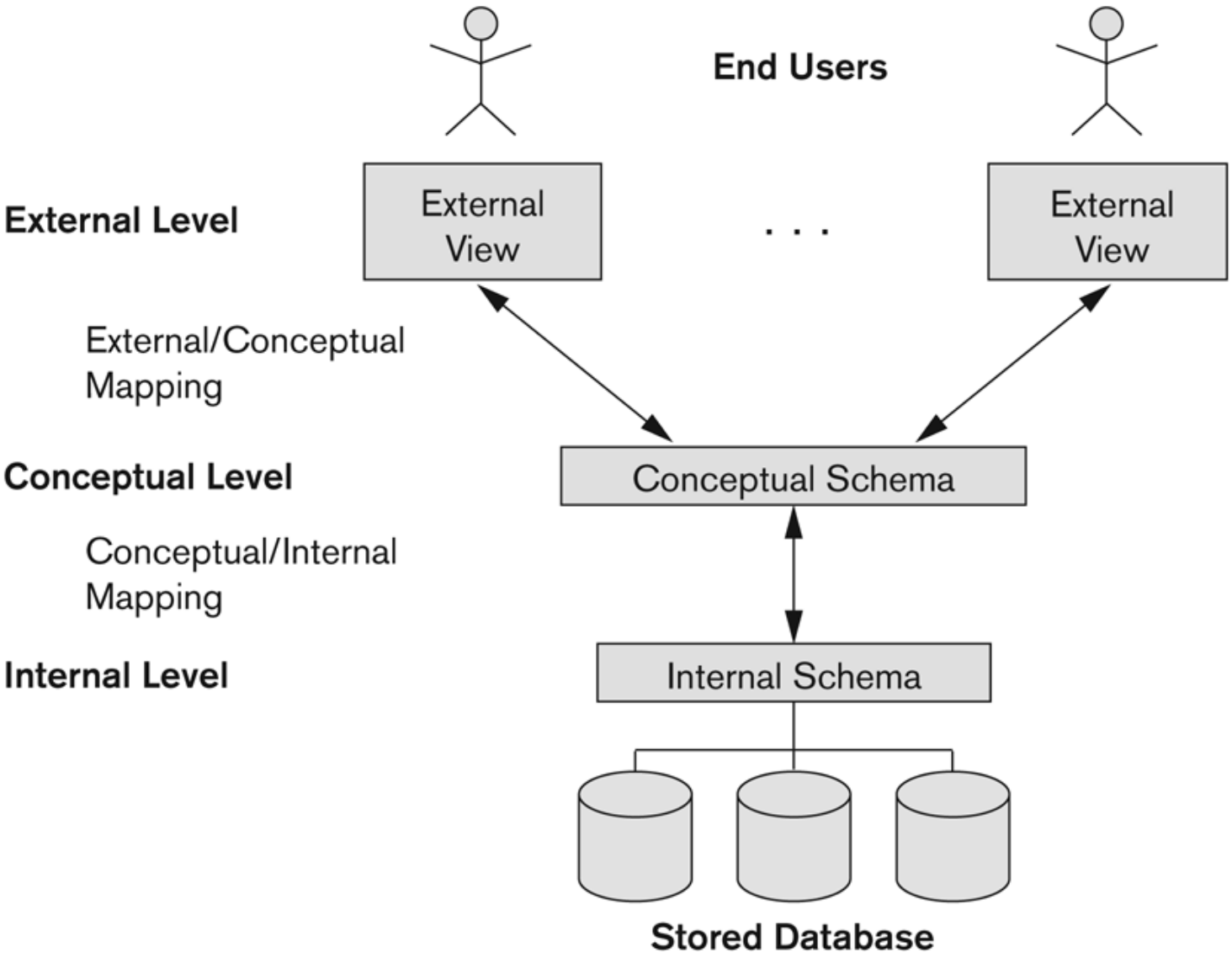
The ANSI/SPARC Three Schema Architecture/1

- Proposed to support DBMS characteristics of:
  - Data independence
  - Multiple views of the data
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization.
- Defines DBMS schemas at three levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a physical data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
    - Uses a conceptual or an implementation data model.
  - **External schemas** at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

The ANSI/SPARC Three Schema Architecture/2

- Mappings among schema levels are needed to transform requests and data.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g., formatting the results of an SQL query for display in a Web page)

The ANSI/SPARC Three Schema Architecture/3



Data Independence

- **Logical Data Independence**:
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- **Physical Data Independence**:
  - The capacity to change the internal schema without having to change the conceptual schema.
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance
- When a schema at a lower level is changed, only the mappings between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are unchanged.
  - Hence, the application programs need not be changed since they refer to the external schemas.

Review 1.3

1. Give real world examples of data independence.

Main Characteristics of Database Approach/1

- Insulation between programs and data:
  - Called **data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.
- Control of **redundancy**:
  - Database systems control (and minimize) redundancy
  - The control allows to avoid inconsistent data (happens if only one copy is updated)
- **Data abstraction**:
  - A data model is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- Support of **multiple views** of the data:
  - Each user may see a different view of the database, which describes only the data of interest to that user.

