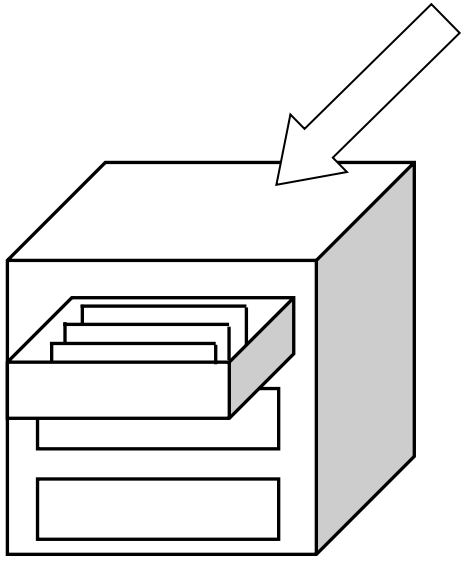
1. Fundamental concepts of database technology

1.1 An intuitive view of a database

What is a database? • **computerised record-keeping system**



MAR

ECO

Name: Economy section

Business: Economy affairs

Section manager: Jana Nováková

Office: A-216

Phone: 253

empNo	firstName	lastName	dOfBirth	address
30	Jana	Nováková	12.3.1950	Brno
58	Karel	Veselý	23.6.1958	Brno
80	Eva	Pokorná	3.9.1965	Blansko

- **Typical structure of a corresponding database**

SECTION

abbrev	name	business	manager	office	phone
ECO	Economy section	Economy affairs, ...	30	A-216	253
MAR	Marketing section	Marketing affairs, …	10	A-320	301
…	…	…			

EMPLOYEE

empNo	firstName	lastName	dOfBirth	address	section
...	...	...	...	...	...
...	...	...	...	...	...
10	Josef	Floryán	18.3.1945	Brno	MAR
...	...	...	...	...	...
30	Jana	Nováková	12.3.1950	Brno	ECO
...	...	...	...	...	...
58	Karel	Veselý	23.6.1958	Brno	ECO
...	...	...	...	...	...
80	Eva	Pokorná	3.9.1965	Blansko	ECO
...		...	...	...	...

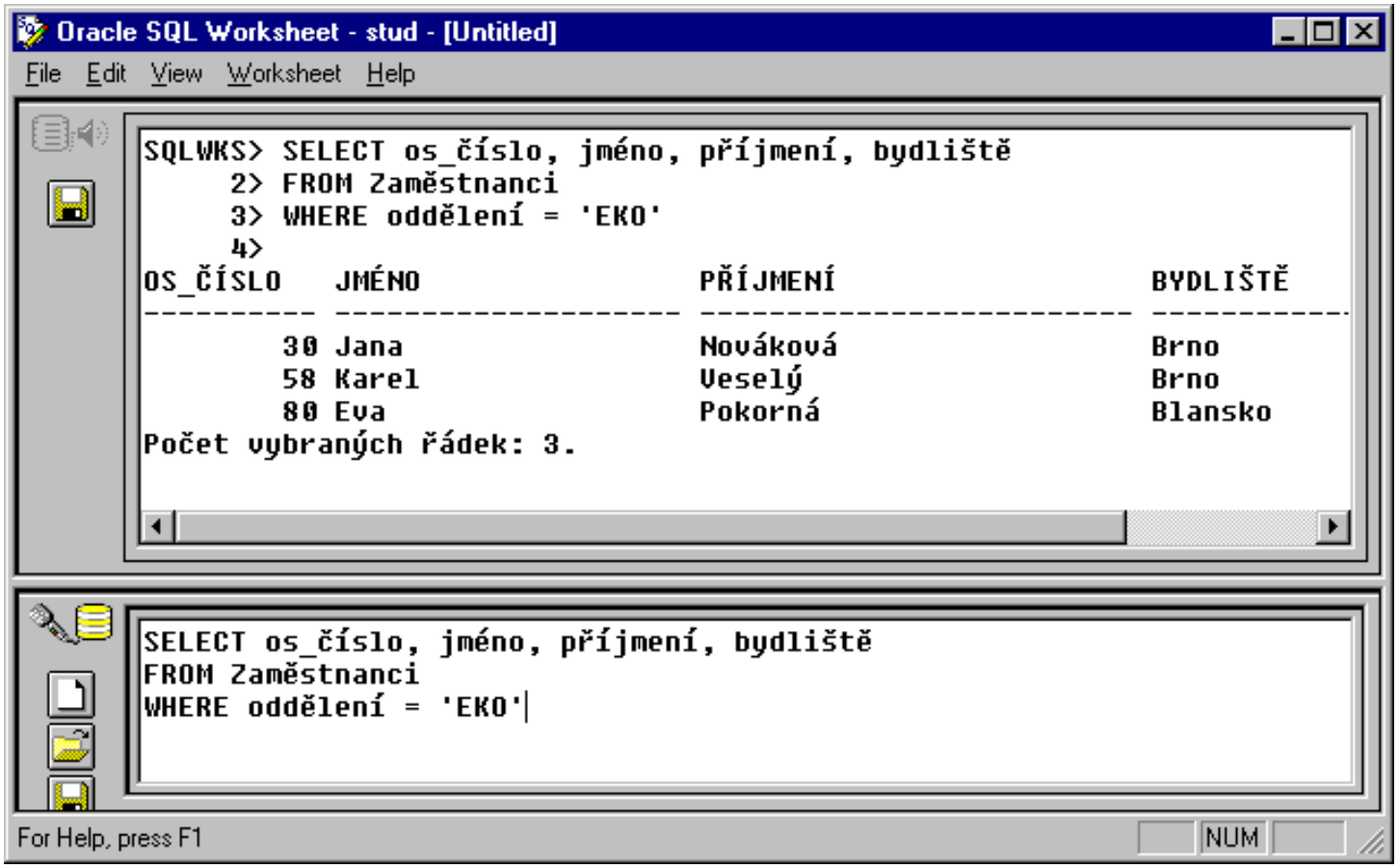
values from another column (a foreign key)

