Introduction To Database Systems

Alessandro Bozzon
TI1506: Web and Database Technology
ti1506-ewi@tudelft.nl

TU Delft

Announcements

- First Peer- Review Assignment starts today
  - Individual assignment, not team work
  - Deadline is December 5th, 11.59pm.

- Werkcollege on Javascript programming
  - DW-TZ 2.
  - Tuesday 22nd November. 13.45 - 15.45
  - Not mandatory, but recommended

- Lab assignment 2
  - Deadline is near!

Course overview [DB]

1. **Introduction to Database Systems**
2. The Relational Model
3. Introduction to SQL
4. SQL Schema and Data Manipulation
5. Conceptual Data Modelling with ER Diagrams
6. Database Conceptual Design
7. Database Logical Design

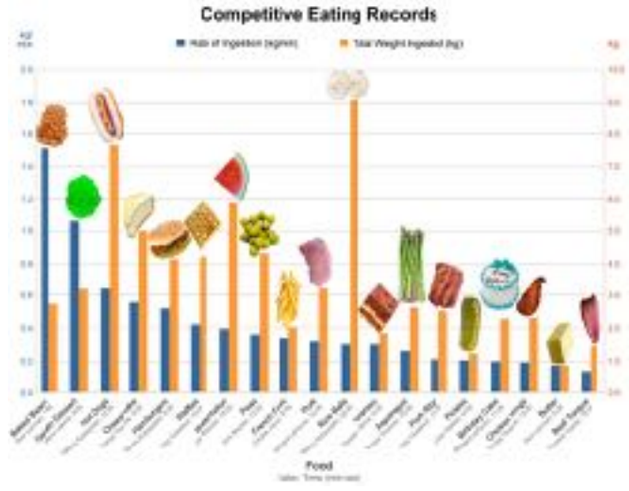
At the end of this lecture, you should be able to …

- **Understand** the reasons for using a Database
- **Understand** the importance of Database Management Systems
- **Describe** the main characteristics of a DBMS
- **Enumerate** and **define** the main elements of the Relational Model

Why Databases?

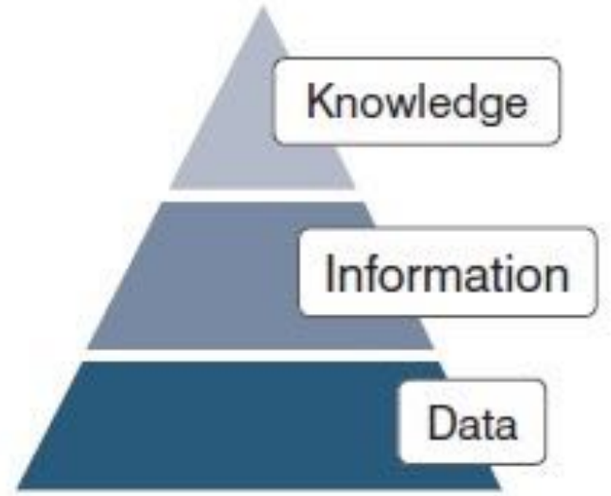
Management of Information

- Information is handled and recorded according to various techniques:
  - Informal ideas
  - Natural language (written or spoken)
  - Drawings, Diagrams
  - Numbers
  - Codes
- As activities become **systematised**, appropriate forms of organisation and **codification** for information have been devised
- E.g. Information about people



Management of Information

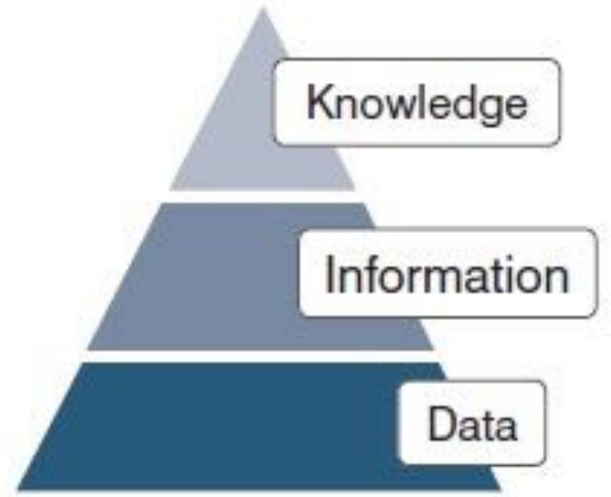
- In most computer-based systems (as well as in many other places) information is **represented** by means of data
  - **Data**: raw facts, to be interpreted and correlated in order to provide information
  - **Information**: Data processed in such a way that the information now has **relevance for a specific purpose or context**, and is therefore meaningful, valuable, useful and relevant
    - answering interrogative questions (e.g., ”who”, ”what”, ”where”, ”how many”, ”when”)
  - **Knowledge**: organisation and processing of information to convey understanding and experience



Management of Information

- In most computer-based systems (as well as in many other places) information is **represented** by means of data

"Alessandro" and 1 are a string and a number
  - two pieces of **data**



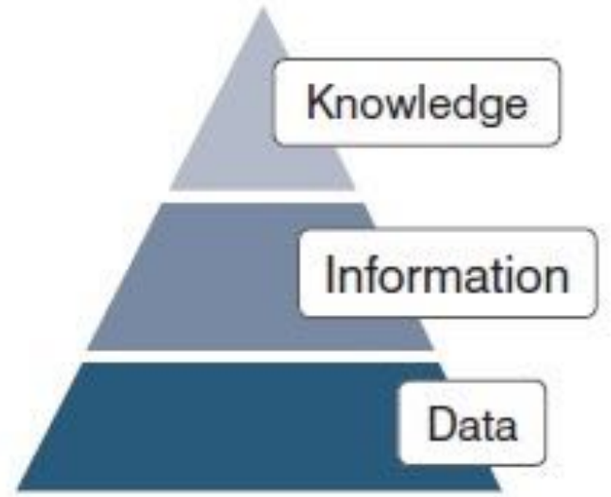
- **Information**: Data processed in such a way that the information now has **relevance for a specific purpose or context**, and is therefore meaningful, valuable, useful and relevant
  - answering interrogative questions (e.g., "who", "what", "where", "how many", "when")
- **Knowledge**: organisation and processing of information to convey understanding and experience

Management of Information

- In most computer-based systems (as well as in many other places) information is **represented** by means of data

"Alessandro" and 1 are a string and a number
- two pieces of **data**

If they are provided as a reply to a request
- "1st year course lectured by a teacher named Alessandro" then we get **information** out of them



- **Knowledge**: organisation and processing of information to convey understanding and experience

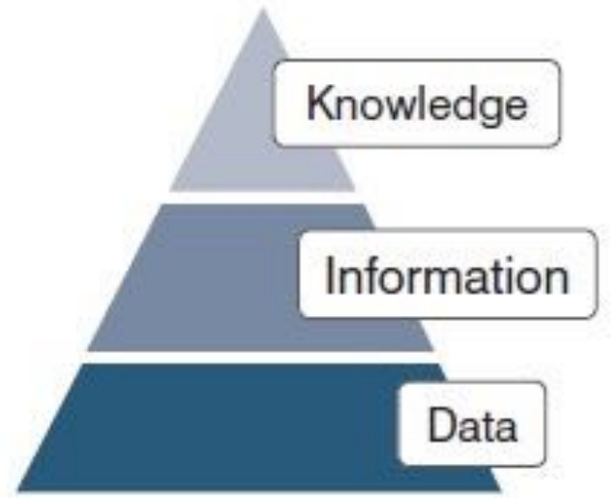
Management of Information

- In most computer-based systems (as well as in many other places) information is **represented** by means of data

"Alessandro" and 1 are a string and a number
- two pieces of **data**

If they are provided as a reply to a request
- "1st year course lectured by a teacher named Alessandro" then we get **information** out of them

When we combine this information with other details from the CourseBase, we gain "understanding" of the didactic offer of the EWI Faculty for bachelor students



Why Are Data so Important?

