The picture can't be displayed.

Chapter 1: Introduction

The original presentation is infused with more information and slides by Verena Kantere

Database System Concepts, 6th Ed.

Outline

- The Need for Databases
- Data Models
- Relational Databases
- Storage Manager
- Query Processing
- Transactions
- Architecture
- History
- Current Status

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large
- Databases touch all aspects of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Levels of Abstraction

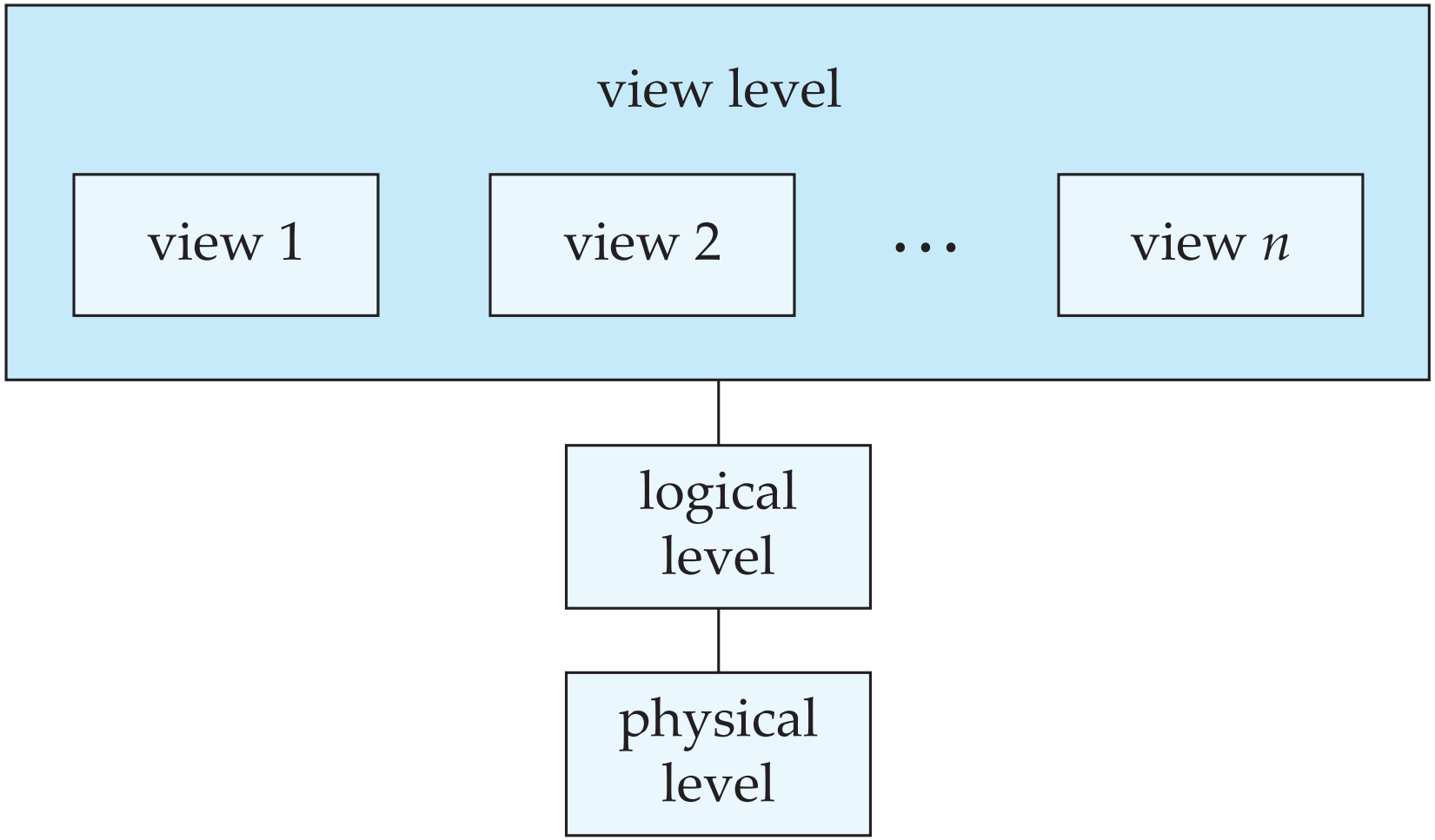
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

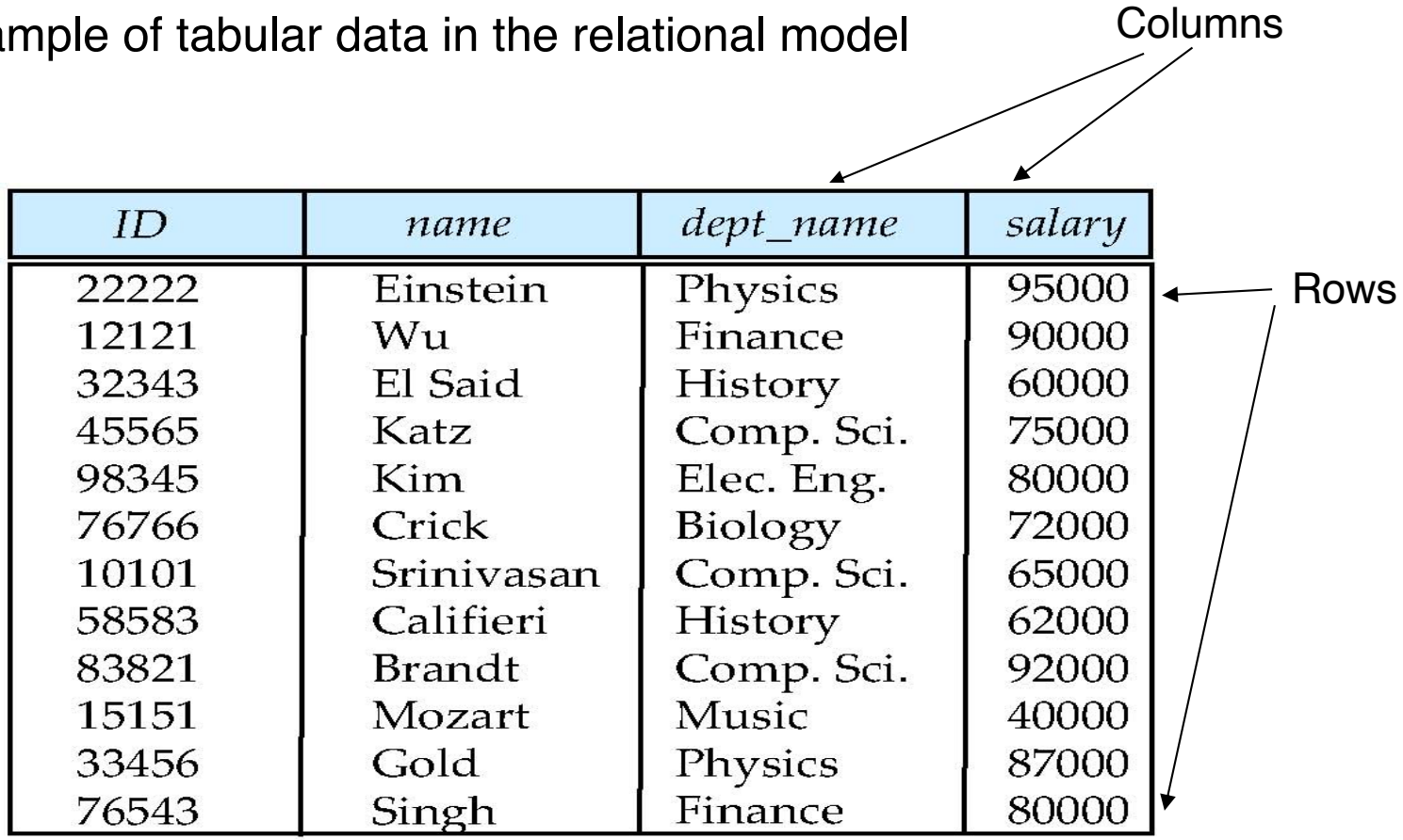
- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well-defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language*
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

* You may come across DQL (Data Query Language). DQL is traditionally considered to be part of DML.

By the way, you may also come across: DCL (Data Control Language) and TCL (Transaction Control Language).

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database
- Open Database Connectivity (ODBC) is a standard API for accessing DBMSs, independently of the DBMS or the OS
  - An application written using ODBC can be ported to other platforms, with few changes to the data access code.
- ODBC accomplishes DBMS independence by using an *ODBC driver* as a translation layer between the application and the DBMS.