unique values (a primary key)

- **Typical operations with data in a database**
  - **Retrieval**

    “Which employees work in the Economy section?“

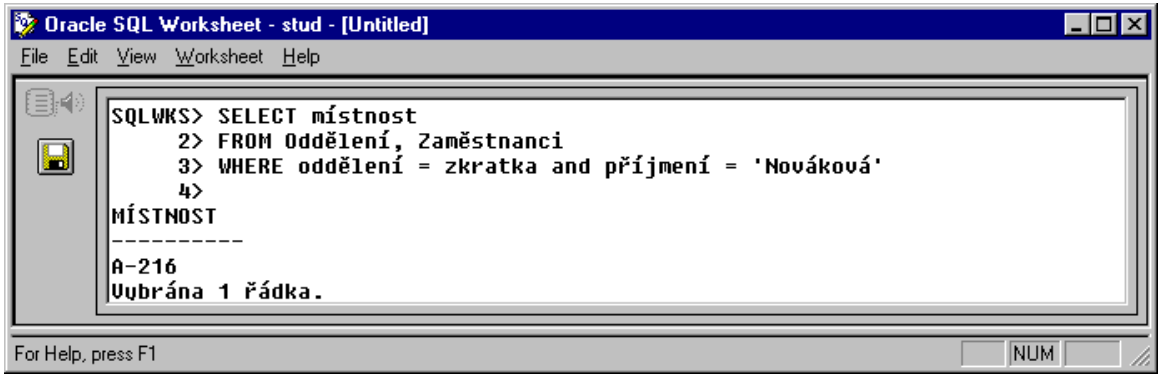
```
SELECT empNo, firstName, lastName, address
FROM Employee
WHERE section=’ECO’
```



- it is possible to join information from several tables

“What is the office where Mrs Nováková works?“

```
SELECT office
FROM Employee, Section
WHERE section=abbrev AND lastName='Nováková'
```



- **Insertion**

```
INSERT INTO Section
VALUES ('PER', 'Personnel section', 'Personnel
       affairs', NULL, 'B-020', 125)
```

- **Modification**

```
UPDATE Employee
SET section = 'PER'
WHERE empNo = 58
```

- **Deletition**

```
DELETE FROM Employee WHERE empNo = 58
```

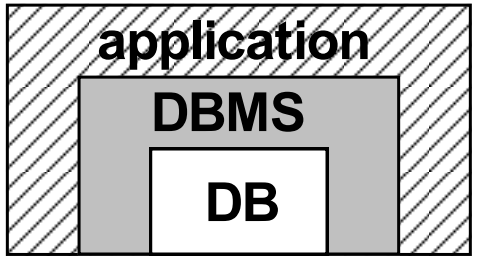
1.2 Fundamental concepts

Database **(DB) - collection of (*structured*) *persistent* data used by application systems of some enterprise or institution.**

Additional properties:

- ***integrated*** **- can be thought of as a unification of several otherwise distinct data files with any redundancy eliminated (wholly or partly)**
- ***shared*** **- usually a multi-user access (concurrent), the access of a particular user can be restricted to only a part of the database**
- ***secure*** **- security restrictions can be applied, integrity can be maintained.**

Database management system - a program layer performing operations over a DB



- users (applications) are isolated from hardware-level technical details

- operations: create database, create table, retrieve, insert, ...

