Technische Hochschule Rosenheim

Chapter 1 – Introduction

Databases lectures

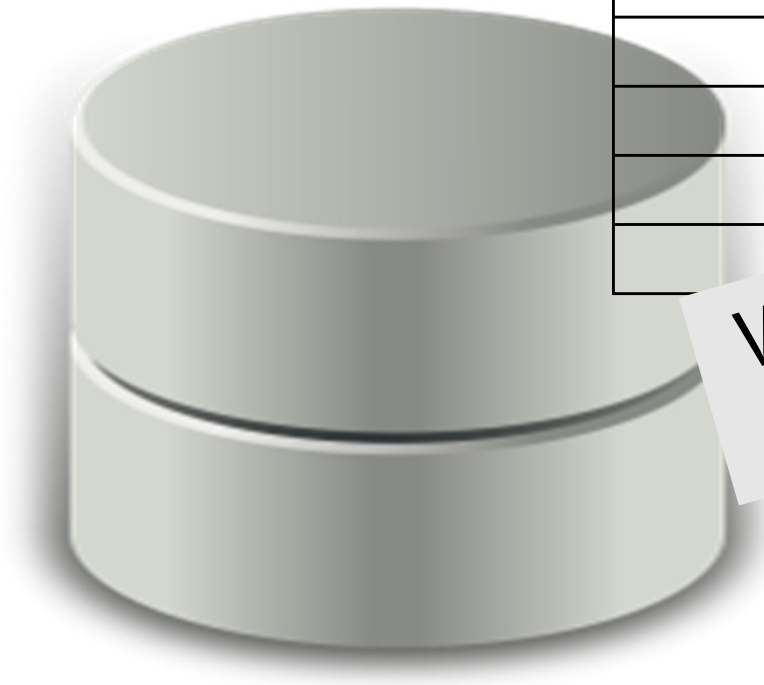
Dr Kai Höfig

Why databases?

So what is a database?

- Database = quantity of logically grouped, structured information units

Martin	Müller	Haupstraße 1	Rosenheim	08031-99999999



What are the general benefits of a database?

What is the easiest way to store data? Flat file

Advantages of the database compared to “storage on paper”

- Search for/filter information
  - Everywhere or only with certain attributes
  - Examples
    - All data sets (records) with the first name “Klaus”
    - All people with a birthday in October
- Sort by various attributes
  - Examples
    - Sort by first name in ascending order (A-Z)
    - Sort by name in descending order (Z-A)
- …and many more operations!
- Security against loss of data
- Long-term storage

Some terms using a simple table as an example

We call a table a **relation**

Table headings of a column are called **attributes.** These have **data types** e.g. number, string or date

First_name	Name	Street	Postcode	Place	Date_of_birth	Telephone
Hans	Huber	Hofstr. 4	83024	Rosenheim	10.10.1990	012-3456
Peter	Petersen	Parkweg 3	83024	Rosenheim	9.9.1965	9999-9999
Susi	Sorglos	Sandstr 2	80801	Munich	1.1.1991	0177-7777
Andrea	Ammer	Am Bach 1	88888	Ambach	21.12.1989	176462
Klaus	Klammer	Klarastr. 9	83646	Bad Tölz	4.4.1984	08041-4444
Mark	Markl	Marktstr. 2	83646	Bad Tölz	3.3.1983	08041-3333
Gabi	Genau	Giselastr. 5	81888	Munich	8.8.1988	089-8888

The content of a cell (or field) is called an **attribute value.**

A row is a data set (or record) and is called a **tuple**

Why do we need a data model instead of using lists?

Disadvantages of flat file databases

Inconsistency: For the same object in reality there is different (redundant) data in the database.

Data redundancy: The same data is saved multiple times.

First_name	Name	Street	Postcode	Place	Date_of_birth	Telephone
Hans	Huber	Hofstr. 4	83024	Rosenheim	10.10.1990	012-3456
Peter	Petersen	Hofstraße 4	83024	Rosenheim	9.9.1965	9999-9999
Susi	Sorglos	Sandstr 2	80801	Munich	1.1.1991	0177-7777
Andrea	Ammer	Am Bach 1	88888	Ambach	21.12.1989	176462
Klaus	Klammer	Klarastr. 9	83646	Bad Tölz	4.4.1984	08041-4444
Mark	Markl	Marktstr. 2	83646	Bad Tölz	3.3.1983	08041-3333
Gabi	Genau	Giselastr. 5	81888	Munich	8.8.1988	089-8888

Insertion anomaly: Just saving a new street with a postcode on its own makes no sense here.

Update anomaly: When changing Hofstraße, inconsistencies can easily arise because this is stored redundantly.

Deletion anomaly: If Andrea is deleted, I also lose the postcode for the street.

Relations resolve these disadvantages

First_name	Name	Date_of_birth	Telephone
Hans	Huber	10.10.1990	012-3456
Peter	Petersen	9.9.1965	9999-9999
Susi	Sorglos	1.1.1991	0177-7777

Street	Postcode	Place
Hofstraße 4	83024	Rosenheim
Sandstr 2	80801	Munich

The designation of Hofstraße is always uniform, since it is only saved once.
The city of Rosenheim only needs to be saved once
Streets can be added without anyone having to live there
People can be deleted without losing information about the postcode

But:

Multiple phone numbers still cannot be saved for one person. The success of systems is therefore linked to the skilful design of such database relations and to the respective domain. More on this later.