Main Characteristics of Database Approach/2

- **Sharing** of data and **multi-user** transaction processing:
  - Allowing a set of concurrent users to retrieve from and to update the database.
  - Concurrency control within the DBMS guarantees that each transaction is correctly executed or aborted
  - Recovery subsystem ensures each completed transaction has its effect permanently recorded in the database
  - OLTP (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.
- **Self-describing** nature of a database system:
  - A DBMS catalog stores the description of a particular database (e.g. data types, data structures, and constraints)
  - The description is called **metadata**.
  - This allows the DBMS software to work with different database applications.

Main Characteristics of Database Approach/3

Example of a DBMS catalog (just the idea; oversimplified):

RELATIONS

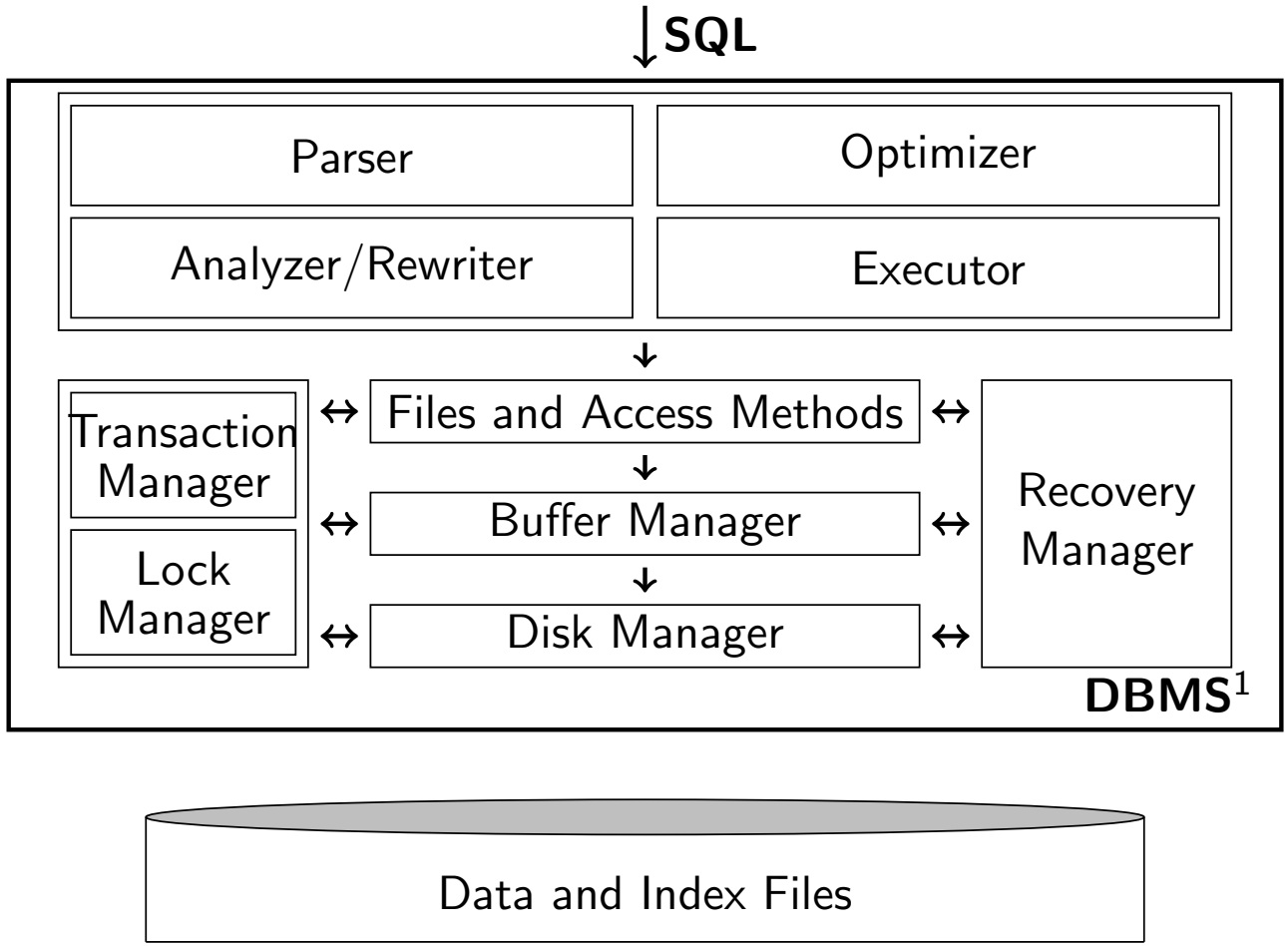
RelationName	NrOfColumns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

ColumnName	DataType	BelongsToRelation
Name	Character(30)	STUDENT
StudentNr	CHARACTER(4)	STUDENT
Class	INTEGER(1)	STUDENT
...	...	...