Database system **(DBS) - a system that includes:**

- **hardware**
- **persistent data - DB**
- **software - DBMS, applications, development tools, design aids, generators, ...**
- **users**

DBS (in a strict sense) = DB + DBMS

1.3 Abstractions on database data

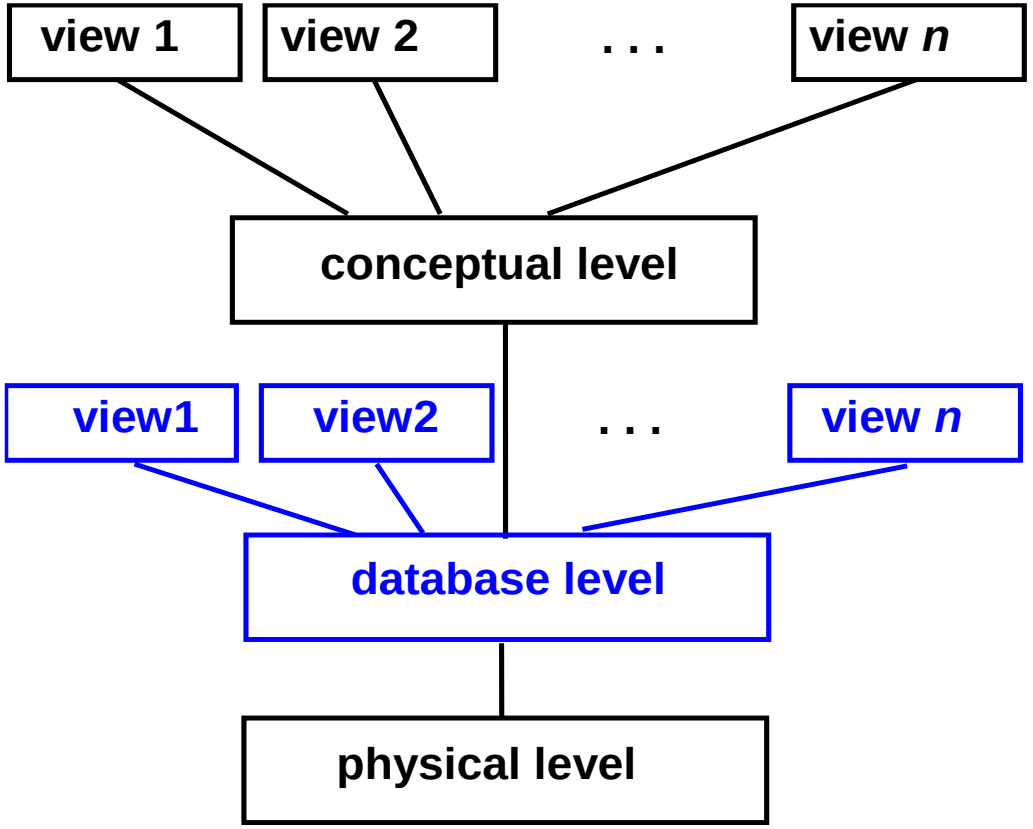
One of important objectives of DBSs is to provide users with abstract views of data in the database (details of storing and management are hidden).

real world

← **several levels of abstraction**

stored data

- **Basic levels of database data abstraction:**
  - ***physical (internal) level*** **– describes data, *how* it is really stored.**
  - ***conceptual (logical) level*** **- describes *what* data is really stored in the database and what are relationships among them.**
  - ***level of views (external)*** **– describes what data is seen by individual roles of users (in general, they see only part of the database that represents real-world objects they see).**



Real-world objects
(Person)

Logical representation
(type Person)

DB views
(portion of table Person)

DB logical structure
(table Person)

Physical storing of data
(database segment)