Why Are Data so Important?

Data are at the very core of information systems

- No organisation can operate without a good strategy for data management and access
- They are a core asset, and they must be managed and protected

Why Are Data so Important?

Data are at the very core of information systems

- No organisation can operate without a good strategy for data management and access
- They are a core asset, and they must be managed and protected

Data have a longer life-cycle than the processes and organisations that manage them

- e.g. bank data management systems did not change in decades (or centuries)
- But the processes that manage them change every year

Definitions of **Database**

Generic Definition

- A collection of interrelated data, used to represent information of interest to an information system
- Represents some aspects of the real world: the Universe of Discourse (UoD)
- Has an intended group of users and applications

More Technical Definitions

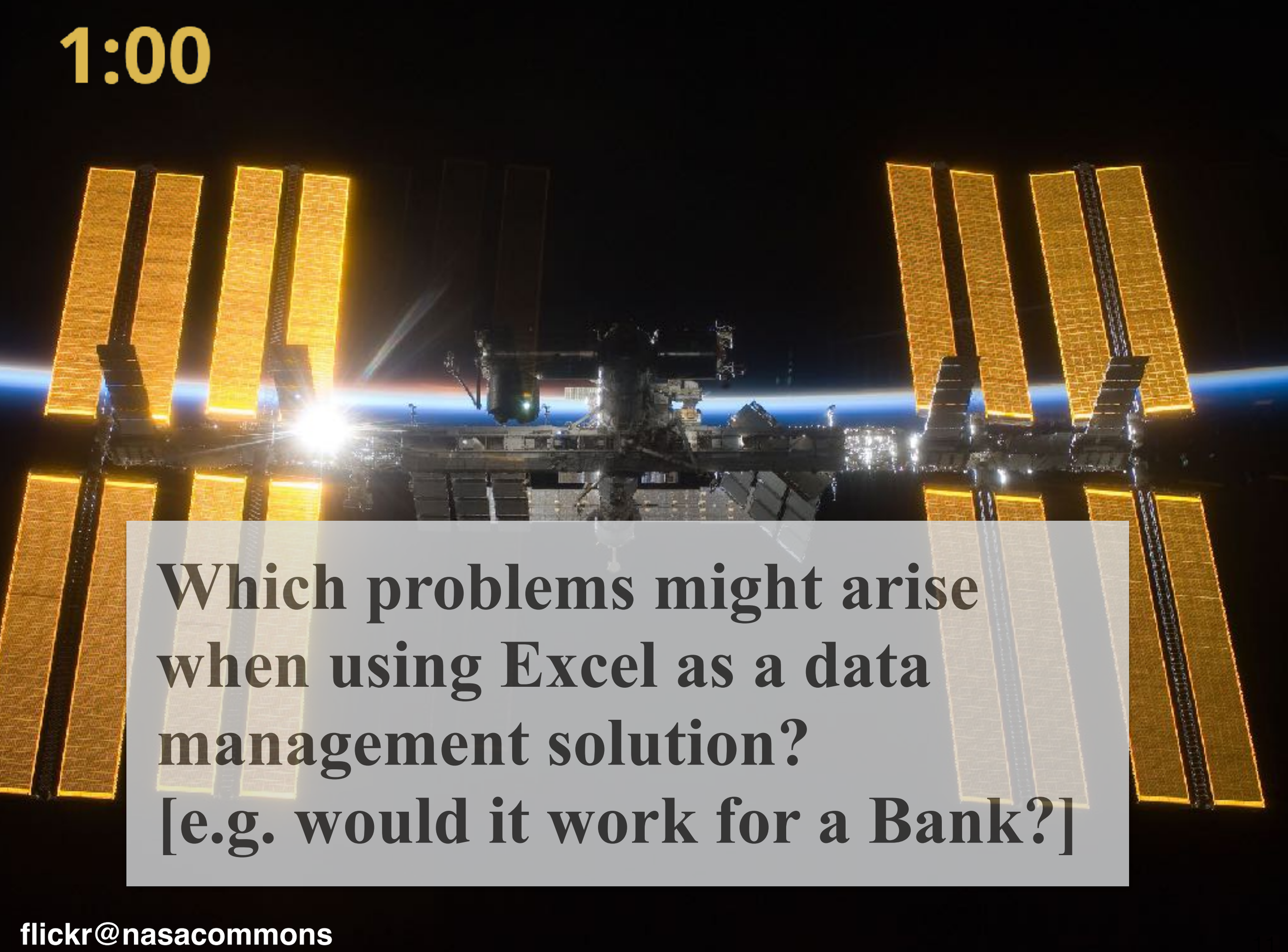
- A collection of files that store the **data**

Dealing with Data

- How to manage a large and persistent sets of data?
  - File systems
  - Store the data in files

- Developers define and implement the files needed for a specific software
  - A description of the organisation of the files (often just a stream of bytes)
  - Application logic to access / query / update files content