- PostgreSQL 8.3.9: 74 objects in the system catalog
- Oracle 10.2: 1821 objects in the system catalog

DBMS Architecture



[1] Image: Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill 2003

History

- History of Database Technology
- Advantages of Database Technology
- Limitations of Database Technology

History of Database Technology/1

- **Network Model**:
  - The first network DBMS was implemented by Honeywell in 1964-65 (IDS System).
  - Adopted heavily due to the support by CODASYL (Conference on Data Systems Languages) (CODASYL - DBTG report of 1971).
- Advantages:
  - The network model is able to model complex relationships.
  - Can handle most situations for modeling using record types and relationship types.
  - Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET, etc.
  - Programmers can do optimal navigation through the database.
- Disadvantages:
  - Navigational and procedural nature of processing
  - Database contains a complex array of pointers that thread through a set of records.
  - Little scope for automated query optimization

History of Database Technology/2

- **Hierarchical Data Model**:
  - Initially implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems.
  - IBM's IMS product had (and still has) a very large customer base worldwide
  - Hierarchical model was formalized based on the IMS system
  - Other systems based on this model: System 2k (SAS inc.)
- Advantages:
  - Simple to construct and operate
  - Corresponds to a number of natural hierarchically organized domains, e.g., organization chart
  - Language is simple; Uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT, etc.
- Disadvantages:
  - Navigational and procedural nature of processing
  - Database is visualized as a linear arrangement of records
  - Little scope for "query optimization"

History of Database Technology/3

- **Relational Model**:
  - Proposed in 1970 by E.F. Codd (IBM)
  - Heavily researched and experimented within IBM Research and universities
  - First commercial system in 1981-82.
  - Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
  - Several free open source implementations, e.g. MySQL, PostgreSQL
  - Currently most dominant for developing database applications.
  - SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, ...
- Advantages:
  - High level of abstraction (conceptual and physical level are separated)
  - Elegant mathematical model
  - High level (declarative) query languages
- Disadvantages:
  - Performance (was slow at the beginning because there is no navigational access to data)

History of Database Technology/4

- **Object-oriented models**:
  - Object-oriented database management systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
  - OBJECTSTORE, VERSANT, GEMSTONE, O2, ORION, IRIS.
  - Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.
  - Pure OODBMSs have disappeared. Many relational DBMSs have incorporated object database concepts, leading to a new category called object-relational DBMSs (ORDBMSs).
- **Data on the web and E-commerce applications**:
  - Web contains data in HTML with links among pages.
  - This has given rise to a new set of applications and E-commerce is using standards like XML.
  - Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database.

History of Database Technology/5

- **New functionality** is being added to DBMSs in the following areas:
  - Scientific Applications
  - XML (eXtensible Markup Language)
  - Image Storage and Management
  - Audio and Video Data Management
  - Data Warehousing and Data Mining
  - Spatial Data Management
  - Time Series and Historical Data Management
  - Key-value stores (NoSQL)
- The above gives rise to new research and development in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

Advantages of Using a DBMS/1

- Controlling redundancy in data storage.
- Restricting unauthorized access to data.
- Providing persistent storage for program objects.
- Providing storage structures (e.g., indexes) for efficient query processing.
- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database (= good data quality).
- Drawing inferences and actions from the stored data using deductive and active rules.

Advantages of Using a DBMS/2

- Potential for enforcing standards:
  - This is very crucial for the success of database applications in large organizations. Standards refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.
- Flexibility to change data structures:
  - Database structure may evolve as new requirements are defined.
- Availability of current information:
  - Extremely important for on-line transaction systems such as airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

When to not use a DBMS

- Main inhibitors of using a DBMS:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change.
  - If there are stringent real-time requirements that may not be met because of DBMS overhead.
  - If access to data by multiple users is not required.
- When no DBMS may suffice:
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS.

Summary/1

- Data models, schemas, instances
  - **data model** = structures + constraints + operations
  - **schema** = intension; schema consists of structures and constraints; schema changes infrequently
  - **relation instance** = relation = extension; relation instance is the actual data that is compatible with the schema; changes often
- Key characteristics of database systems
  - **controlled redundancy**: database systems is aware of redundancy and provides support for updates that could violate the consistency of the data
  - **data independence**: separation of program and data; makes it possible to, e.g., reorganize internal schema without changing conceptual schema
  - **data abstraction**: high level query language that is independent of storage structure
  - **data dictionary** (metadata) that stores information about the database itself (self-describing)

Summary/2

- Three-Schema Architecture
  - multiple views of the data
  - ANSI/SPARC three schema arcitecture
  - external, conceptual, and internal schema
- DBMS Languages and Interfaces
  - stand-alone command line interfaces: psql, sqlplus, ...
  - programming interfaces: ODBC, JDBC
  - database development tools: pgadmin, SQL developer
- Architectures and History
  - 1-tier, 2-tier, 3-tier
  - network, hierarchical, relational, object-oriented, object-relational