Summary: Challenges of (relational) flat file databases

- Inconsistencies due to redundant storage of the same identity with different attribute values.
- Forgetting changes and the resulting inconsistencies (update anomaly).
- No central, “standardised” data storage system: each individual application must know the internal representation of the data and the location where it is stored
- Each individual application must be optimised for efficient data processing
- Data is often more valuable and durable than applications, but lives “inside” the application
- Multiple users or applications cannot work on the same data in parallel without interfering with each other
- Data protection and data security are not ensured
- Waste of storage space
- Insertion anomaly
- Update anomaly
- Deletion anomaly

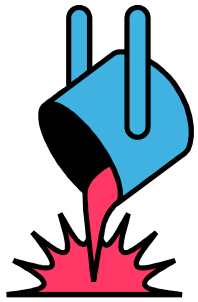
Objectives and challenges of DBS

- Efficient management of very large volumes of data
  ➔ How can huge volumes of data (terabytes) be processed efficiently?
- Parallel access of multiple users to the data
  ➔ How can many users (>10,000) work with the data at the same time?
- Ensuring data independence
  ➔ How do you organise (model and use) data?
- Ensuring data protection and data security
  ➔ How is data stored reliably in the long term?
  ➔ How do you control the access?

Is it good to have one data model?

Schema architecture introduction

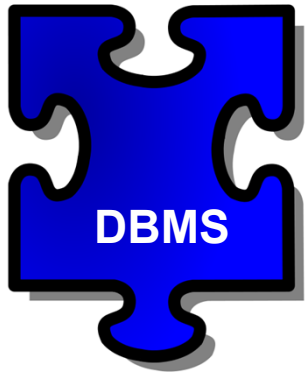
- At the different levels of a database-based system, the data is available in different representations and extracts.
- This type of purpose-related form of data representation is called a **schema**, which does not mean the data itself (**instance**).
- Example: the manufacturer Xoco produces many types of chocolate bars from individual ingredients (beans, butter, sugar, etc.).
- What does the data look like in the individual schemas?



Production system Ordering system

external schemas

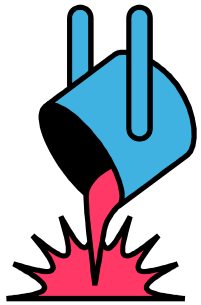
Conceptual schemas



Internal schemas

External schemas

- (External) **schema** for the production system



Name	Ingredients	Temperature	Stirring_time
Acapulco dark	30% Criollo Venezuela 20% Nacional Ecuador 48% sugar 2% vanilla	76 degrees	90hrs
Criollo Venezuela 80%	80% Criollo Venezuela 20% sugar	77 degrees	80hrs
Dark milk	35% Arriba Venezuela 65% sugar	78 degrees	30hrs

- (External) **schema** for the ordering system

Name	Cocoa	Price	Delivery_time
Acapulco dark	50%	2.22 EUR	24hrs
Criollo Venezuela 80%	80%	5.30 EUR	48hrs
Dark milk	35%	1.50 EUR	12hrs

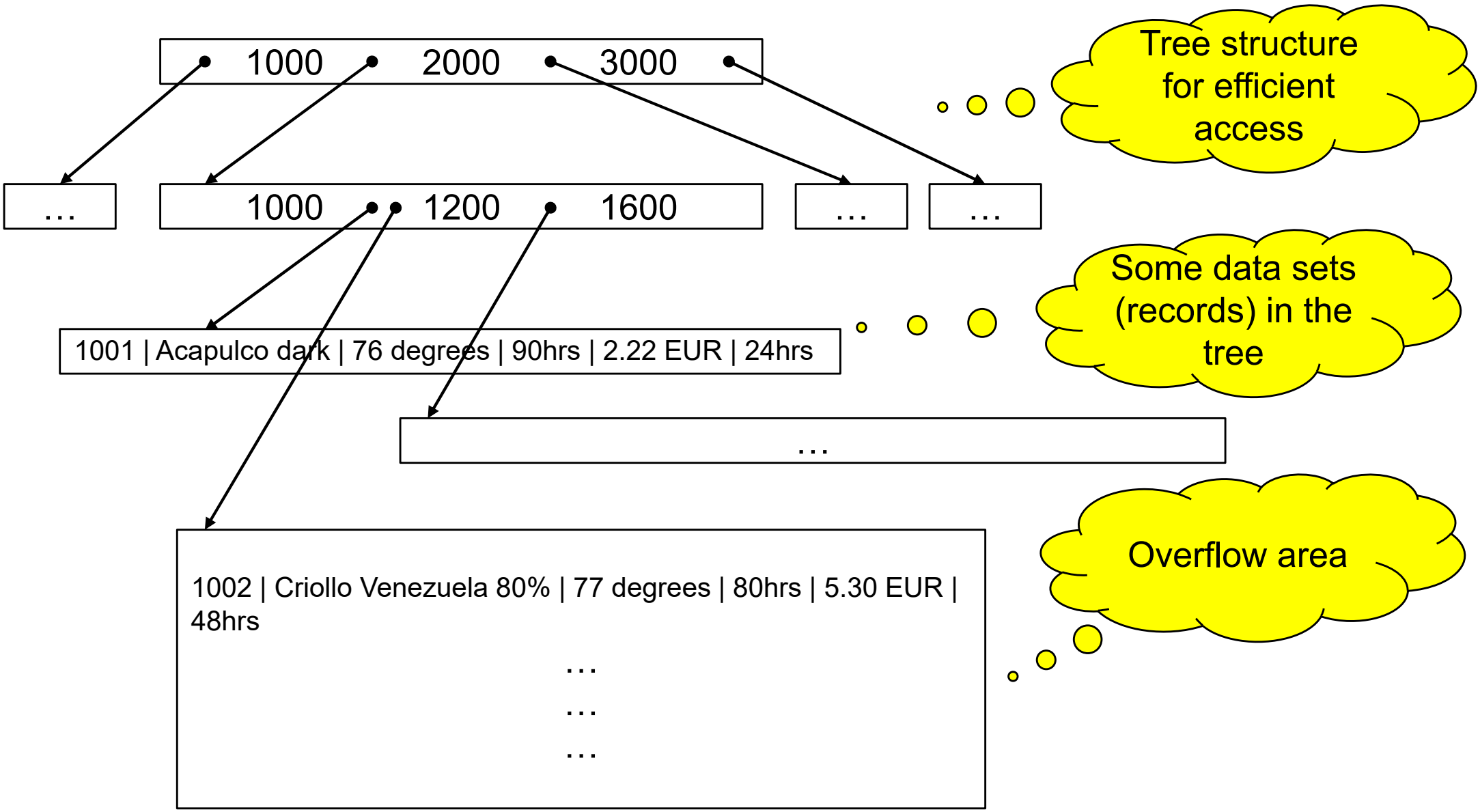
Conceptual schema – representation in tables (relations)