1.4 Data models

Data model **- a collection of concepts and tools used for description of data, relationships between data elements, semantics of data, logical structure of data, integrity rules etc.**

Can be classified into two categories:

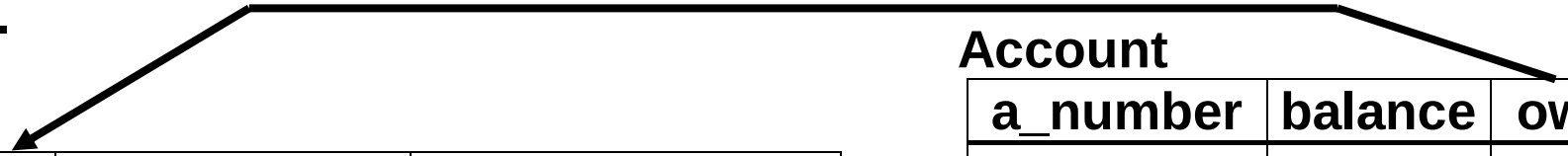
a) *logical models* - describe data at the external and conceptual levels

- ***object-based*** **models (ER, OO, functional model, ...)**
- ***database*** **models - describe logical structure of the database, integrity rules and possible manipulations (relational, network, hierarchical, ...)**

b) physical models - describe data at the internal level

Examples of database models

Relational model **- data and relationships among them are represented by a collection of tables, no other logical structure is visible by the user.**



Customer

c_number	name	address
100	Jan Novak	Cejl 8, Brno
110	Pavla Zelena	Hajkova 1, Brno
200	Jan Novák	Cejl 8, Brno

Account

a_number	balance	owner
800	10000	100
557	50000	200
500	3000	100
486	2000	110

Network model **- data are represented by collections of records and links (pointers).**

Hierarchical model **- similar to the network model, but records are organised as collections of trees rather than more general graphs (networks)**

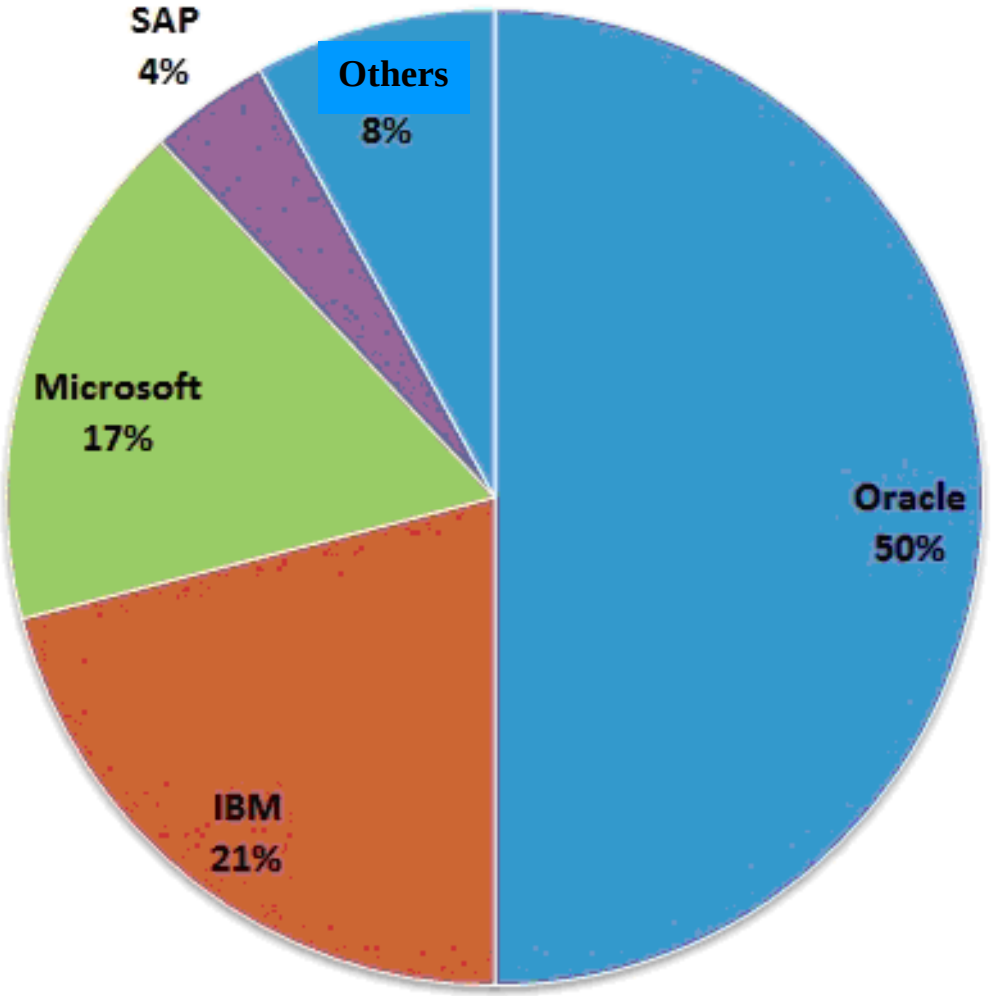
Object-oriented model **– includes all fundamental concepts of OO (object types, encapsulation, inheritance, polymorphism, ...) applied to persistent data**