Object-Relational Data Models

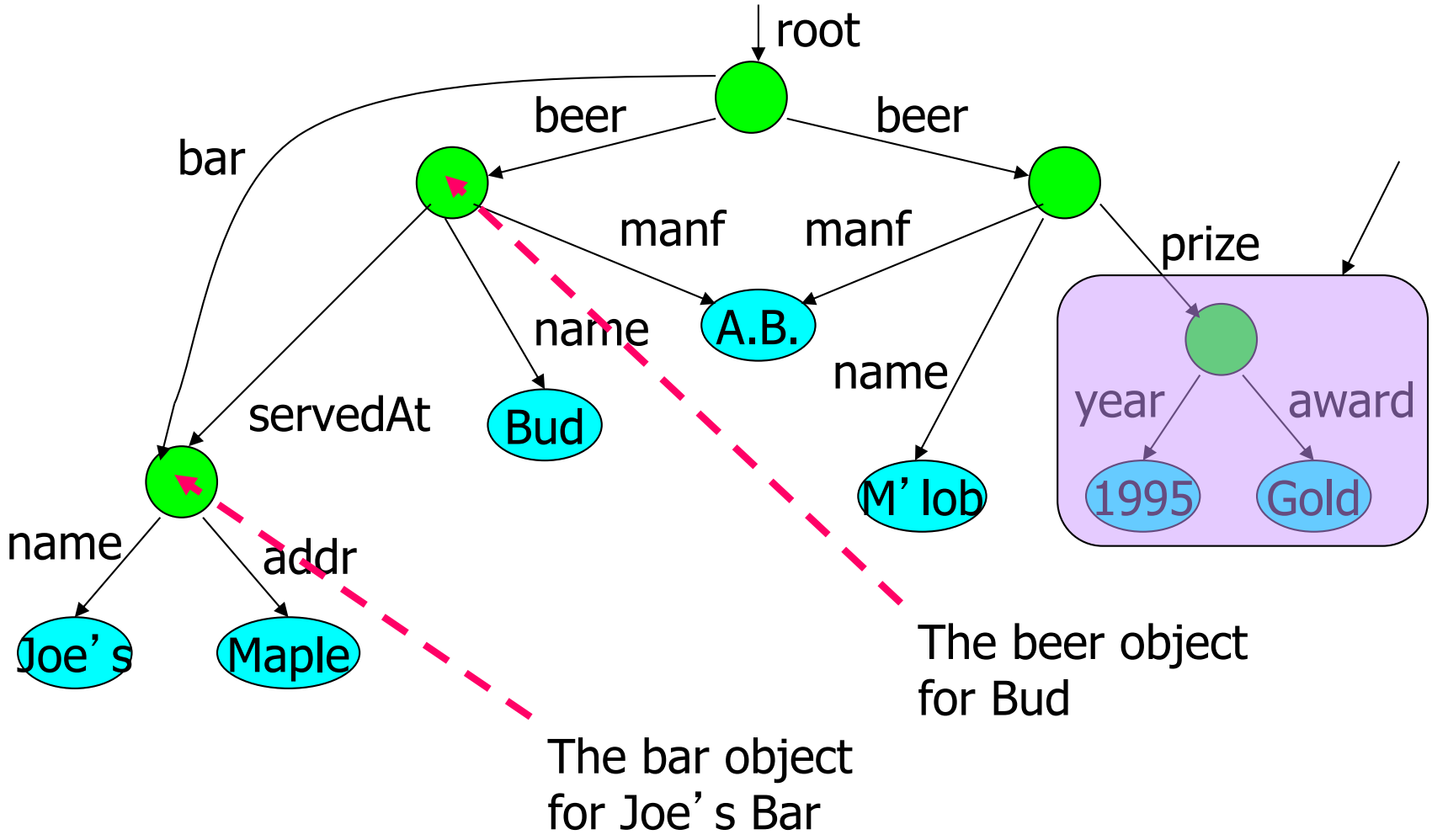
- Relational model: flat, “atomic” values
- Object-Relational Data Models
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

- XML is a language for semistructured data

Graph for semi-structured data



Example: Well-Formed XML

```
<? XML VERSION = “1.0” STANDALONE = “yes” ?>
<BARS>
    <BAR><NAME>Joe’s Bar</NAME>
        <BEER><NAME>Bud</NAME>
            <PRICE>2.50</PRICE></BEER>
        <BEER><NAME>Miller</NAME>
            <PRICE>3.00</PRICE></BEER>
    </BAR>
    <BAR> …
</BARS>
```

Database Engine

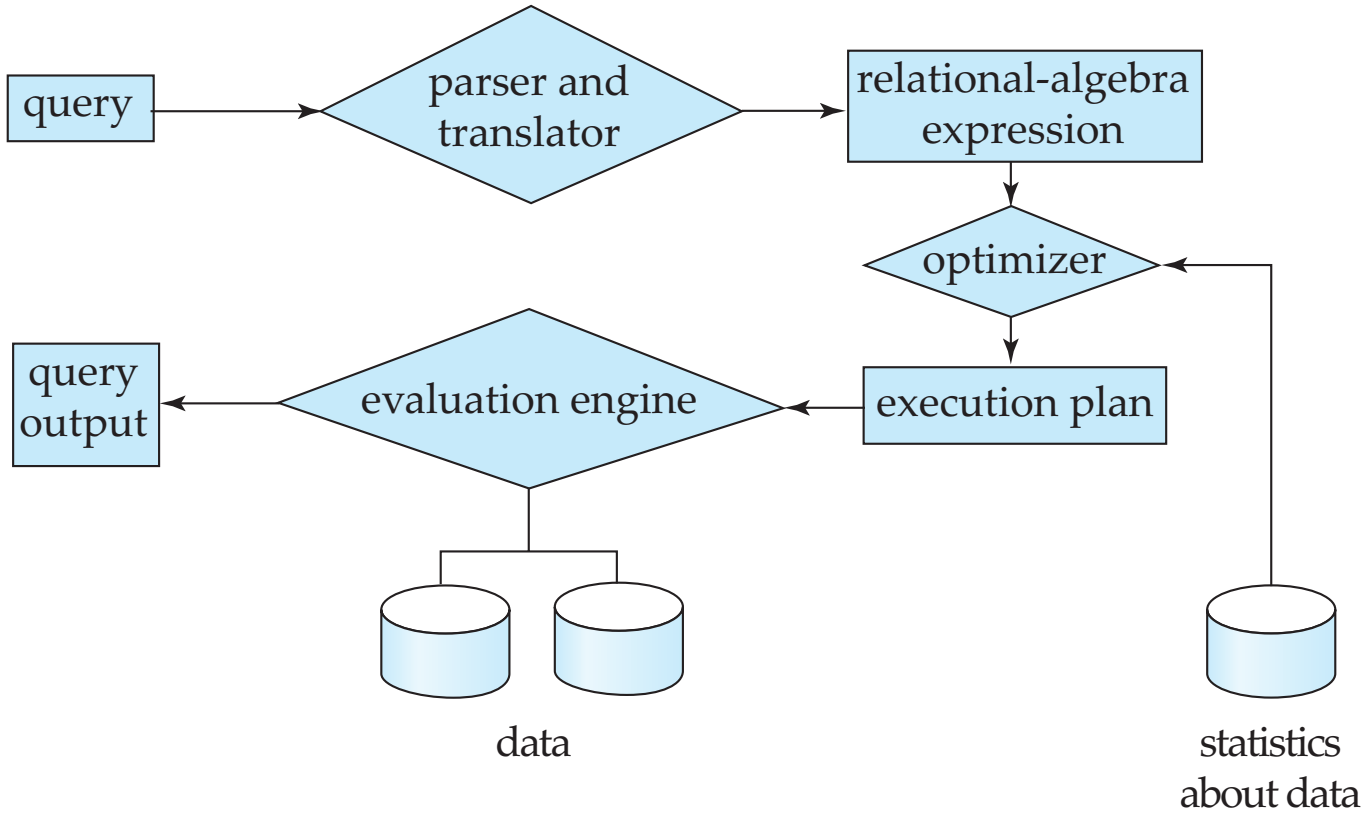
- Storage manager
- Query processing
- Transaction manager

Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



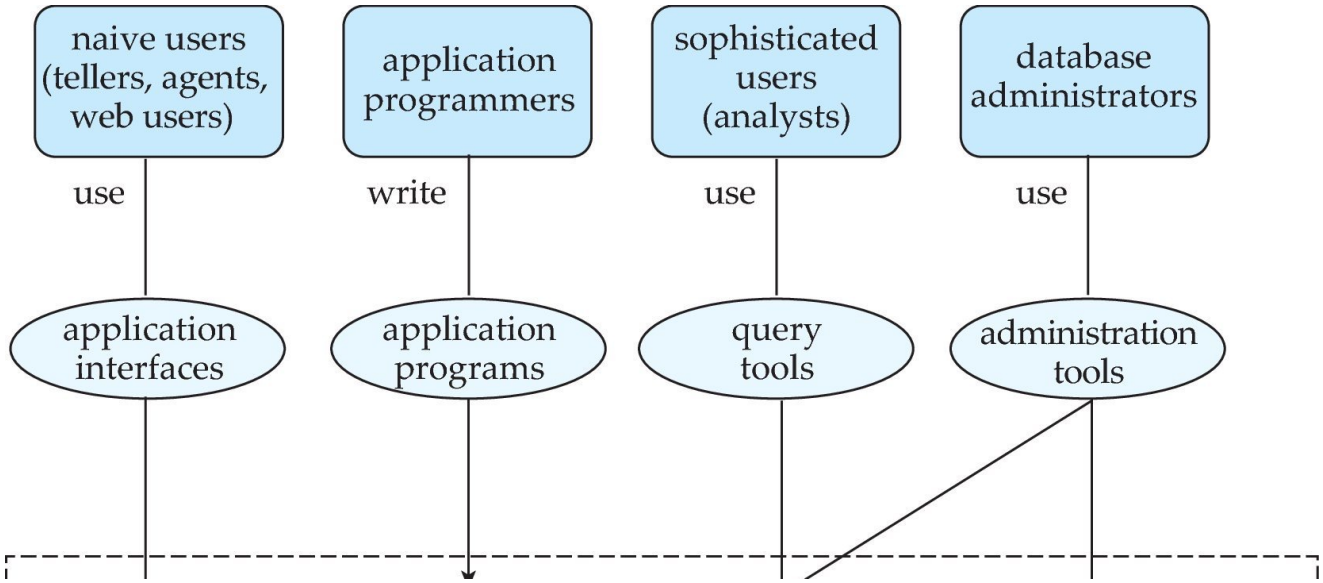
Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

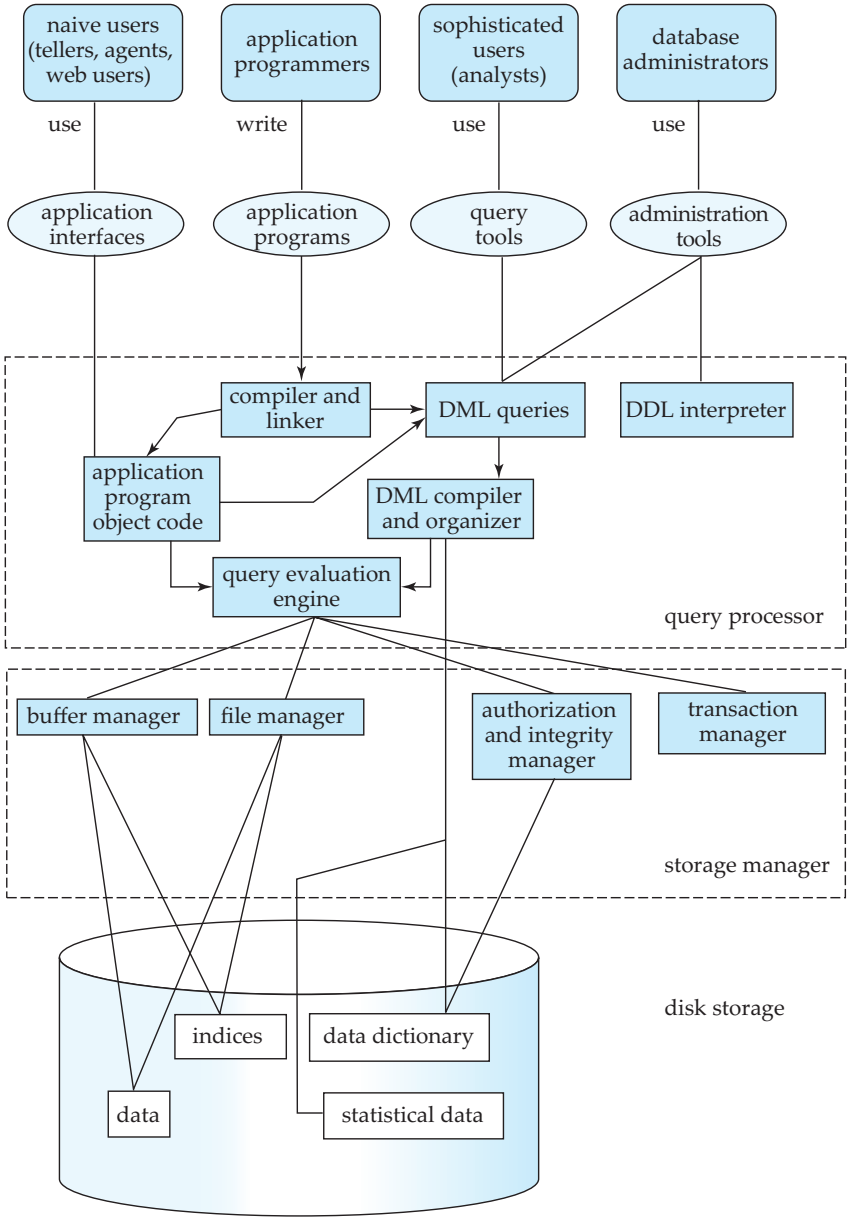
Database Users and Administrators



Databases make these folks happy ...

- End users and DBMS vendors
- DB application programmers
  - e.g. webmasters
- *Database administrator (DBA)*
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve
  - ***Must understand how a DBMS works!***

Database System Internals



Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

History of Database Systems

Slide for extended discussion

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing

History (cont.)

Slide for extended discussion

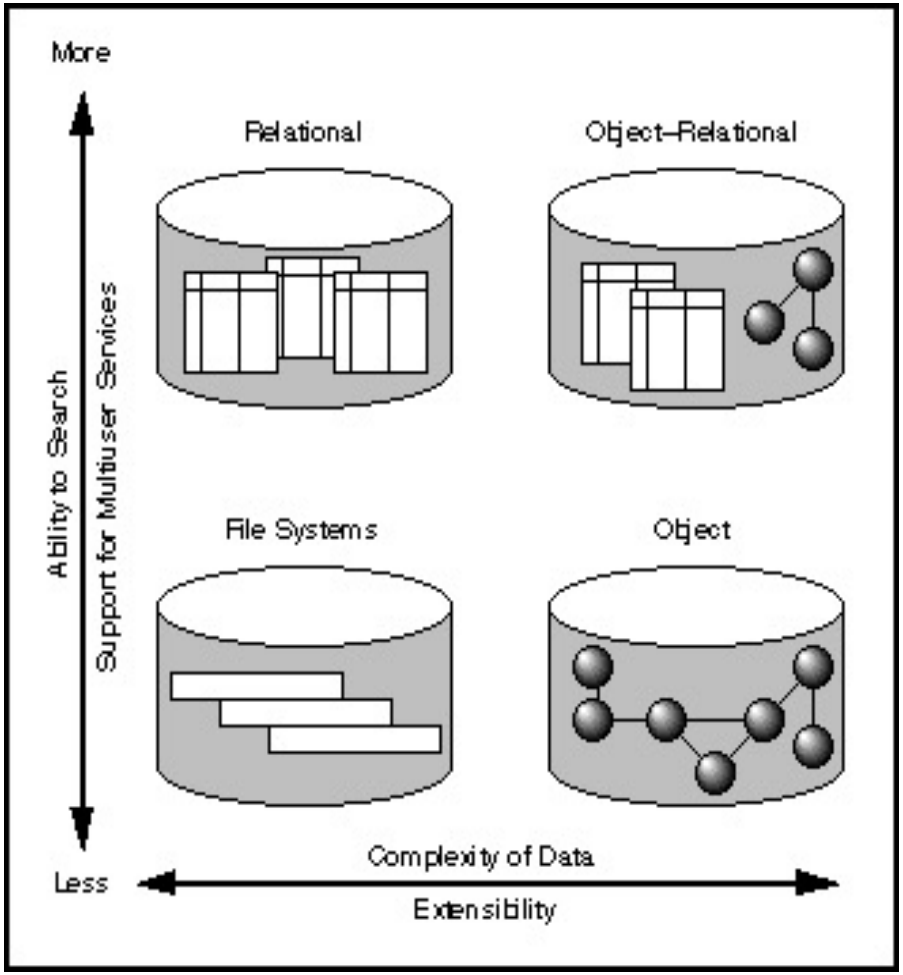
- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

History (cont.)