SNo.	Name	Temperature	Stirring_ time	Price	Delivery_ time
1001	Acapulco dark	76 degrees	90hrs	2.22 EUR	24hrs
1002	Criollo Venezuela 80%	77 degrees	80hrs	5.30 EUR	48hrs
1003	Dark milk	78 degrees	30hrs	1.50 EUR	12hrs

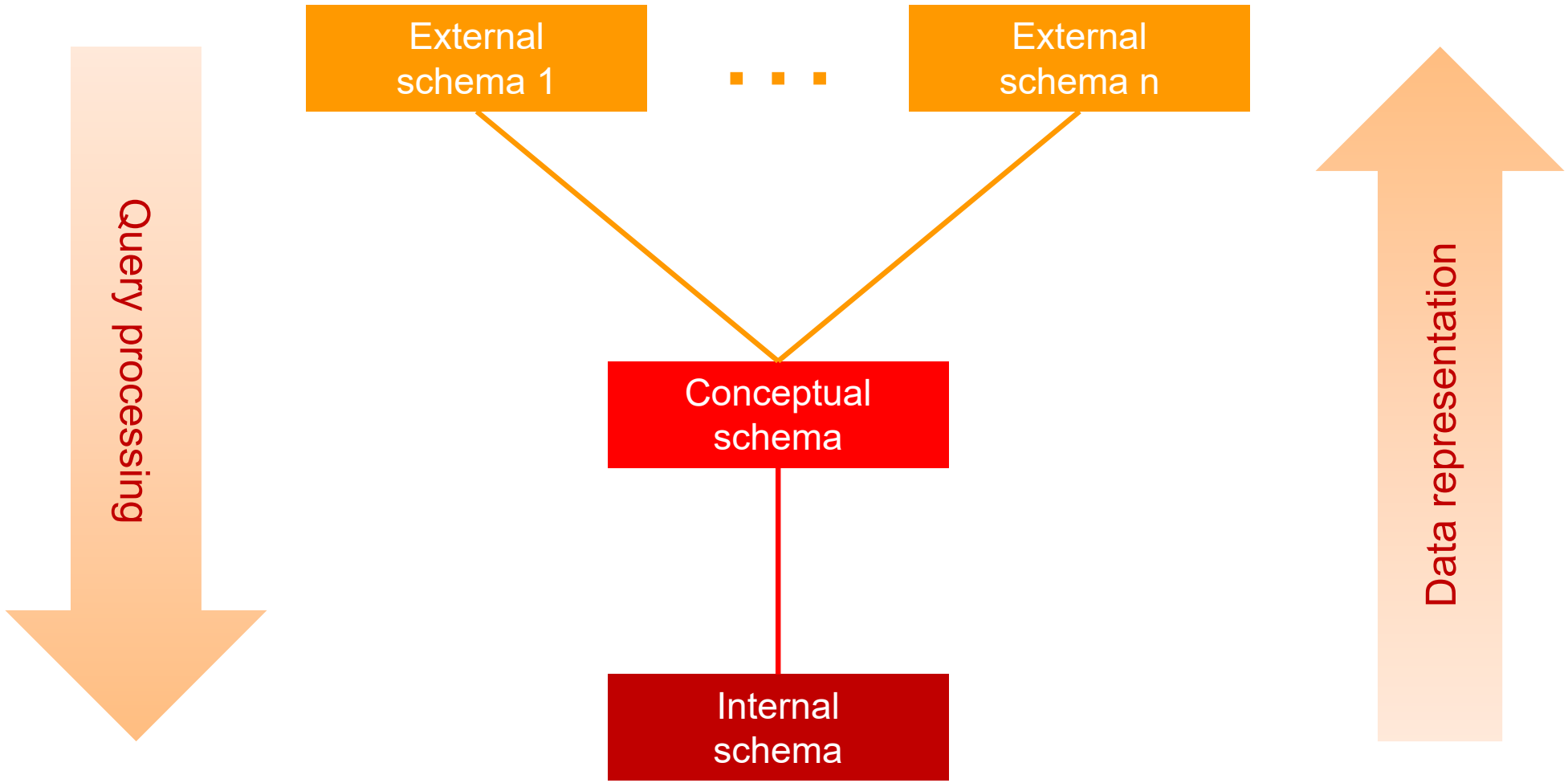
ANo.	Name	IsCocoa
2001	Criollo Venezuela	True
2002	Nacional Ecuador	True
2003	Arriba Venezuela	True
2004	Vanilla	False
2005	Sugar	False