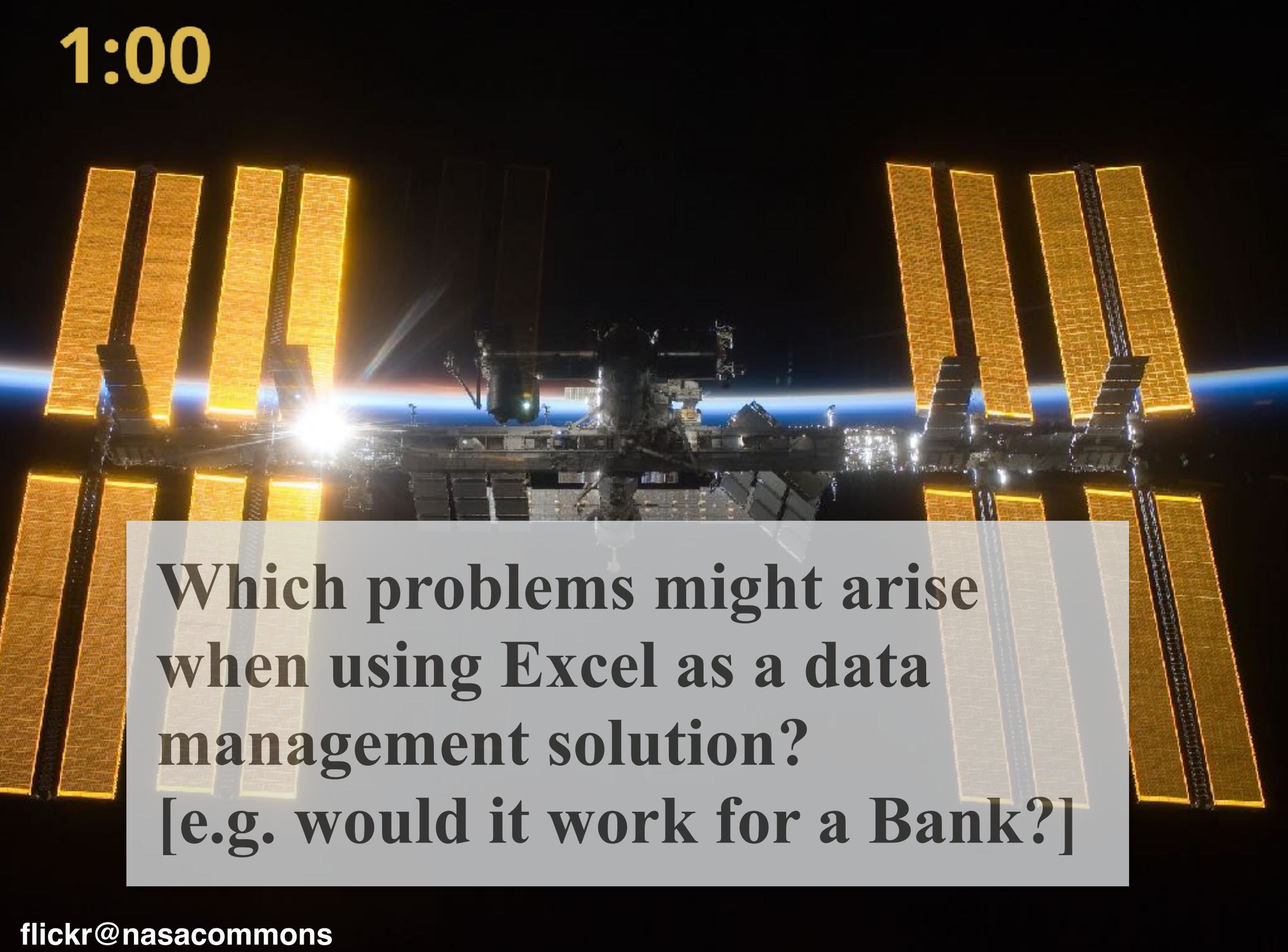
1:00



Which problems might arise when using Excel as a data management solution?
[e.g. would it work for a Bank?]

flickr@nasacommons

1:00



Which problems might arise when using Excel as a data management solution?
[e.g. would it work for a Bank?]

flickr@nasacommons

What if….

- Several applications make use of the same data
- The system crashes
- The dataset grows (let's say, up to 1 Tb)
- Many users want to access to the data, possibly concurrently
- Information needs are not pre-defined
- ….

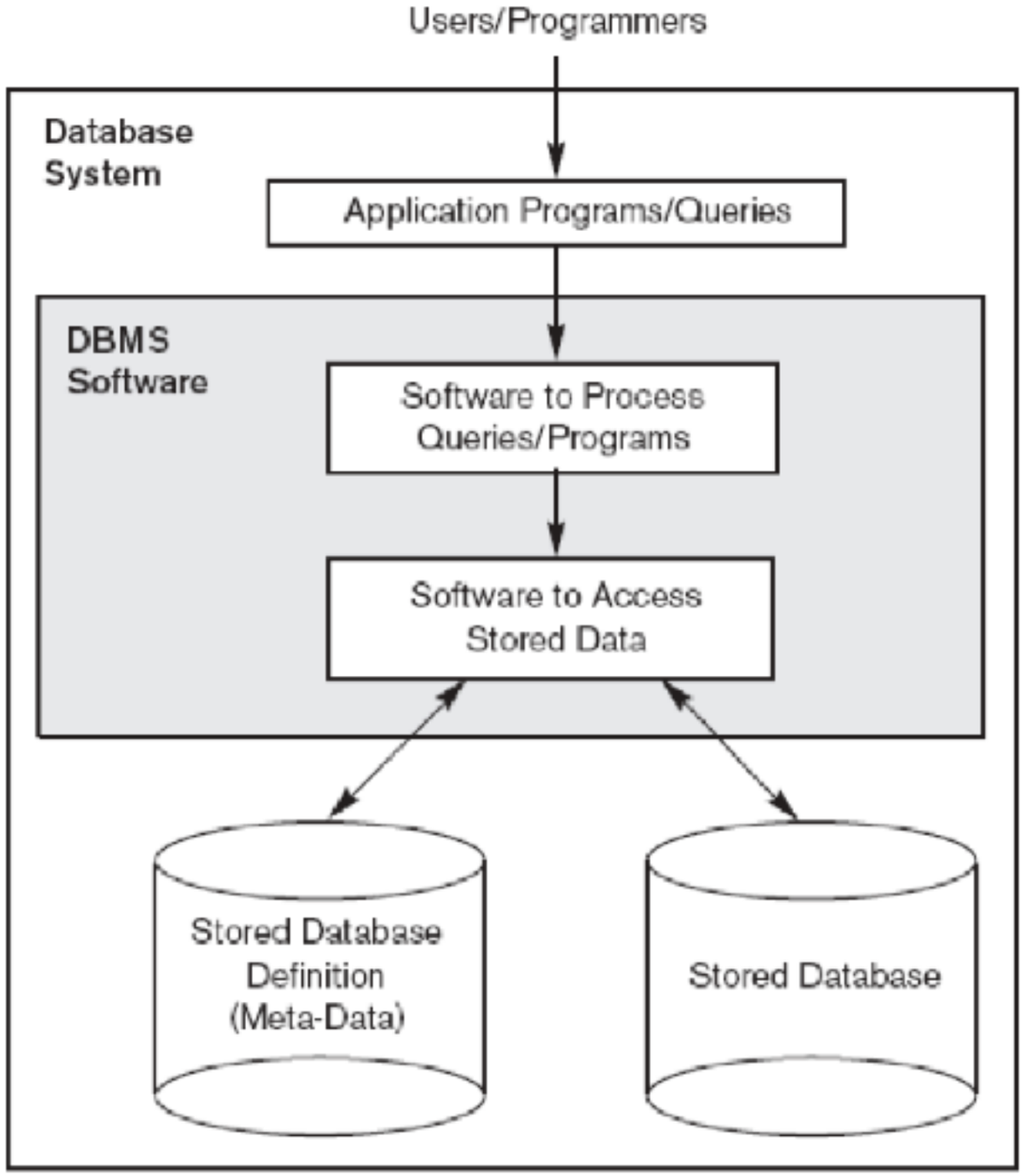
What if….

- Several applications make use of the same data
- The system crashes
- The dataset grows (let's say, up to 1 Tb)
- Many users want to access to the data, possibly concurrently
- Information needs are not pre-defined
- ….