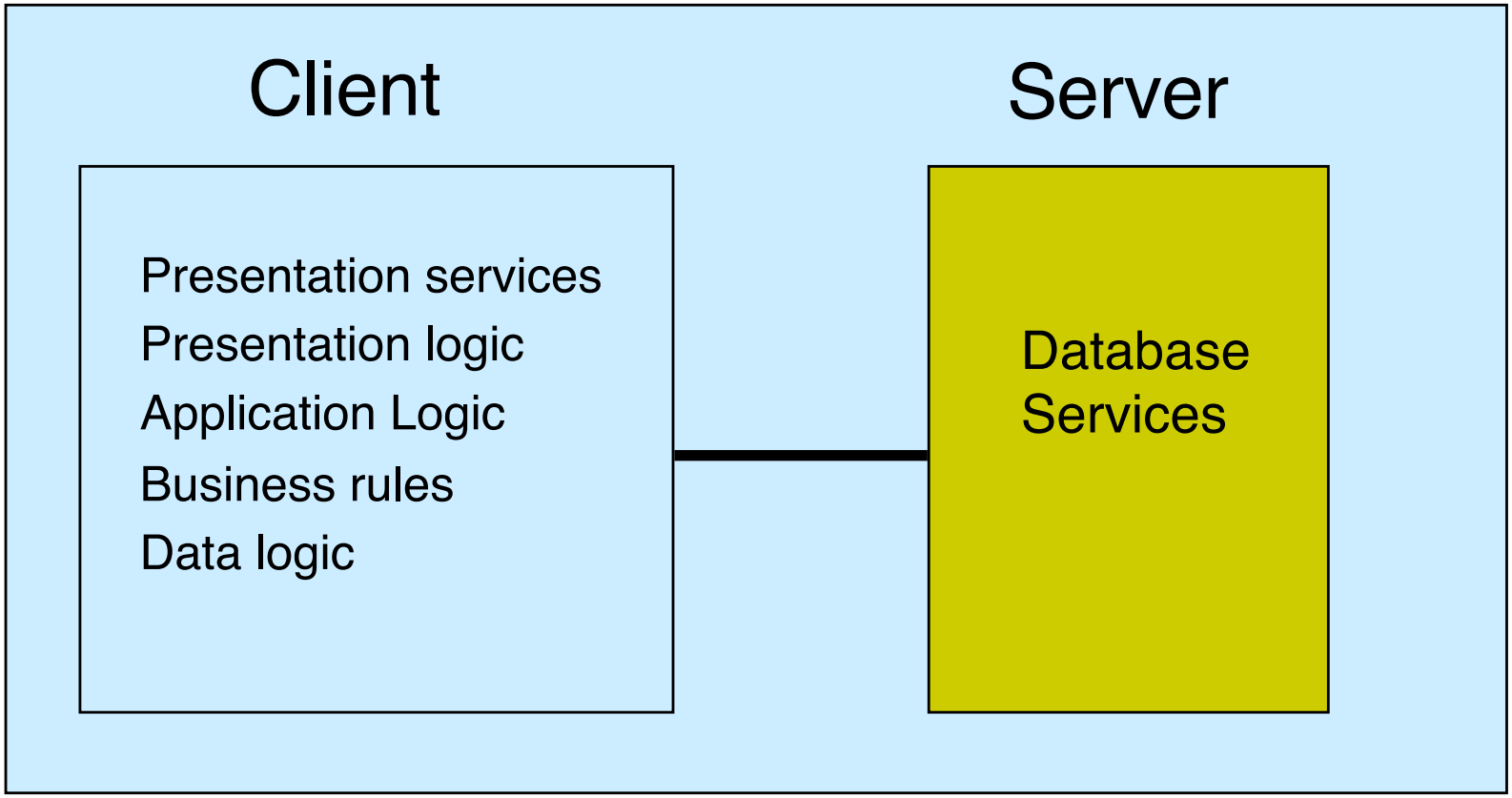
Slide for extended discussion

- 1990-2000s (Meta-relational era)
  - It is the era of COMPLEX ENTITIES (engineering objects, multimedia, software objects)
    - Object-Relational Database Systems
  - Active Databases, Intelligent Systems, Multimedia databases
  - Client-Server
  - Data Warehouses
  - Data Mining
  - Distributed Databases, parallelization
    - DBMS in PCs
    - DBMS on the Internet (Web-based), Java, XML, …, Cloud