SNo.	ANo.	Proportion
1001	2001	30%
1001	2002	20%
1001	2005	48%
1001	2004	2%
1002	2001	80%
1002	2005	20%
1003	2003	35%
1003	2005	65%

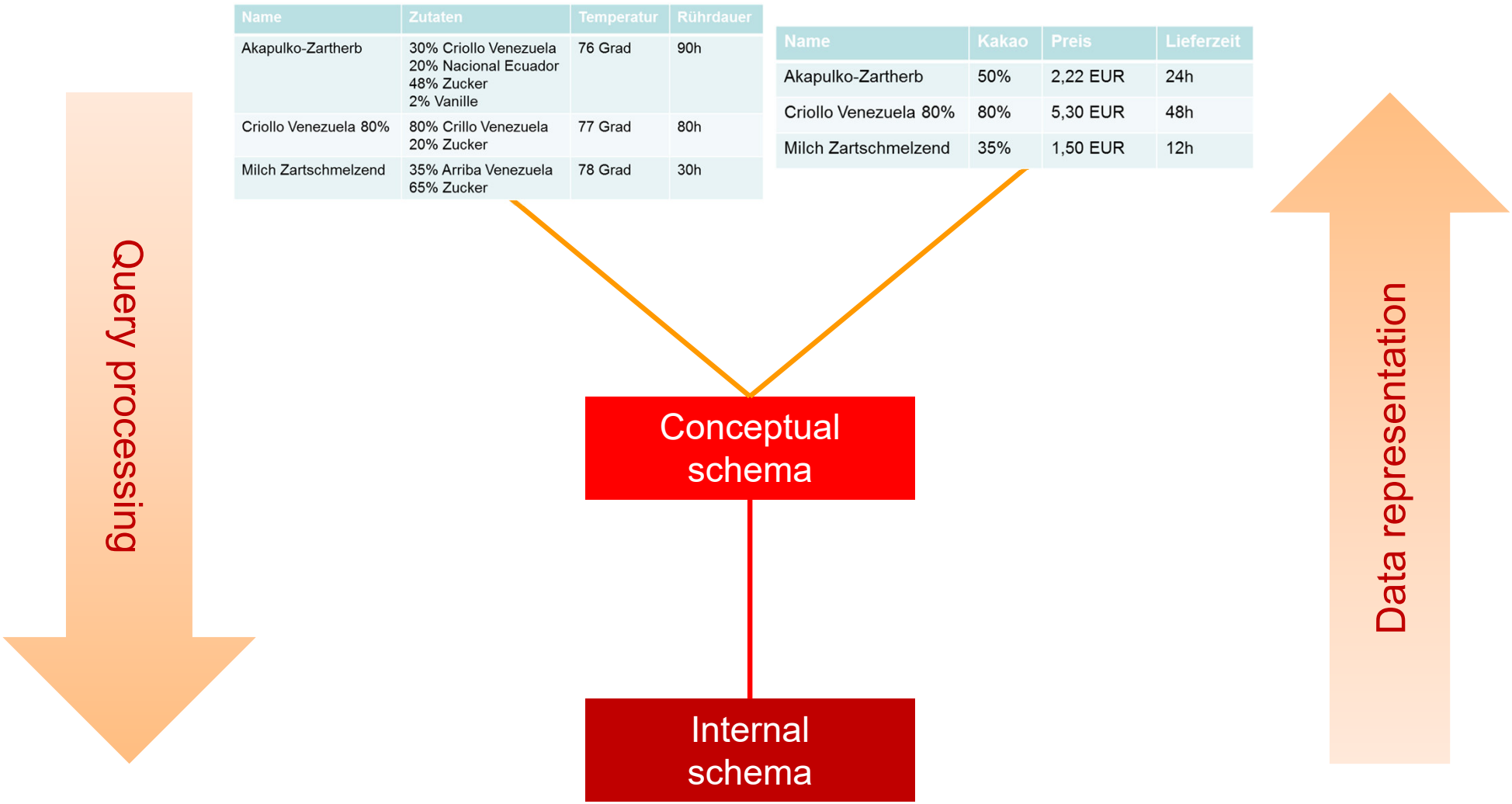
Internal schema – internal organisation of the data