We need a more **effective** solutions:
A **Database Management System**

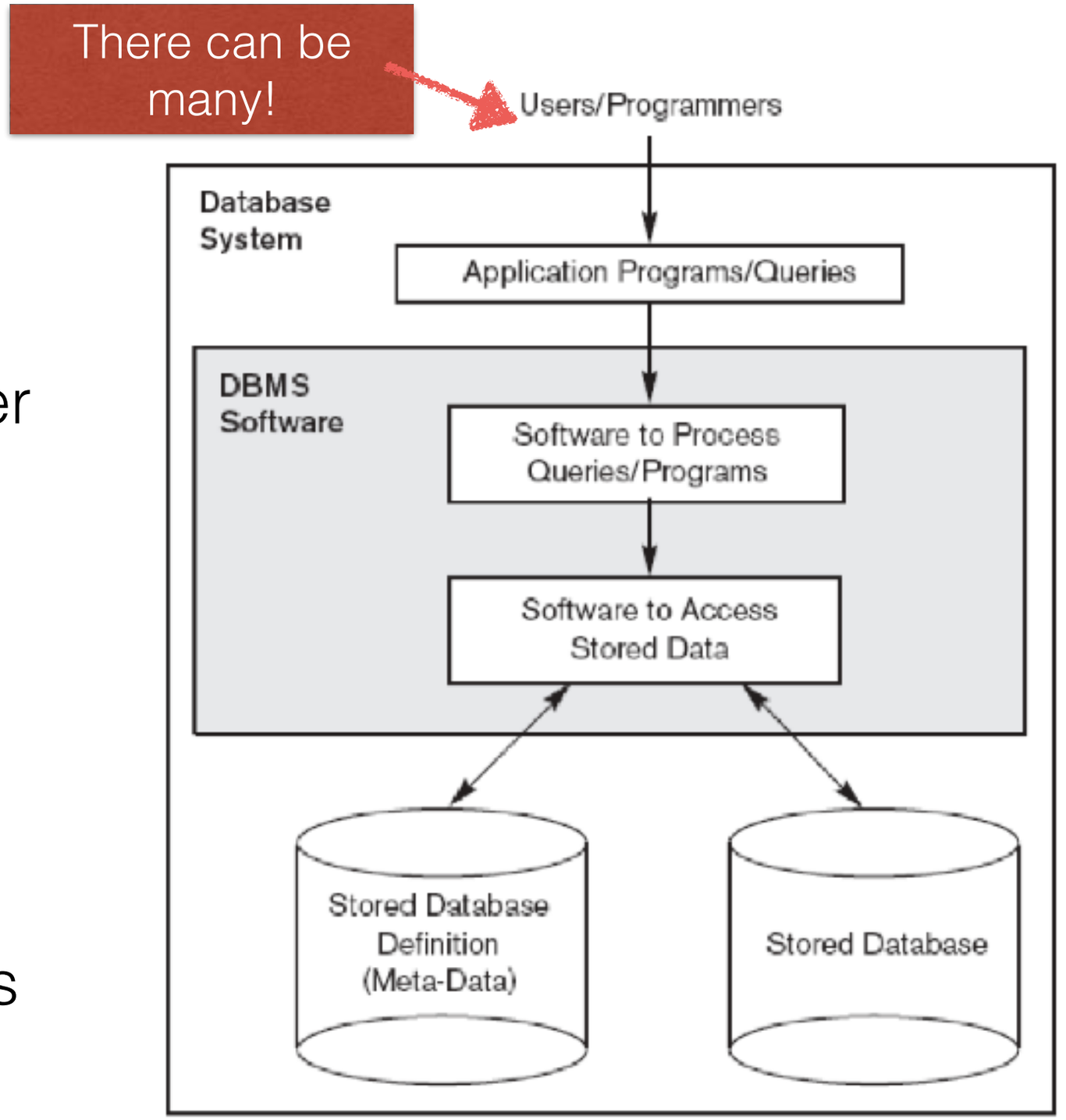
Database Management System

- A software system able to manage **collections of data** that are
  - **Large**: bigger, much bigger that the main memory available
  - **Shared**: used by various applications and users
  - **Persistent**: with a lifespan that is not limited to single executions of the programs that use them



Database Management System

- A software system able to manage **collections of data** that are
  - **Large**: bigger, much bigger that the main memory available
  - **Shared**: used by various applications and users
  - **Persistent**: with a lifespan that is not limited to single executions of the programs that use them



Typical Features of Database management systems

- **Reliability:** preservation of database content in case of hardware or software failure

- **Privacy**: controlling access and authorisations

- **Efficiency**: use the appropriate amount of resources (e.g. time and space)
- **Effectiveness**: supporting the productivity of its users
- **Self-describing nature** database system contains complete definition of structure (Meta-data) and rules of validity
  - Catalog describes the structure of the database

Types of DataBase Management Systems

Relational

- Based on the relational model (Codd, 1970)
- Data organised in homogeneous set of tuples (rows) forming relations (tables)
- SQL as generic data definition, manipulation and query language for relational data
- Ensure *atomicity*, *consistency, isolation*, and *durability* (ACID paradigm)
- Examples: **MySQL**, PostgresQL, SQL Server, SQLite, Oracle, MariaDB, etc.

Non-Relational

- **Key-Value**
  - Storage of records of tuples only containing a key and a value. The key uniquely identify the record, the value is an arbitrary chunk of data
  - Examples: Amazon Dynamo, Redis
- **Document-oriented**
  - Similar to key/value, but require structure data as values using formats like XML or JSON
  - Examples: MongoDB, CouchDB
- **Column-Oriented**
  - Store data by Columns rather than by row
  - Examples: Google BigTable, HBase, Apache Cassandra
- **Graph**
  - Data organised in graphs. Nodes describe main data entities, edges describe relationships.
  - Example: Neo4J

Additional Advantages of DBMSs

- Reduced application development time
- Flexibility
- Economies of scale
- Enforce integrity constraints on data
- Efficient query processing
- Several (built-in or external) user interfaces

When Not to Use a DBMS

- More desirable to use regular files for:
  - Simple, well-defined database applications not expected to change at all
  - Stringent, real-time requirements that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity
  - No multiple-user access to data

When Not to Use a DBMS

- More desirable to use regular files for:
  - Simple, well-defined database applications not expected to change at all
  - Stringent, real-time requirements that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity
  - No multiple-user access to data

It is just someone else's C program
In the beginning we may be impressed by its speed
But later we discover that it can be frustratingly slow
We can do any particular task faster outside the DBMS

When Not to Use a DBMS

- More desirable to use regular files for:
  - Simple, well-defined database applications not expected to change at all
  - Stringent, real-time requirements that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity
  - No multiple-user access to data

It is just someone else's C program
In the beginning we may be impressed by its speed
But later we discover that it can be frustratingly slow
We can do any particula

Question: would you use a DBMS on your Arduino ?

1:00

The DBMS acts as an interface between what two components of a database system ?

A. Database and SQL
B. Data and Database
C. Database Application and Database
D. Database and User

flickr@nasacommons

1:00

The DBMS acts as an interface between what two components of a database system ?

A. Database and SQL
B. Data and Database
C. Database Application and Database
D. Database and User

flickr@nasacommons

Data Abstraction and Modelling

Data Abstraction

- The major purpose of a database system is to provide users with an **abstract** view of the system
  - Complexity should be hidden from database users
  - Understanding improved by highlighting only the essential features of data

Data Abstraction

- The major purpose of a database system is to provide users with an **abstract** view of the system
  - Complexity should be hidden from database users
  - Understanding improved by highlighting only the essential features of data

Program-Data Independence

The system hides certain details of how data is created, stored, maintained, and accessed

Data Abstraction

- The major purpose of a database system is to provide users with an **abstract** view of the system
  - Complexity should be hidden from database users
  - Understanding improved by highlighting only the essential features of data

Program-Data Independence

The system hides certain details of how data is created, stored, maintained, and accessed

- Need a **Conceptual representation** of data
  - No details about **how** data is stored
  - With databases, we use **Data Models**