Relational and Object-oriented DBs



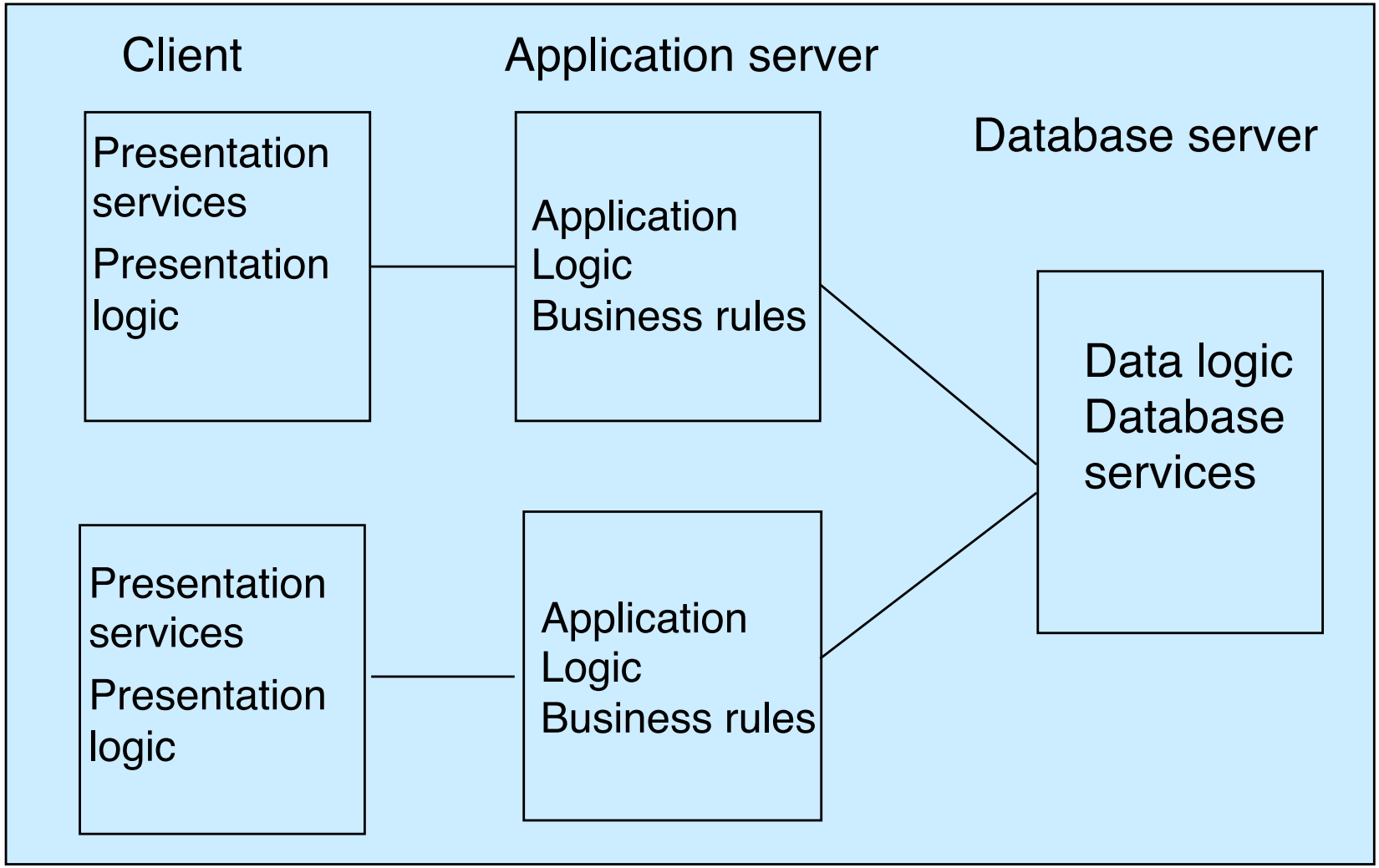
First generation of Client-Server



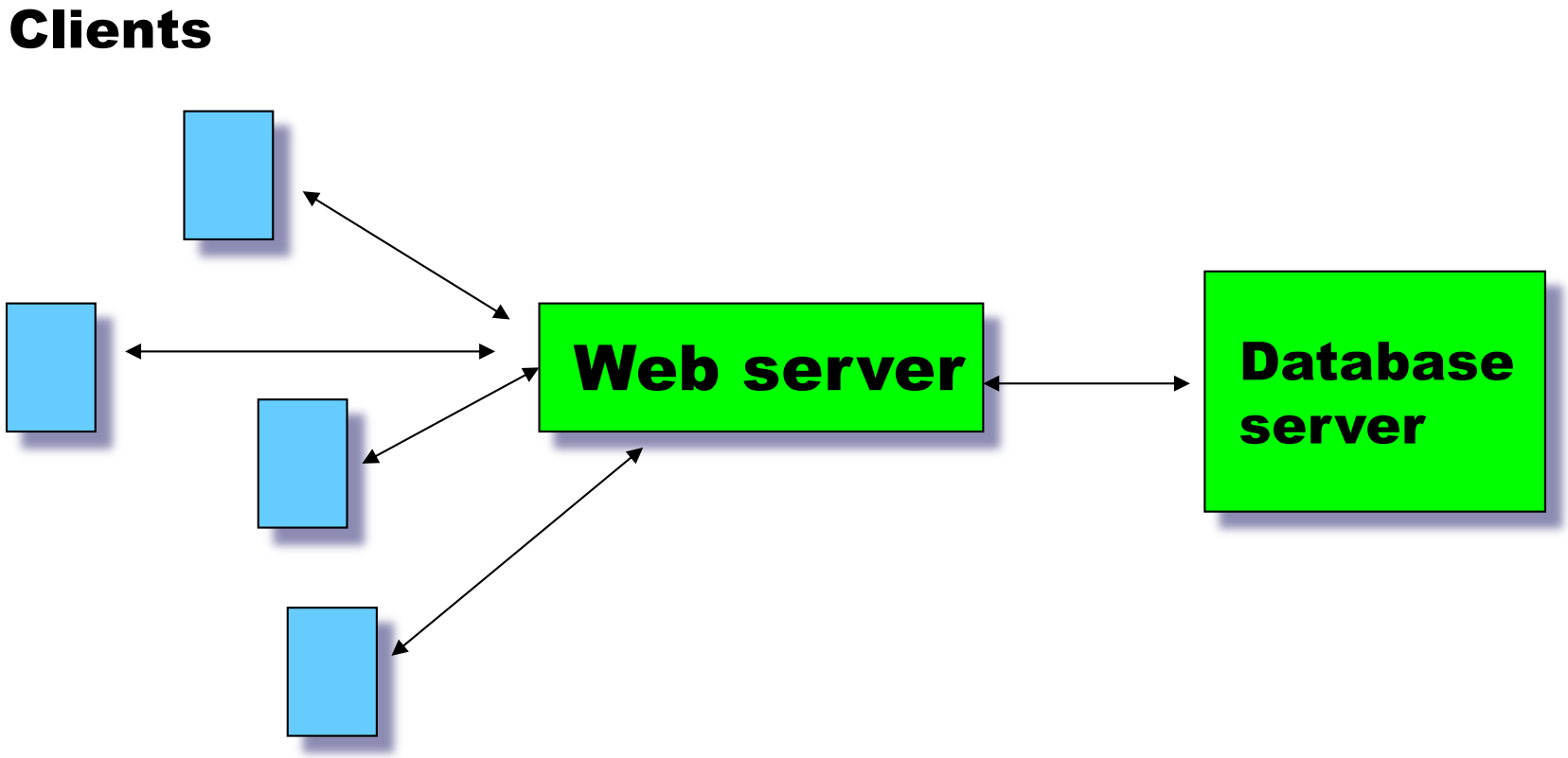
Architectural elements

- **Presentation services**
  - Take inputs and show results
- **Presentation logic**
  - Controls the relation user-application
- **Application logic**
  - Calculations, decisions, actions for the application
- **Business rules**
  - From the whole business
- **Data logic**
  - Creation of queries (e.g. in SQL)
- **Database services**
  - Service reference to data