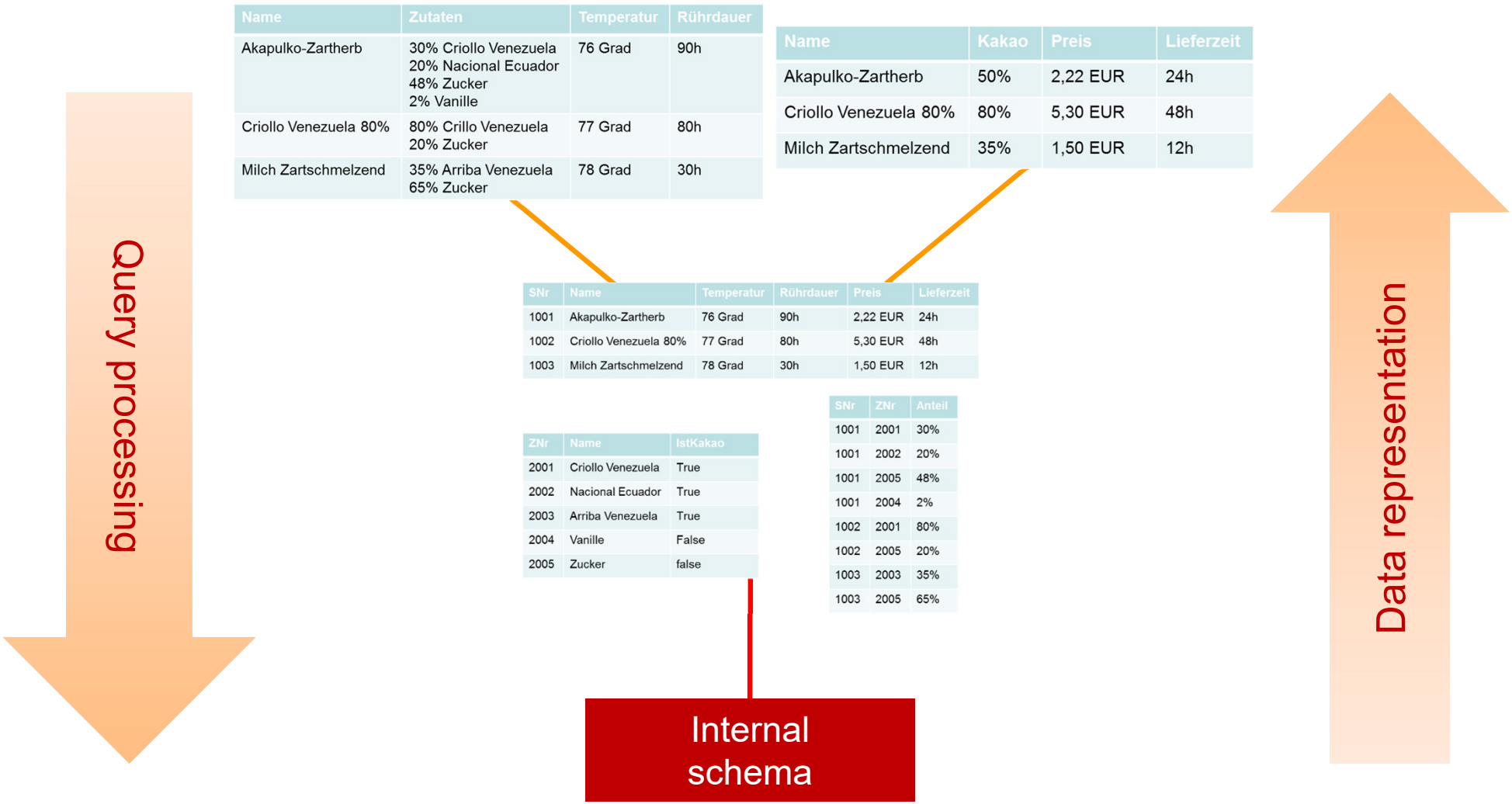
Schema architecture - general



Schema architecture - general



Schema architecture - general



Name	Zutaten	Temperatur	Rührdauer
Akapulko-Zartherb	30% Criollo Venezuela 20% Nacional Ecuador 48% Zucker 2% Vanille	76 Grad	90h
Criollo Venezuela 80%	80% Crillo Venezuela 20% Zucker	77 Grad	80h
Milch Zartschmelzend	35% Arriba Venezuela 65% Zucker	78 Grad	30h

Name	Kakao	Preis	Lieferzeit
Akapulko-Zartherb	50%	2,22 EUR	24h
Criollo Venezuela 80%	80%	5,30 EUR	48h
Milch Zartschmelzend	35%	1,50 EUR	12h

SNr	Name	Temperatur	Rührdauer	Preis	Lieferzeit
1001	Akapulko-Zartherb	76 Grad	90h	2,22 EUR	24h
1002	Criollo Venezuela 80%	77 Grad	80h	5,30 EUR	48h
1003	Milch Zartschmelzend	78 Grad	30h	1,50 EUR	12h

ZNr	Name	IstKakao
2001	Criollo Venezuela	True
2002	Nacional Ecuador	True
2003	Arriba Venezuela	True
2004	Vanille	False
2005	Zucker	false

SNr	ZNr	Anteil
1001	2001	30%
1001	2002	20%
1001	2005	48%
1001	2004	2%
1002	2001	80%
1002	2005	20%
1003	2003	35%
1003	2005	65%

Schema architecture – general – with example

Query processing

Data representation

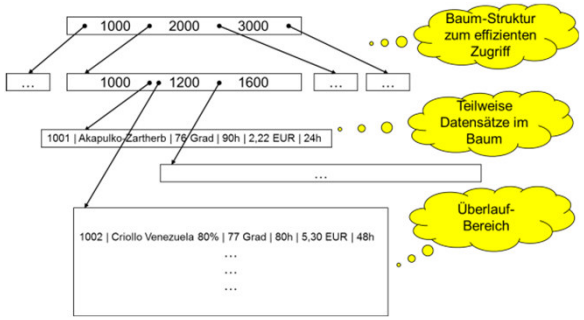
Name	Zutaten	Temperatur	Rührdauer
Akapulko-Zartherb	30% Criollo Venezuela 20% Nacional Ecuador 48% Zucker 2% Vanille	76 Grad	90h
Criollo Venezuela 80%	80% Crillo Venezuela 20% Zucker	77 Grad	80h
Milch Zartschmelzend	35% Arriba Venezuela 65% Zucker	78 Grad	30h