and others**.**

Ex) Relational (currently mostly object-relational) DBMSs:

Oracle 11g, DB2 (IBM), Informix (IBM), Microsoft SQL Server, Sybase (SAP), MySQL (Oracle), PostgreSQL

Database market



1.5 Database schema

Database schema **- metainformation describing data in the database.**

We can distinguish two levels of database schemas:

- ***Logical schema*** **– describes data at a logical level (e.g. tables in relational database)**
- ***Physical schema*** **– describes data at a physical level (organization in files, access methods etc.).**

1.6 Data independence

- immunity of applications to changes in logical or physical schema.

Physical independence **- ability to modify the internal schema without the need to modify application programs.**

Logical independence **- ability to modify the conceptual schema without the need to modify application programs.**

Comments**:**

- it is one of basic advantages of database technology

- it is easier to achieve physical independence than logical one

1.7 Database languages

Every database language must allow us to:

a) specify a database scheme ⇒ DDL - *Data Definition Language*
(general name for such sublanguage, it does not correspond to any specific language)

The result of compilation is a set of objects (e.g.tables) and information in a system catalogue.

b) manipulate database data ⇒ DML - *Data Manipulation Language*
(retrieve, insert, delete, modify database data)
There are two types of manipulation languages:

- ***Procedural* - *WHAT* data is needed and HOW to get it.**
- ***Nonprocedural* - *WHAT* data is needed.**

Nonprocedural languages are usually easier to learn and use but generated code may be less efficient (optimisation needed).

c)*data integrity and security ⇒ DCL - *Data Control Language

d) transaction control, ..

Access to a database from applications (what languages we use for programming):

- Using specialized (native) database languages – extend SQL by other data types and statements.

  Ex) PL/SQL (Oracle), Transact SQL (Microsoft SQL Server, Sybase Adaptive Server Enterprise), Informix 4GL, …

- Using general programming languages:
  - Together with libraries and APIs for access to a database

    Ex) native (OCI – Oracle Call Interface, ADO – ActiveX Data Objects,…), standardized (ODBC – Open DataBase Connectivity, JDBC – Java DataBase Connectivity)
  - Together with possibility to embed SQL statements to the host source code

    Ex) Embedded SQL for Pascal, C, …, SQLJ
  - Object- relational mapping (ORM, O/RM) – allows applications to see data in a relational database as objects