3-Tier architecture
Second generation Client-Server



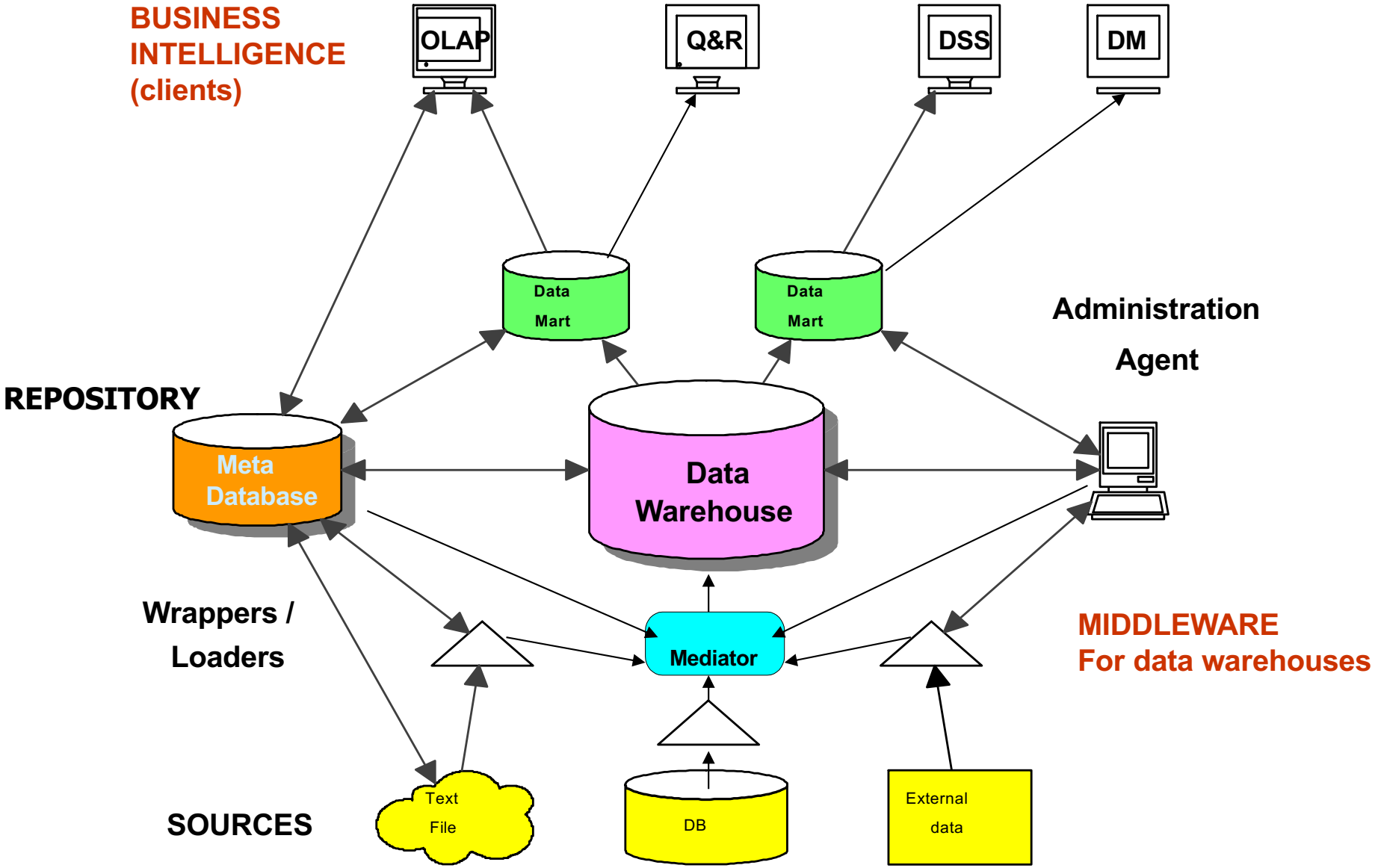
Today's typical architecture



Technology for managing data

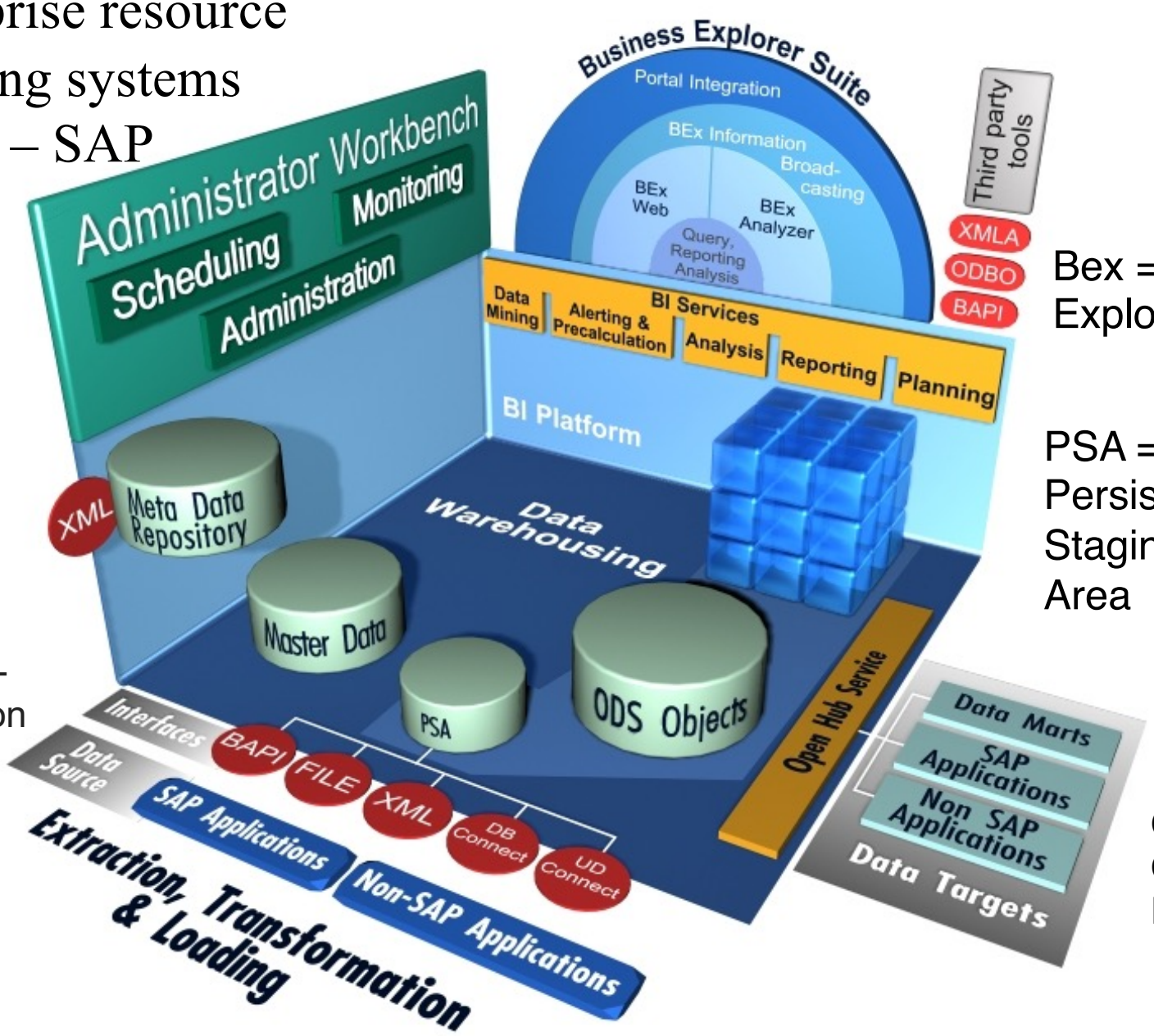
- **OLAP (Online Analytic Processing)**
  - Analysis in a multi-dimensional space
    - » Sales per (product, customer, time)
    - » Cube
  - drill down, rollup
  - Essbase, Commander, Oracle Express, SAS, Excel, SQL Server,…
- **Data Warehouses**
  - New form for Decision Support Systems
  - Redbrick, Oracle DW, Informix, Sybase, Micro Strategy
- **Data Marts (smaller Data Warehouses)**
- **Data Mining (extraction of information)**

Classical architecture for data warehouse



Slide for extended discussion

Enterprise resource planning systems (ERP) – SAP



An **ODS object** acts as a storage location for consolidated and cleaned-up transaction data

Bex = Business Explorer

PSA = Persistent Staging Area

ODS = Operational Data Store

Data Mining

- **Automatic analysis of data**
- **Statistics**
  - Correlation
  - Regression (multiple correlation)
  - Clustering
  - Classification
  - Nonlinear relationships
- **More automatizations**
  - Analysis of the market basket
- **Numerical data and non-numerical data**
  - Language analysis

Market Basket Analysis

What do customers buy at the same time?

Data Mining: Market Basket Analysis

- **Goal: Measure the relation between two objects**
  - What products do customer buy together?
  - Which web pages do users visit together?
- **Classical examples**
  - In convenient stores that are open on Sunday, it was found that customers buy together beers and baby pampers.
  - Amazon.com: shows correlated purchases
- **Strategic use of such information**
  - Decide to place products together in order to increase cross selling
  - Alternatively, put them at the start an the end of the corridor in order for customers to buy even more products

DBMS : Old protagonists

- **SYBASE**
- **INGRES** now it is called Computer Associates-Ask Group and has become and open source system
- Others:
  - **Rdb, Gupta Quadbase, Ralma, Watcom, XDB**, ...

DBMS : Old protagonists

Today there are 3 very popular systems:

- **IBM Db2 13**, (also in Unix, Linux, Windows)
- **ORACLE 21 (21C)** (2021) almost everywhere – first in the market especially for Unix and large installations
- **Microsoft SQL Server 2022,** in Microsoft platforms
  - **INFORMIX** (Bought by IBM!)

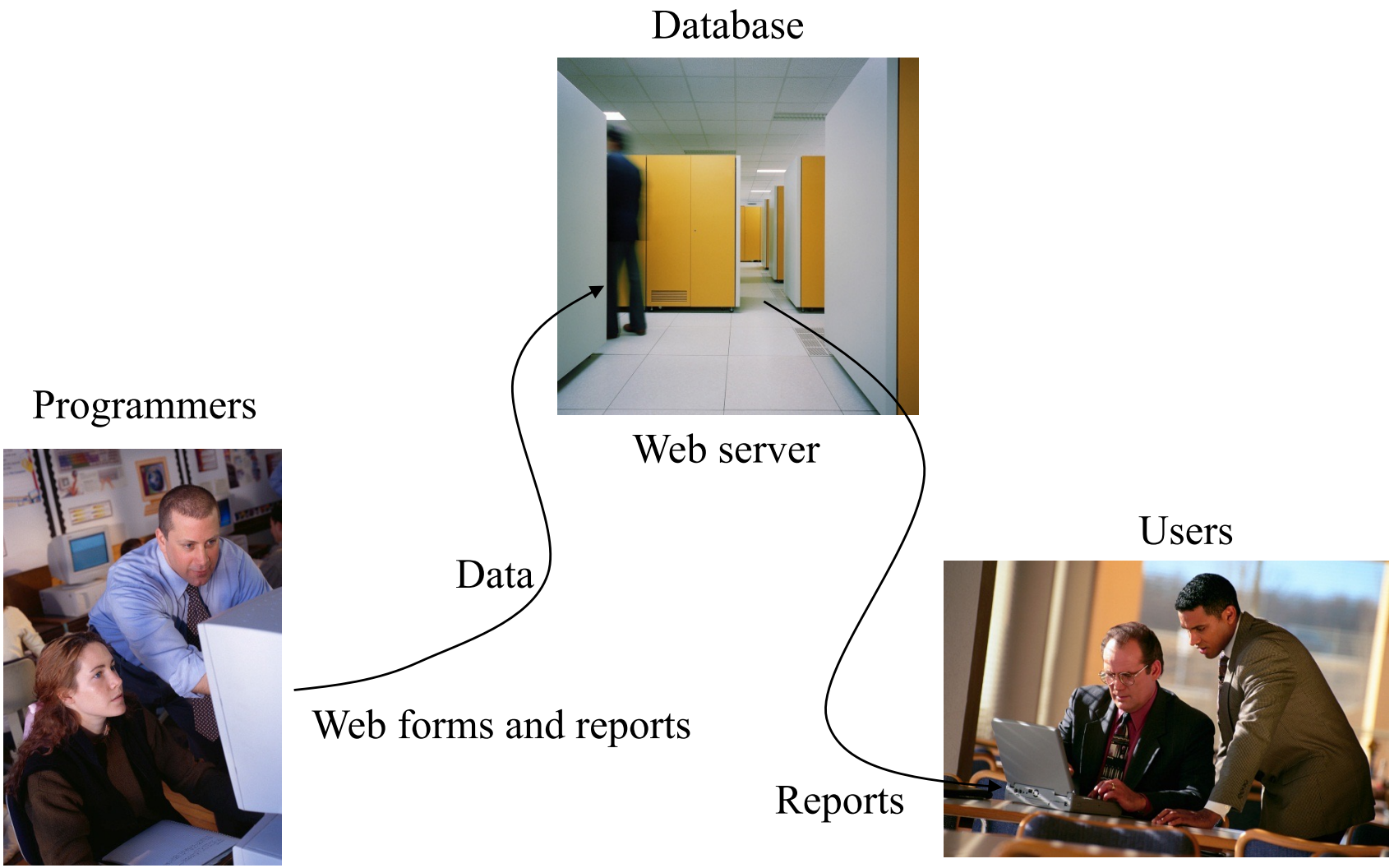
OPEN Source DBMS

- MySQL
- PostgreSQL + EnterpriseDB
- Berkeley DB, Firebird, etc.