“Data Models” as programming languages

A notation used to ***describe*** data or information

- Provides means to achieve data abstraction

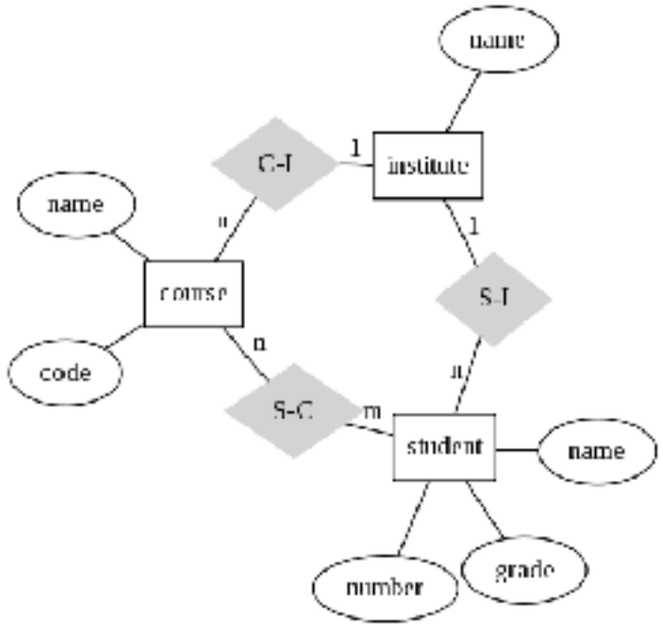
Typically describe

- Data
  - **Basic data types** (integer, char, etc.)
  - **structuring mechanism** (or **type constructor**), as in programming languages
  - **Relationships**
- Constraint
- Operations

“Data Models” as programming languages

A notation used to ***describe*** data or information

- Provides means to achieve data abstraction

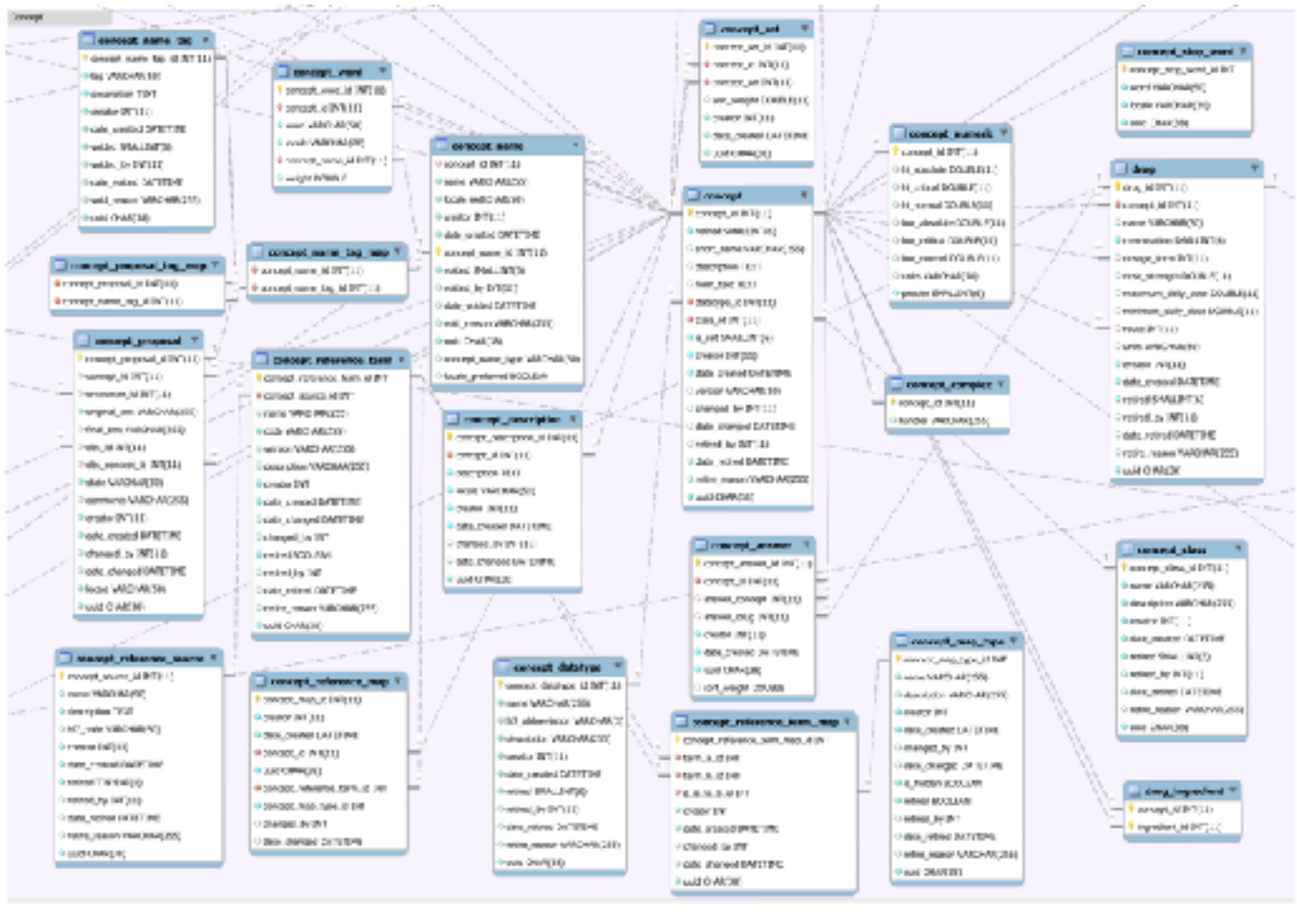


Typically describe

- Data
  - **Basic data types** (integer, char, etc.)
  - **structuring mechanism** (or **type constructor**), as in programming languages
  - **Relationships**
- Constraint
- Operations

“Data Models” as programs …or schema

- An abstraction (model) of the persistent data of some particular application or system.
  - In other words, the **database design**
  - It uses the facilities provided by the data model, in the previous sense of that term, to solve some specific problem



1:00

Which of the following data management details should be hidden from the users of a DBMS?

A. The logical organisation of data (e.g. tables)
B. The physical distribution of data (e.g. one or multiple servers)
C. The data model used to conceptually organise data
D. The data types handled by the database

flickr@nasacommons

1:00

Which of the following data management details should be hidden from the users of a DBMS?

A. The logical organisation of data (e.g. tables)
B. The physical distribution of data (e.g. one or multiple servers)
C. The data model used to conceptually organise data
D. The data types handled by the database