    Ex) Hibernate framework

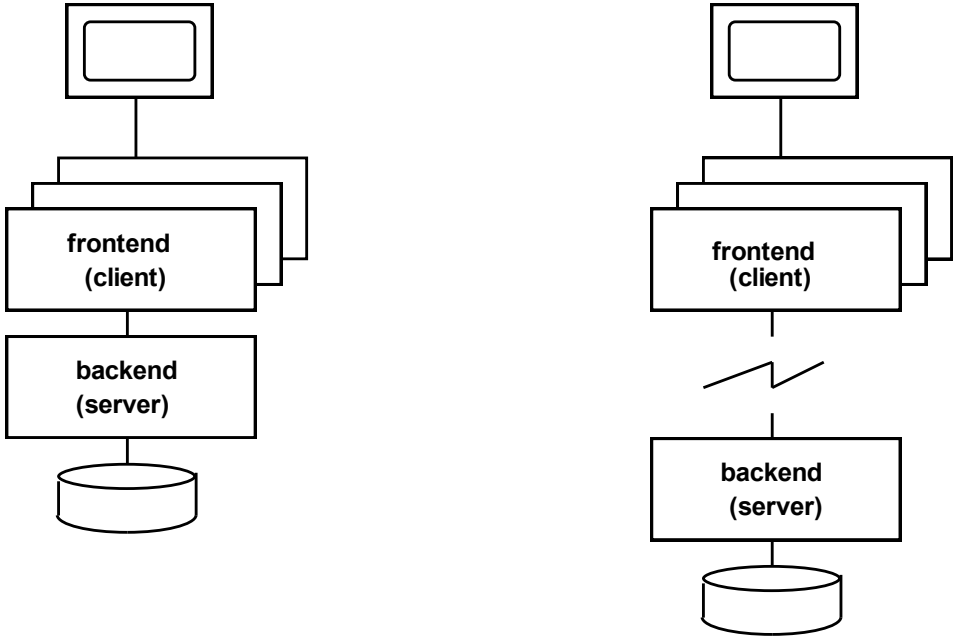
1.8 Database users

- ***Database administrator*** **- has central responsibility for the data (logical and physical scheme definition, scheme modifications, access rights, performance tuning, backup the data etc.)**
- ***Application programmers*** **- develop applications using development tools and a database language.**
- ***Knowing users*** **- do not write programs, but are able to write an unplanned request in a query language.**
- ***Naive users*** **- use only applications (planned requests)**

1.9 Architectures of database systems and applications

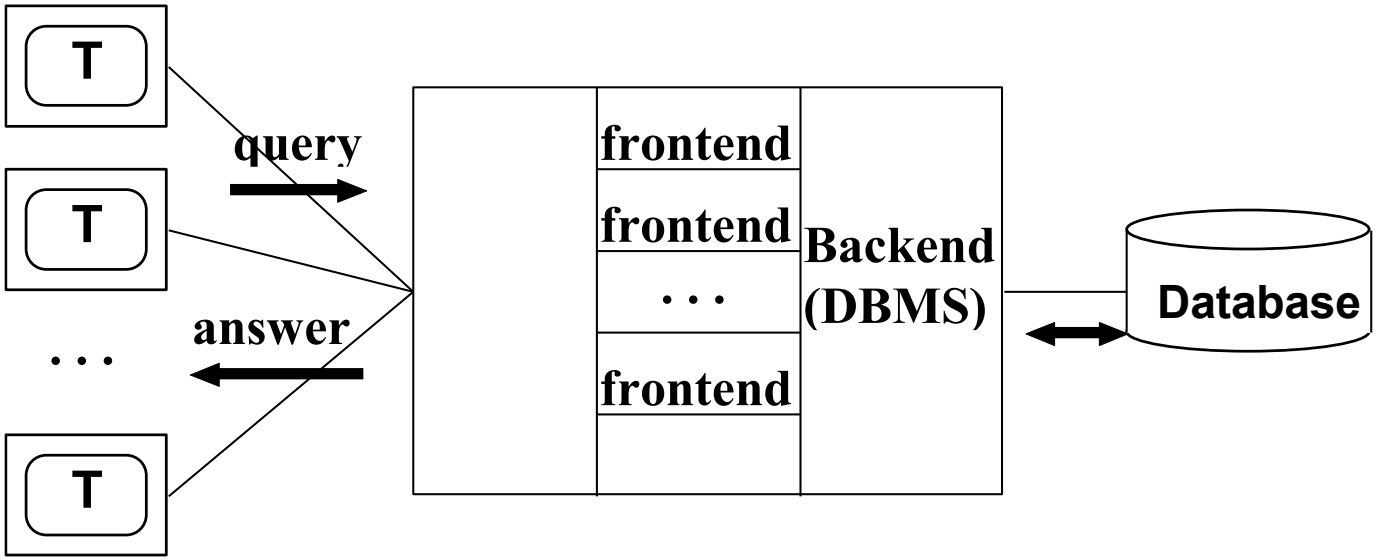
Each database application together with a DBMS contains the following two programming layers:

- ***frontend*** **– the part of an application running on top of the DBMS (it uses services of the DBMS)**
- ***backend*** **– processes that support all basic DBMS functions and application-oriented functionality managed by the DBMS**