There are also memory DBMSs, e.g.:

- SQLite (classical opensource DBMS)

Web data bases



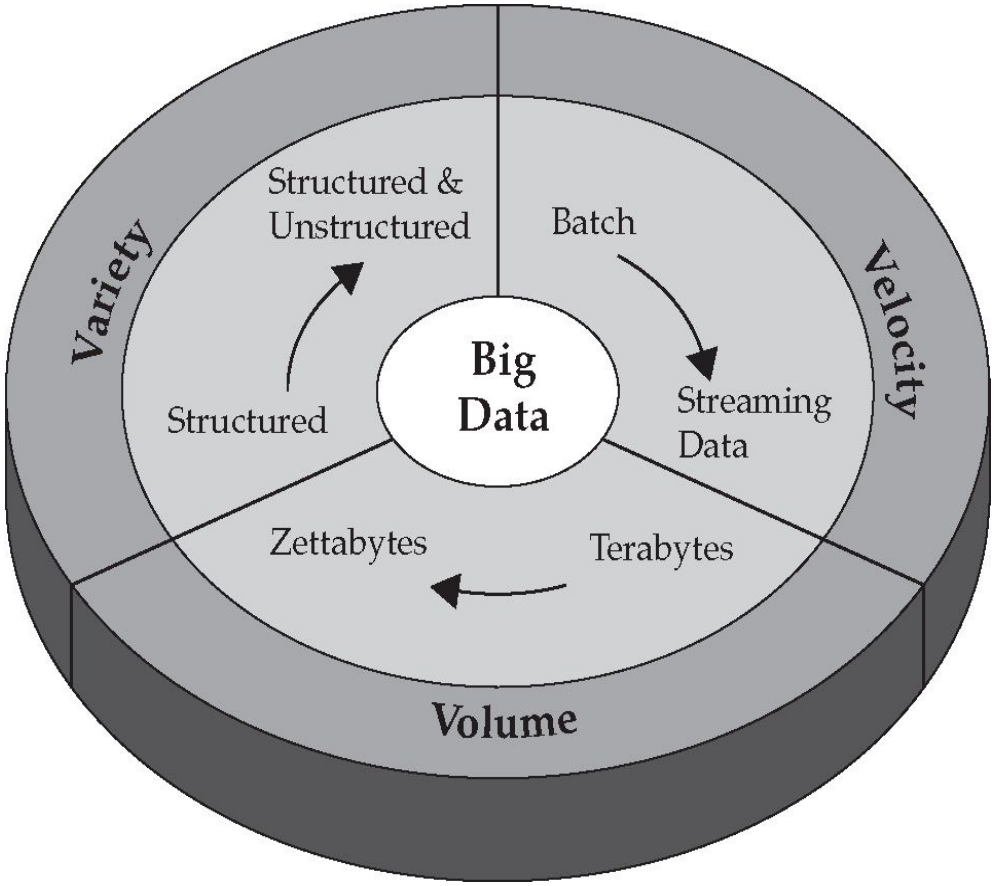
Web data bases

- More than 1.7B web pages
- Extraordinary amount of information
  - Bookstores, restaurants, trip advisors, purchases, news, financial, guides, maps etc etc
  - Many forms: text, image, voice, video…
  - Many formats: HTML, XML, postscript, pdf, JPEG, MPEG, MP3
- With special dynamic characteristics
  - More than 500’000 new pages a day
  - Graph structure among pages
- 100M queries a day

BIG DATA – Characteristics

- Volume -- In 2000 we had about 800,000 petabytes (PB) stored data in the whole world. We anticipate that this number will reach 35 zettabytes (ZB) until 2020. (Twitter alone produces more than 7 terabytes (TB) per day, Facebook 10 TB, and several enterprises many terabytes an hour. – update: it in 2020 we've reached 44 ZB and we'll reach 175ZB by 2025!
- Variety -- Structured, semi-structured, unstructured
- Velocity – Speed of production (Batch to streams)

BIG DATA – Characteristics



Later on:

4th v: Veracity = αλήθεια

5th v: Value

Today:

6th v: Variability = μεταβλητότητα

7th v: Visualization

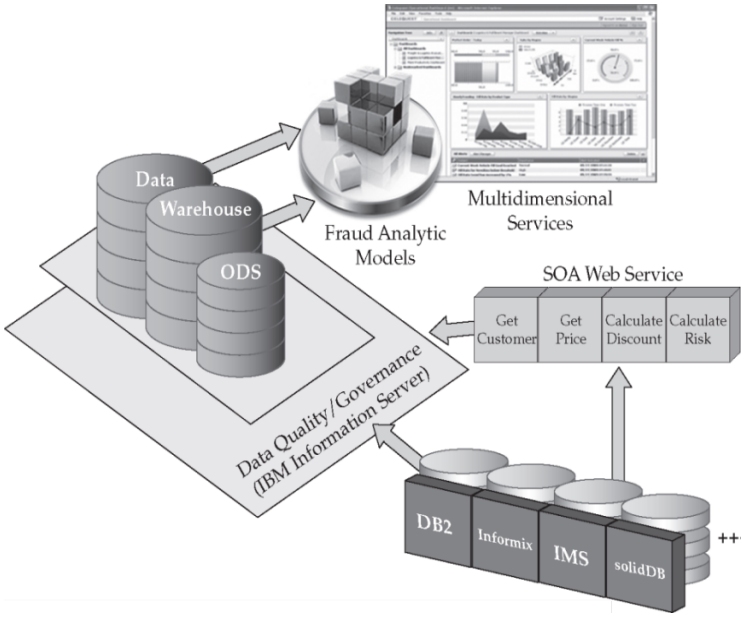
Some also add:

8th v: Validity = ισχύ, εγκυρότητα

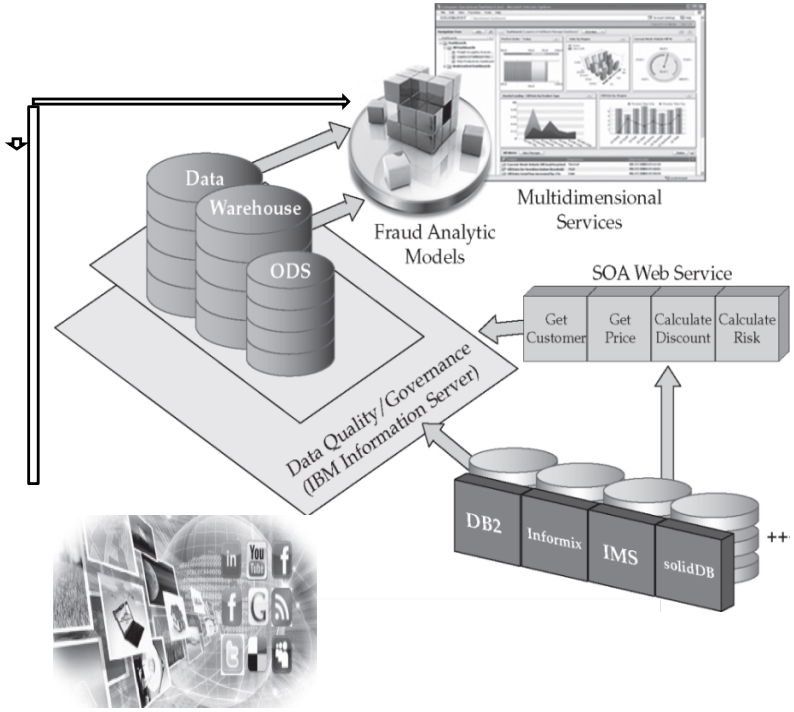
9th v: Vulnerability

10th v: Volatility

Usage of 80% of data (non-relational)



Classical model



New model

Characteristics of NoSQL Systems

- They do not use SQL
- They do not use relations
- They do not support relationships
- They do not support classical transaction properties ACID

- They do not have a formal schema
- They are designed for the support of web applications
- They are designed for the support of very big applications
- They are Open Source

Types of NoSQL Systems (a)

- DOCUMENT STORES
  - Examples: CouchDB, MongoDB

```
{
   “title” : “Foundations of Databases”
   “rating” : 10
}
```

Types of NoSQL Systems (b)