flickr@nasacommons

Three-Schema Architecture

GOAL

Separate the user application from the physical database

End Users

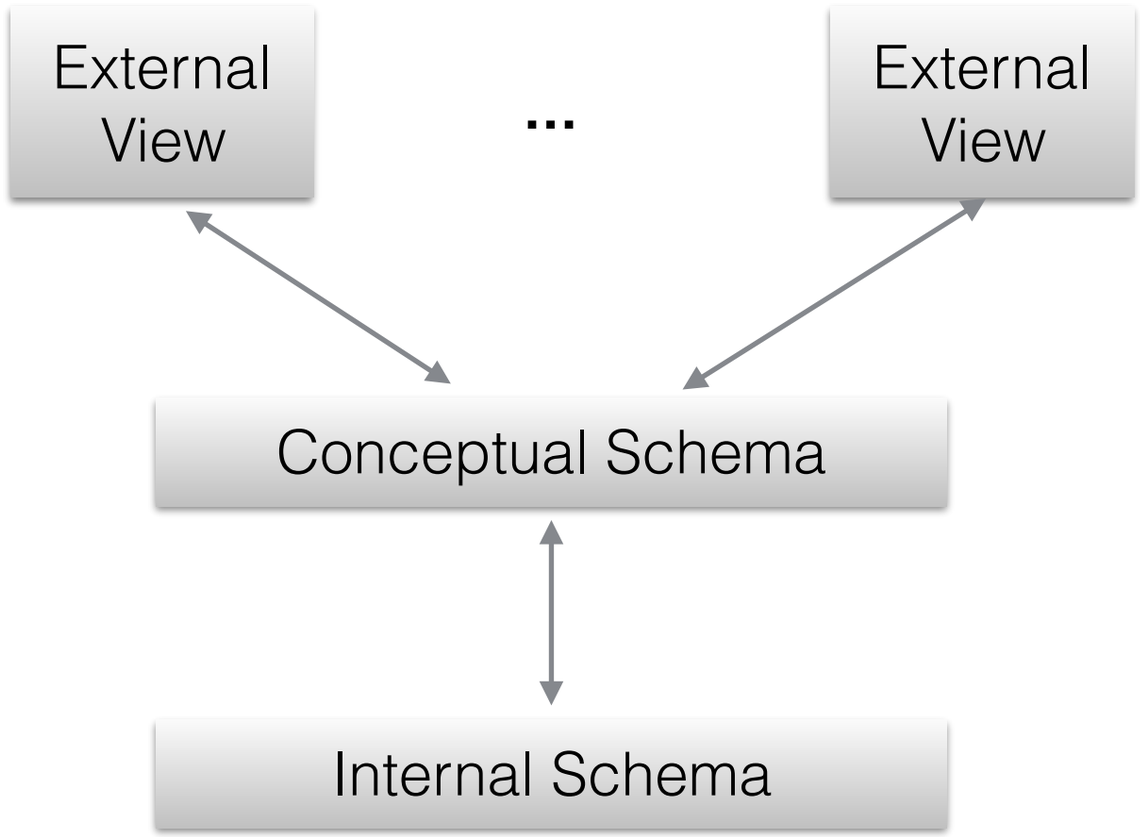
External Level

External/Conceptual Mapping

Conceptual Level

Conceptual/Internal Mapping

Internal Level



Three-Schema Architecture

GOAL

Separate the user application from the physical database

End Users

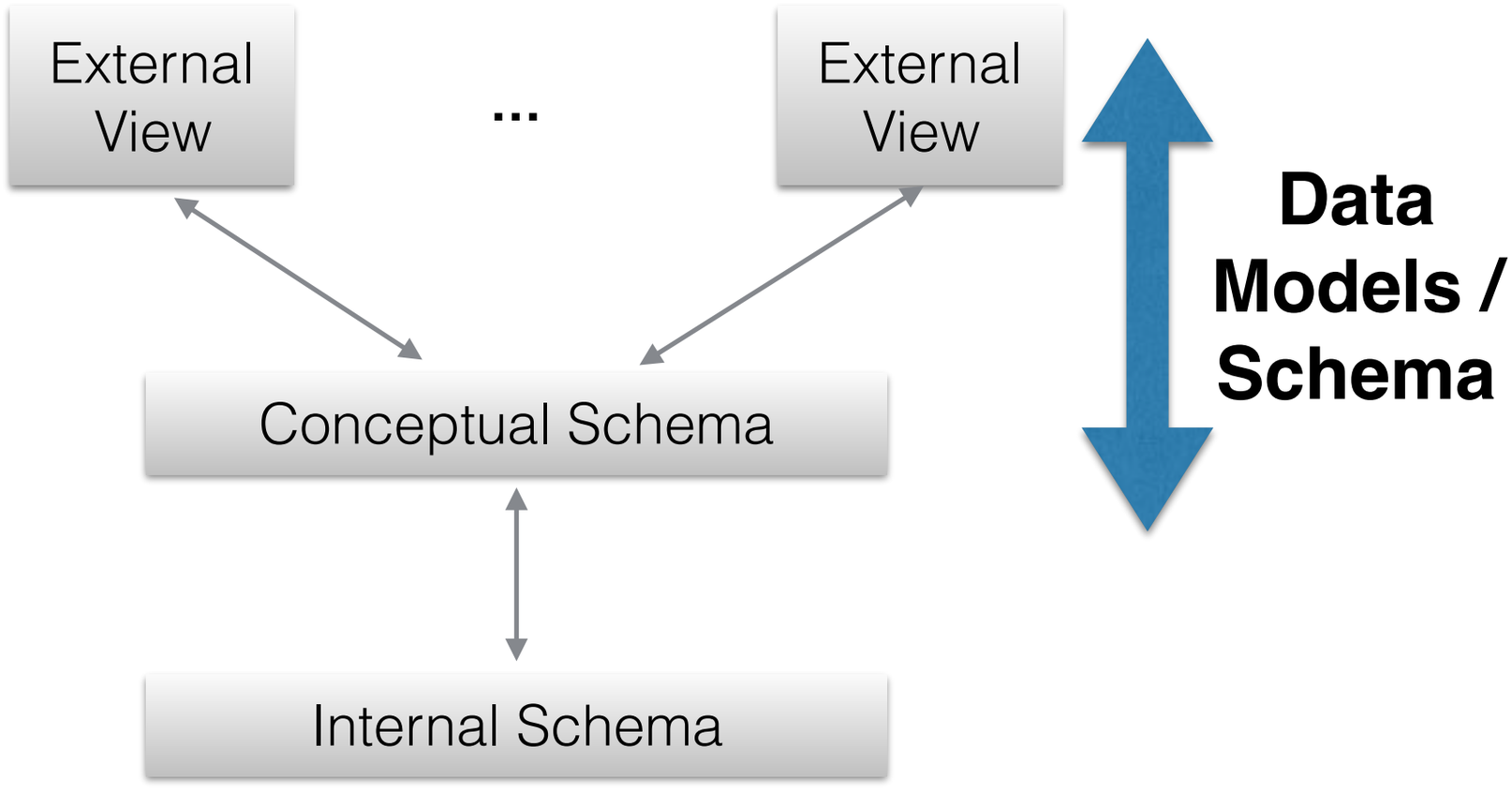
External Level

External/Conceptual Mapping

Conceptual Level

Conceptual/Internal Mapping

Internal Level



Three-Schema Architecture

GOAL

Separate the user application from the physical database

End Users

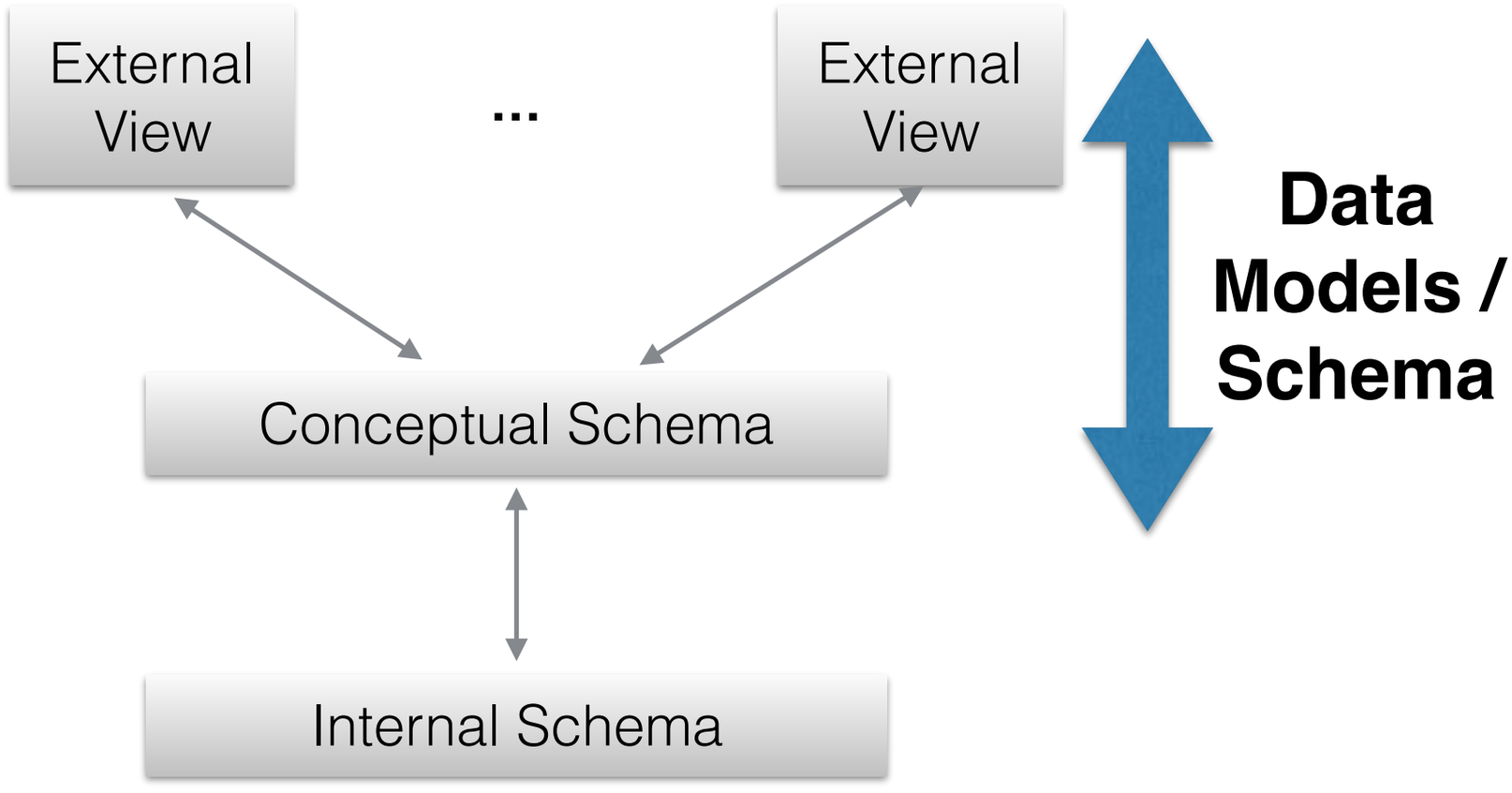
External Level

External/Conceptual Mapping

Conceptual Level

Conceptual/Internal Mapping

Internal Level



Not al DBMS separate the three levels completely and explicitly

Data Independence

Capacity to change the schema at one level of a database system without having to change the schema at the next higher level

Data Independence

Capacity to change the schema at one level of a database system without having to change the schema at the next higher level

- Guaranteed by the multi-level architecture

Data Independence

Capacity to change the schema at one level of a database system without having to change the schema at the next higher level

- Guaranteed by the multi-level architecture

- Two forms of data independence:
  - **Logical**: change the conceptual schema without changing external schema or application programs