Name	Kakao	Preis	Lieferzeit
Akapulko-Zartherb	50%	2,22 EUR	24h
Criollo Venezuela 80%	80%	5,30 EUR	48h
Milch Zartschmelzend	35%	1,50 EUR	12h

SNr	Name	Temperatur	Rührdauer	Preis	Lieferzeit
1001	Akapulko-Zartherb	76 Grad	90h	2,22 EUR	24h
1002	Criollo Venezuela 80%	77 Grad	80h	5,30 EUR	48h
1003	Milch Zartschmelzend	78 Grad	30h	1,50 EUR	12h

ZNr	Name	IstKakao
2001	Criollo Venezuela	True
2002	Nacional Ecuador	True
2003	Arriba Venezuela	True
2004	Vanille	False
2005	Zucker	false

SNr	ZNr	Anteil
1001	2001	30%
1001	2002	20%
1001	2005	48%
1001	2004	2%
1002	2001	80%
1002	2005	20%
1003	2003	35%
1003	2005	65%



Schema architecture

- Separation of schema – instance
  - **Schema** (metadata, data descriptions)
  - **Instance** (user data, database status or version)
- **Schema architecture** describes the relationship between
  - conceptual schema (result of data definition)
  - internal schema (definition of file organisations and access paths)
  - external schema (result of view definition)
  - application programmes (result of application programming)
- **Database schema**: internal + conceptual + external schemas
- DBMS generally have two types of commands (“languages”)
  - DDL (Data Definition Language) → changes to schema (structure)
  - DML (Data Manipulation Language) → changes to content (data sets)

Data independence. Why?