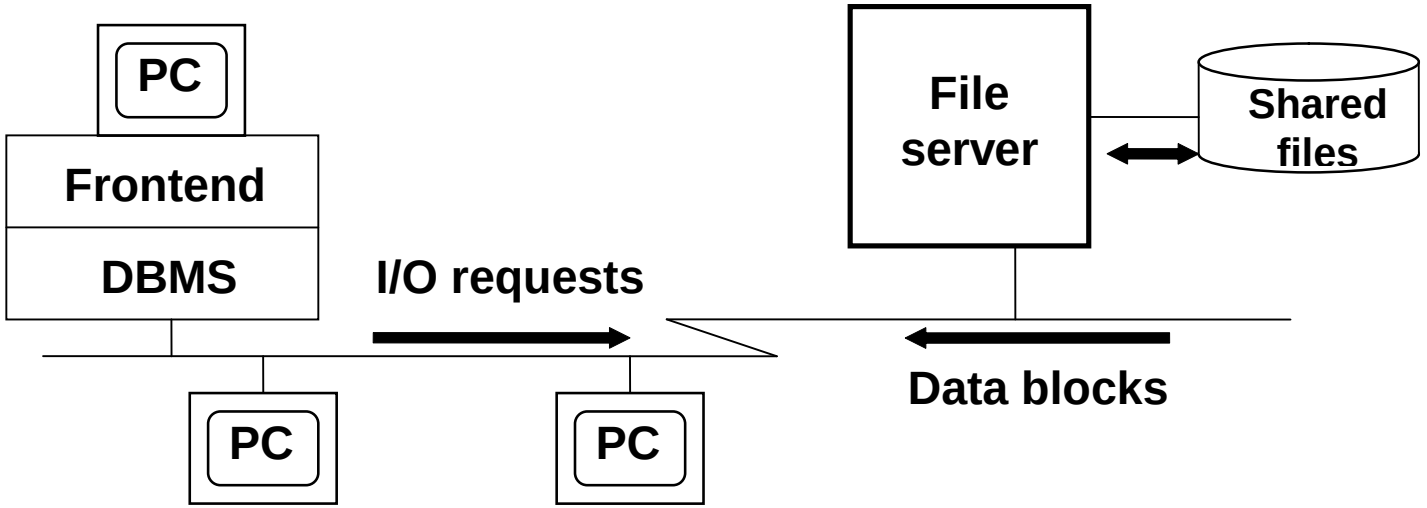


Historical view

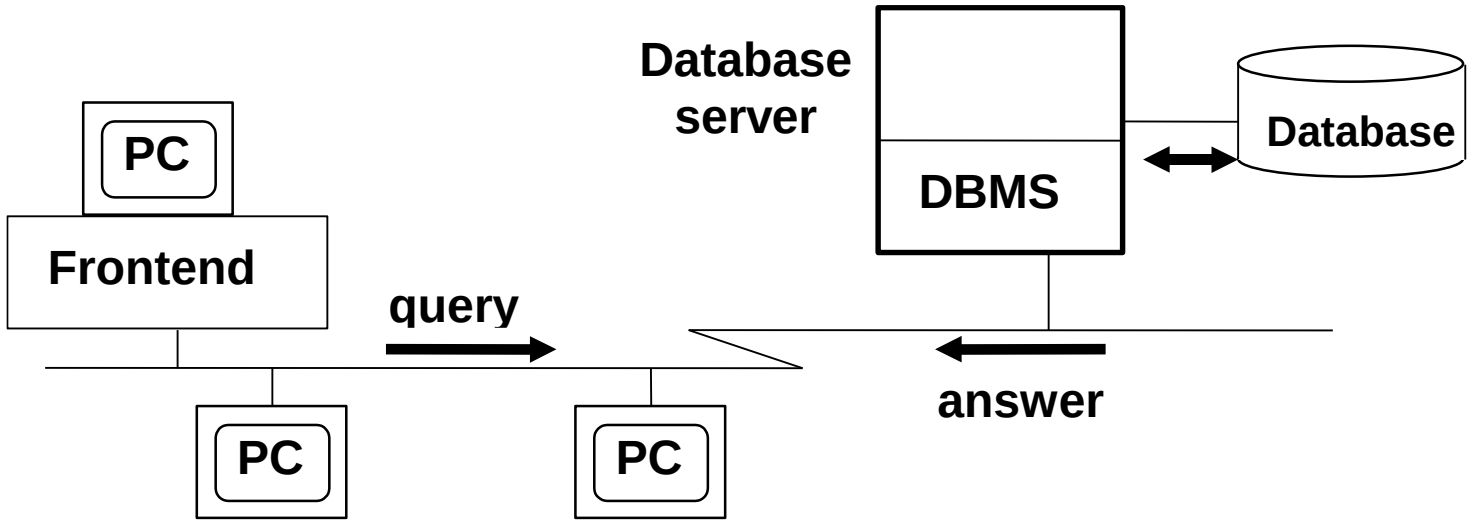
- centralised architecture (′mainframe based′)