    - Adding/Removing entities, data items, data types, constraints
  - **Physical**: change the internal schema without changes the conceptual schema
    - Physical file re-organization, indexes, etc.

Functional Independence

Capacity to change the implementation of a function (method) without impact on software applications

- This is often referred to as **information hiding**
  - E.g. sorting
- Function
  - Interface —> name of the function, and its arguments
  - Method —> how the functions is executed

- According to Codd, a DBMS should provide at at least 8 functions
  - Data storage, retrieval and update; user-accesible catalog; transaction support; concurrency control services; recovery services; authorisation services; support for data communication; integrity services

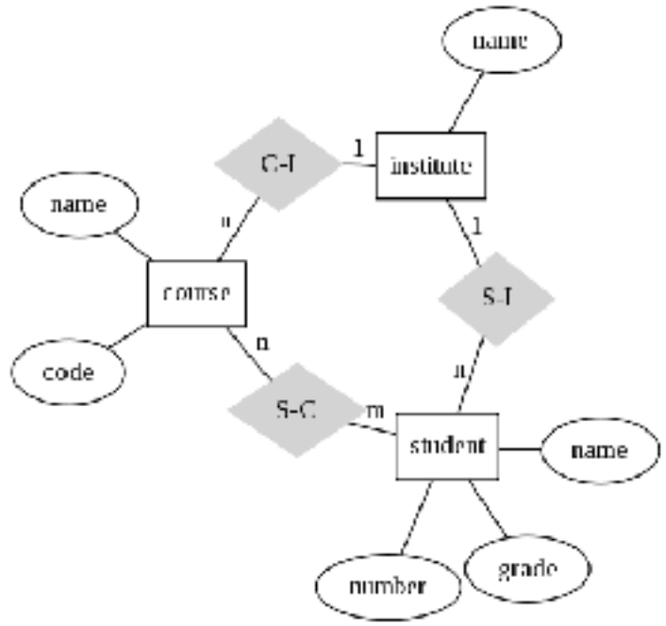
Again on Data Models

Again on Data Models

At the conceptual level of the three schema architecture, we distinguish two types of data models, according to their level of abstraction

Again on Data Models

At the conceptual level of the three schema architecture, we distinguish two types of data models, according to their level of abstraction

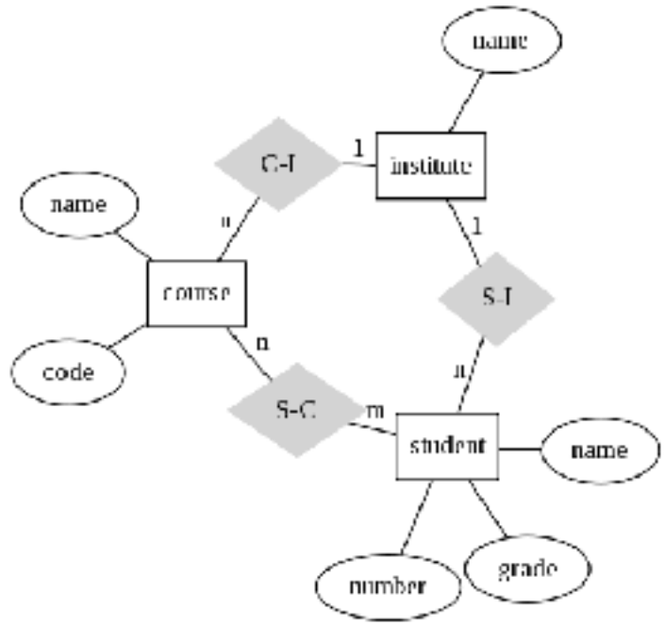


- **Conceptual** Data Models
  - Describe data in a way that is completely independent of any system
  - Representing the concepts of the real world
  - Used in the early stages of database design

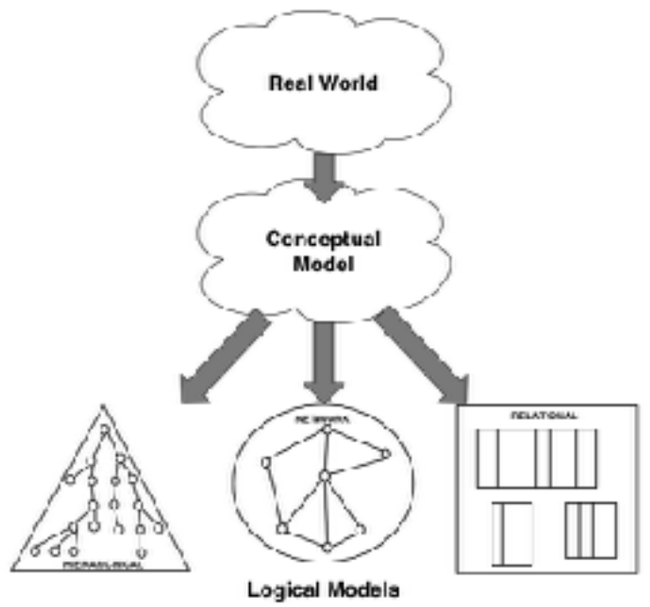
- The most popular is the **Entity-Relationship model**