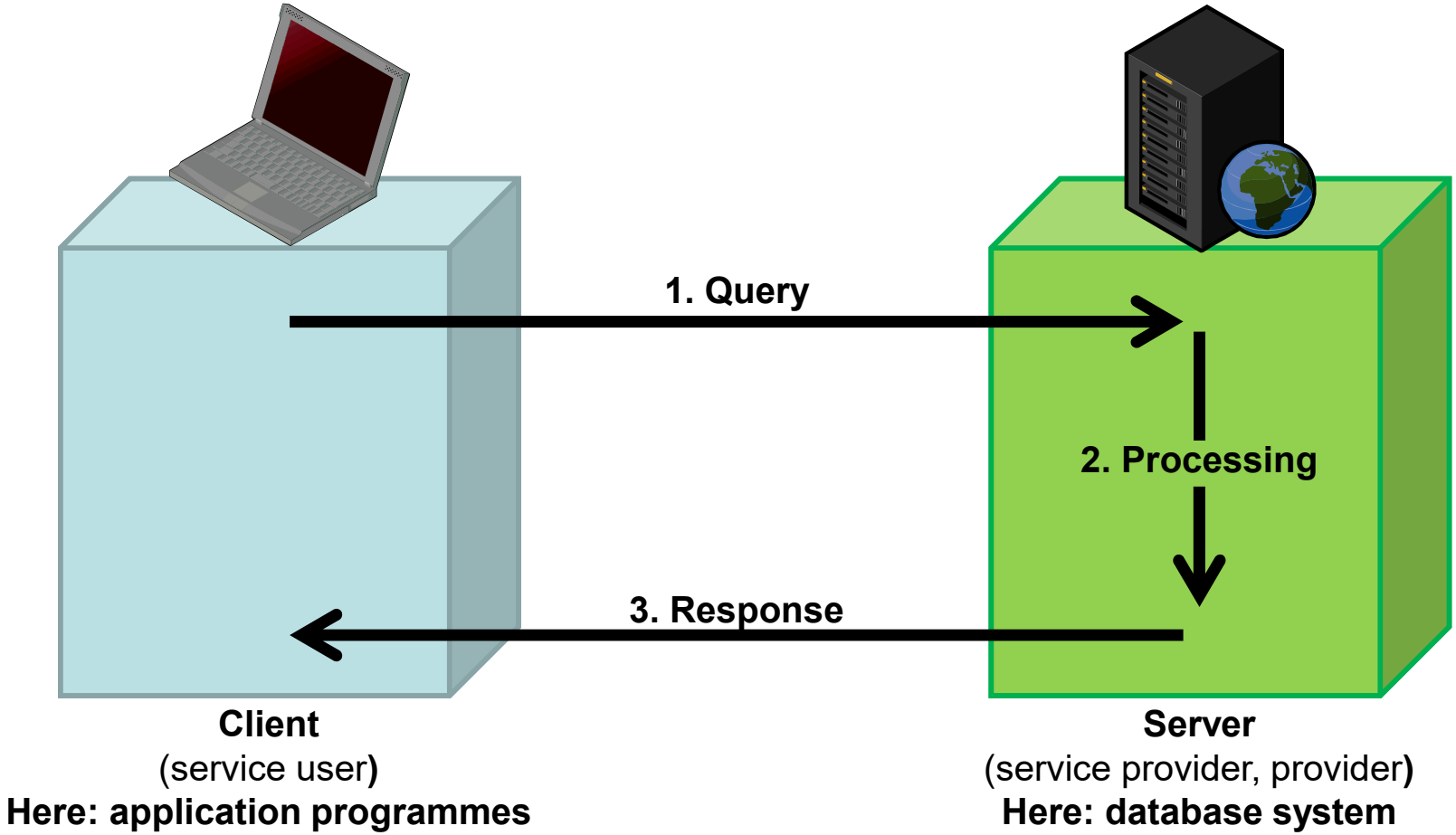
Data independence

- Stability of the user interface against changes
  - physical: changes to file organisations and access paths do not have any effect on the conceptual schema (access, location, scaling and migration transparency)
  - logical: changes to the conceptual schema and certain external schemas do not have any effect on other external schemas and other application programmes

- Possible impact of changes to the conceptual schema:
  - External schemas may be affected (change of attributes)
  - Application programmes may be affected (recompiling the application programmes, changes may be necessary)
  - However, required changes are detected and monitored by the DBMS

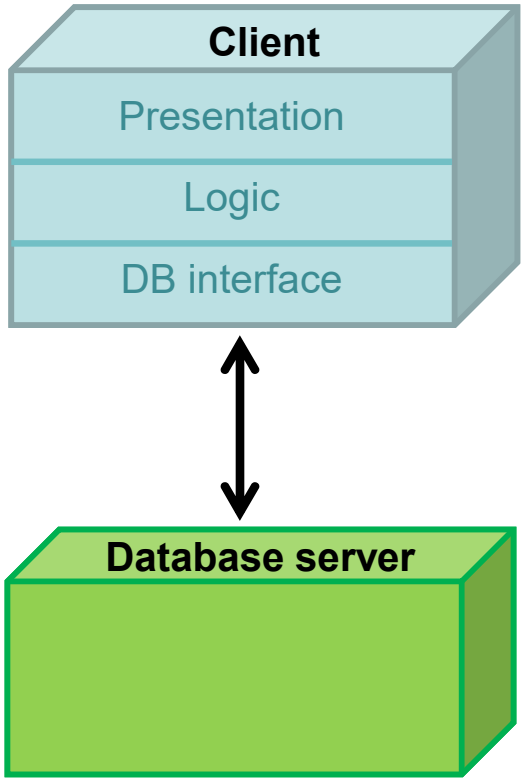
Application architectures (1)

- The architecture of database applications is typically based on the client-server model: server = database system

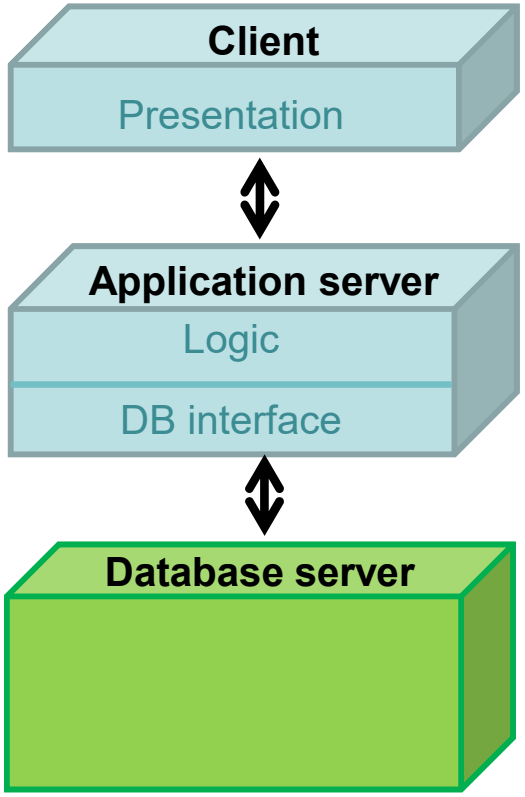


Application architectures (2)

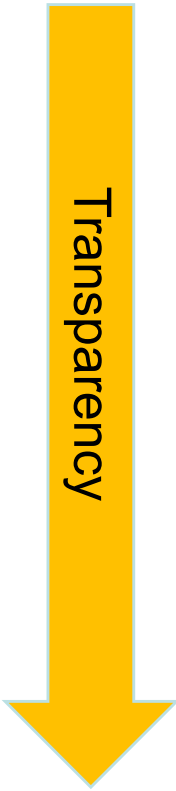
- Separation of the functionalities of an application
  - Presentation: presentation and user interaction
  - Logic: application logic (“business” logic)
  - Database interface: store, query, …