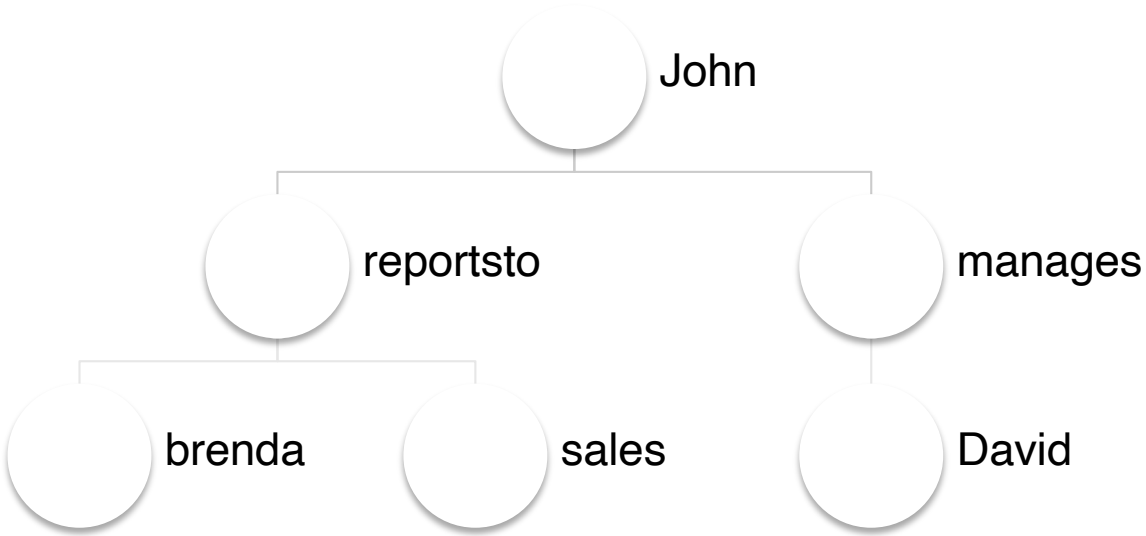
- KEY – VALUE Stores
  - Everything is stored in pairs (Key, Value)
  - Examples: Memcached, Riak, …

KEY	VALUE
Name1	bob
Course_44	DBMS
V32_phote	{binary data}
…	

Types of NoSQL Systems (c)

- **GRAPH DATABASES**
  - **Everything is stored as a graph**

    Examples: Neo4j, AlegroGraph, DB2 NoSQL, …



Why NoSQL???

- If you want a FLEXIBLE SCHEMA
- If you have HUGE AMOUNTS OF DATA
- **If you are NOT interested in data consistency so much!**

Cloud Computing with Databases

Google: BigTable, for internal storage

AppEngine: http://code.google.com/appengine/

Excellent for complex objects

Generic: Hadoop (Apache) - Open source for Cloud

Amazon:

S3	Big files http://aws.amazon.com/s3/
SimpleDB	Similar to BigTable http://aws.amazon.com/simpledb/
RDS	Service for relational data MySQL or Oracle 12g http://aws.amazon.com/rds/

Microsoft:

Azure	SQL Server http://www.microsoft.com/windowsazure/

Advantages of using cloud services for DBMS

- There are no standard costs
  - No cost for infrastructure or software
  - No maintenance
  - Easy management
- Pricing is depends on usage
  - Monthly cost proportional to the DB
  - Monthly cost based on data transfer
- Extensibility
  - Multiple distributed servers
  - Multiple high speed Internet connections
- Reliability
  - Distributed
  - Managed by specialists
  - Security

Limitations for cloud DBs

- Cost may become very high
- Then it may be better to buy in-house

Slide for extended discussion

Cloud Database Pricing

Example: Amazon RDS (MySQL), U.S. East

1 Extra large instance
20 hours/day
20 GB/month at 50 million I/O per month
10 GB/month data transfer in
500 GB/month data transfer out
20 GB/month regional transfer
=> $616 per month ($7400/year)

All values are estimates and might not include all fees.

Example: Microsoft SQLAzure Business Edition

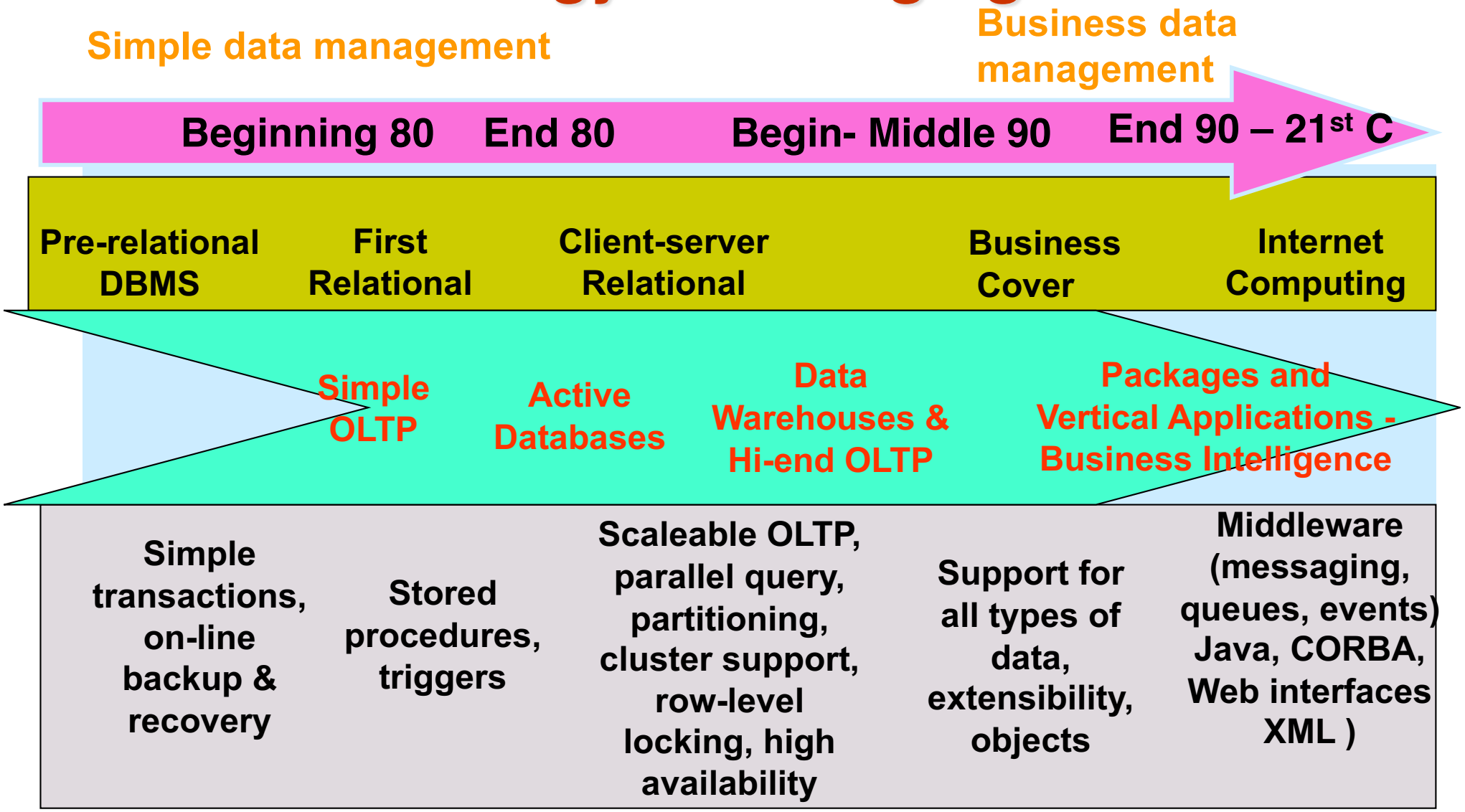
1 Extra large instance ($0.96/hour = $576/month)
20 GB/month ($200/month)
10 GB/month data transfer in ($1/month)
500 GB/month data transfer out ($75/month)
=> $852 per month ($10,224/year)

You get a relatively large database with T1-level data transfer for less than 10 percent of the cost of a DBA.

Cloud and Databases

"It was much nicer before people started storing all their data in the Cloud."

Evolution in the Application and Technology of Managing Data



Final Thoughts

- The DBMS is used for the management of Big data bases.
- Among the advantages is the recovery from system failures, fast application development, concurrency, integrity, security.
- Abstraction levels help in the independence of applications for physical entities.
- DB administrations have very important tasks and are usually well paid.
- DBMS R&D is one of the most attractive computer science domains (perfect combination of theory and practice)

End of Chapter 1