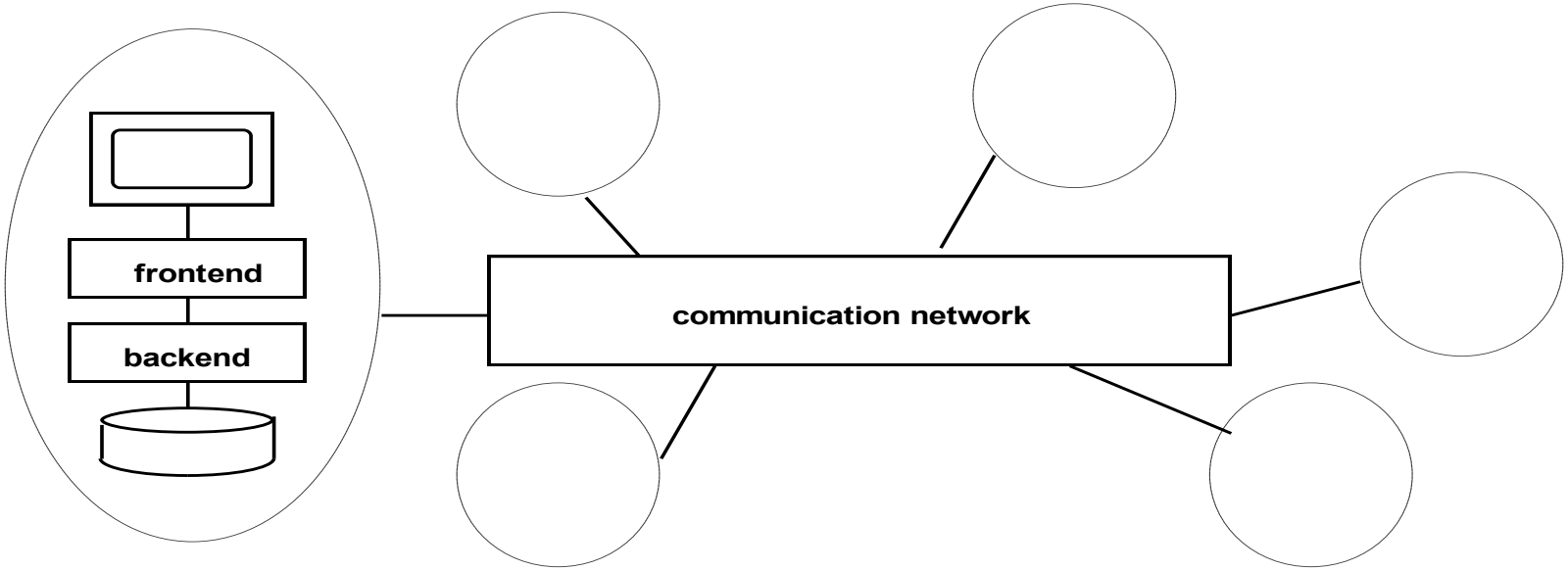
- file server architecture



- **client/server**



- **distributed DBS**



1.10 Types of database systems

According to a data model which the database is based on:

- ***prerelational*** **(hierarchical, network) - IMS, IDMS**
- ***relational*** **- DB2, Oracle, Ingres, Informix, SQLBase, dBASE IV, ..**
- ***postrelational*** **(OO, OR, deductive, temporal, ..., NoSQL) - O2, ObjectStore, databases supporting SQL:1999, ...**

Most of existing commercial systems are relational systems. The fundamental features of relational databases are:

1. the data is perceived by the user as collections of tables

2. operations over the tables result in tables, too

According to architecture:

- ***Desktop*** **- oriented toward single-user applications and reside on standard personal computers (e.g. Microsoft Access, FoxPro, ...)**
- **Server – oriented toward multi-user, client/server and multi-layer applications (Oracle, IBM DB2, Microsoft server, MySQL, ...)**

Bibliography

1. Silberschatz, A., Korth H.F, Sudarshan, S.:Database System Concepts. Fourth Edition. McGRAW-HILL. 2002, pp. 1 - 24.