2-tier architecture



3-tier architecture

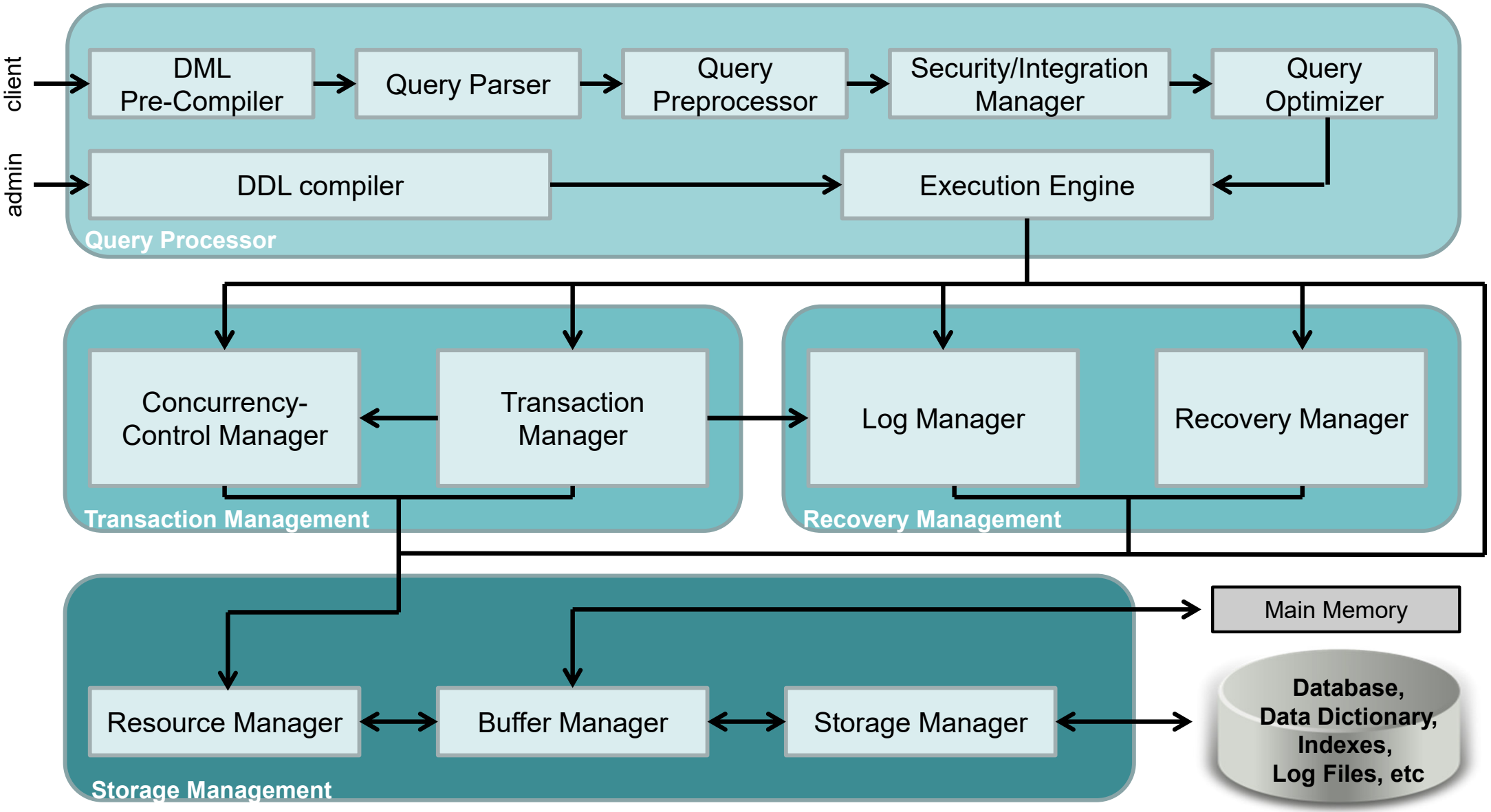


Why do we use a tier-architecture?

System architectures

- Description of the components of a database system
- Standardisation of interfaces between components
- Architecture proposed standards
  - ANSI-SPARC Architecture → three-level architecture (1978)
    - ANSI: American National Standards Institute
    - SPARC: Standards Planning and Requirement Committee
    - Essentially a refinement of the 3-level architecture presented
      - Internal level / refined operating system
      - More interactive and programming components
      - Interfaces are designated and standardised
  - Five-tier architecture (1987)
    - Describes transformation step in detail
    - Presented as part of IBM's “System R” DBMS prototype
- Each DBMS has its own architecture, often similar to the architecture proposed standards

Example: Architecture of MySQL (simplified)



Simplified image based on: R. Bannon et al.: MySQL Conceptual Architecture

Some specific systems

- (Object-)Relational DBMS
  - Oracle, IBM DB2, Microsoft SQL, MS Access
  - MySQL (www.mysql.org), PostgreSQL (www.postgresql.org),
  - Ingres (www.ingres.com), FireBird (www.firebirdsql.org)
  - CoreData (iOS), SQLite (Android)
- Object-oriented DBMS
  - Poet, Versant, ObjectStore
- XML-DBMS
  - Tamino (Software AG), eXcelon

The term data model

A data model (also called a database model) specifies...

1) Static properties: structure of the data
   a) Objects
   b) Relationships

   Examples: as graphs, in relations, as objects, as key-value pairs, …

2) Dynamic properties: operations
   a) Query operations (queries) and change operations
   b) Relationships between operations

   Example: “Give me all customers who ordered something in the last quarter”

3) Integrity conditions (constraints): conditions for
   a) Objects
   b) Operations

   Examples: values of the “age” attribute must be between 1 and 150

Examples of data models

- Design models: data models for designing DBs
  - ER model (entity-relationship model)
  - UML (Unified Modeling Language)
- Implementation models: data models for implementing DBs
  - Relational model
  - Hierarchical model → legacy databases
  - Network model → legacy databases
- Newer data models for specific applications
  - Object-oriented model
  - Key-value stores
  - Graph databases

Discussion

- What is a database and what is the motivation to use it?
- What are the advantages of data storage in relations?
- What is meant by the schema architecture?
- What are the architectural patterns for databases and what are they good for?
- What is meant by a data model?