Again on Data Models

At the conceptual level of the three schema architecture, we distinguish two types of data models, according to their level of abstraction



- **Conceptual** Data Models
  - Describe data in a way that is completely independent of any system
  - Representing the concepts of the real world
  - Used in the early stages of database design
  - The most popular is the **Entity-Relationship model**



- **Logical** Data Models
  - Used in DBMSs for the organisation of data at a level that abstracts from physical structures
  - Examples: **Relational**, network, hierarchical

Tables: representation of relations

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

1:00

Which data model requires the most detail in its internal data representation?

A. Network
B. Relational
C. Semantic
D. Entity Relationship

flickr@nasacommons

1:00

Which data model requires the most detail in its internal data representation?

A. Network
B. Relational
C. Semantic
D. Entity Relationship

flickr@nasacommons

Database Languages

DBMS Language Classes

- **Data Definition Language** (**DDL**)
  - Defines the **logical** and **physical** schema
  - Often used also for access authorisation specification

- **Data Manipulation Language** (**DML**)
  - Allows **retrieval**, insertion, deletion, modification of database instances

- **Storage Definition Language** (**SDL**)
  - Specifies the internal schema

- **View Definition Language** (**VDL**)
  - Specifies user views/mapping to conceptual schema
  - Typically the same as DDL

DBMS Language Classes

- **Data Definition Language** (**DDL**)
  - Defines the **logical** and **physical** schema
  - Often used also for access authorisation specification

SQL

- **Data Manipulation Language** (**DML**)
  - Allows **retrieval**, insertion, deletion, modification of database instances

SQL

- **Storage Definition Language** (**SDL**)
  - Specifies the internal schema

- **View Definition Language** (**VDL**)
  - Specifies user views/mapping to conceptual schema
  - Typically the same as DDL

SQL

Database Languages

Various forms (a contribution to effectiveness)

- Interactive textual and declarative language
  - **SQL** (Structured Query Language)
- Interactive commands embedded in a host language (Java, C++, Python, Javascript)

- By means of non-textual **user-friendly** interfaces
  - Graphical user interfaces, Natural language interfaces, Speech Input and Output, Interfaces for the DBA

1:00

Which model “adapts” a conceptual model to a specific DBMS?

A. Entity
B. Internal
C. External
D. Database

flickr@nasacommons

1:00

Which model “adapts” a conceptual model to a specific DBMS?

A. Entity
B. Internal
C. External
D. Database

flickr@nasacommons

SQL - Structured Query Language

- A text-based declarative language for **relational** databases used as:
  - DDL and VDL by DBMS designer and DBA
  - DML by users

An Example of SQL Query

```
SELECT Name FROM Student S, Test_Tesult R, Test T
WHERE S.studId=R.studId
  AND T.testId=R.testId AND T.City= "Delft"
  AND R.AnswerCorrect="Yes" AND T.topic="Database"
GROUP BY S.studId,Name
HAVING SUM(R.Score) >= 6
```

DBMS Modules

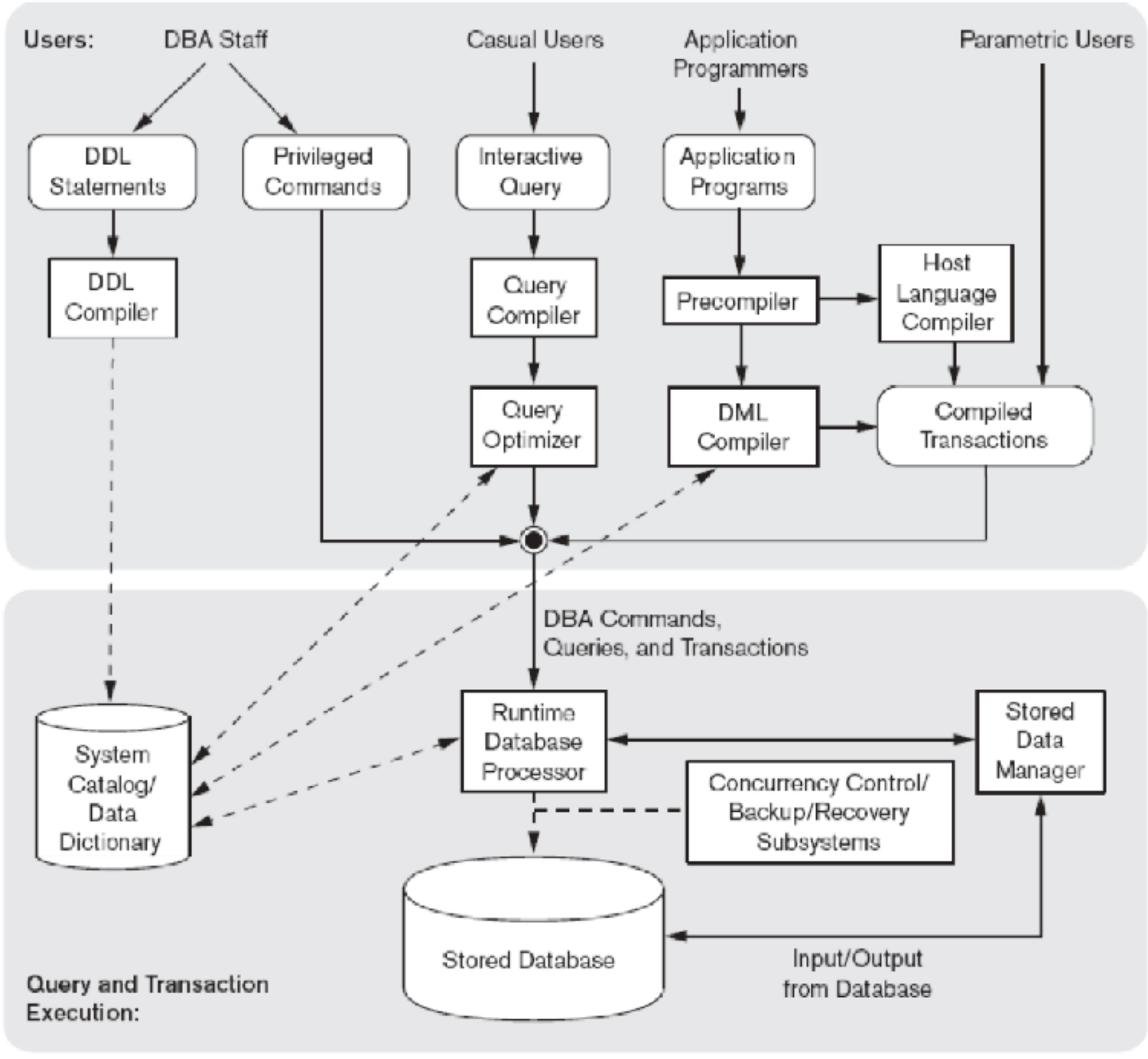
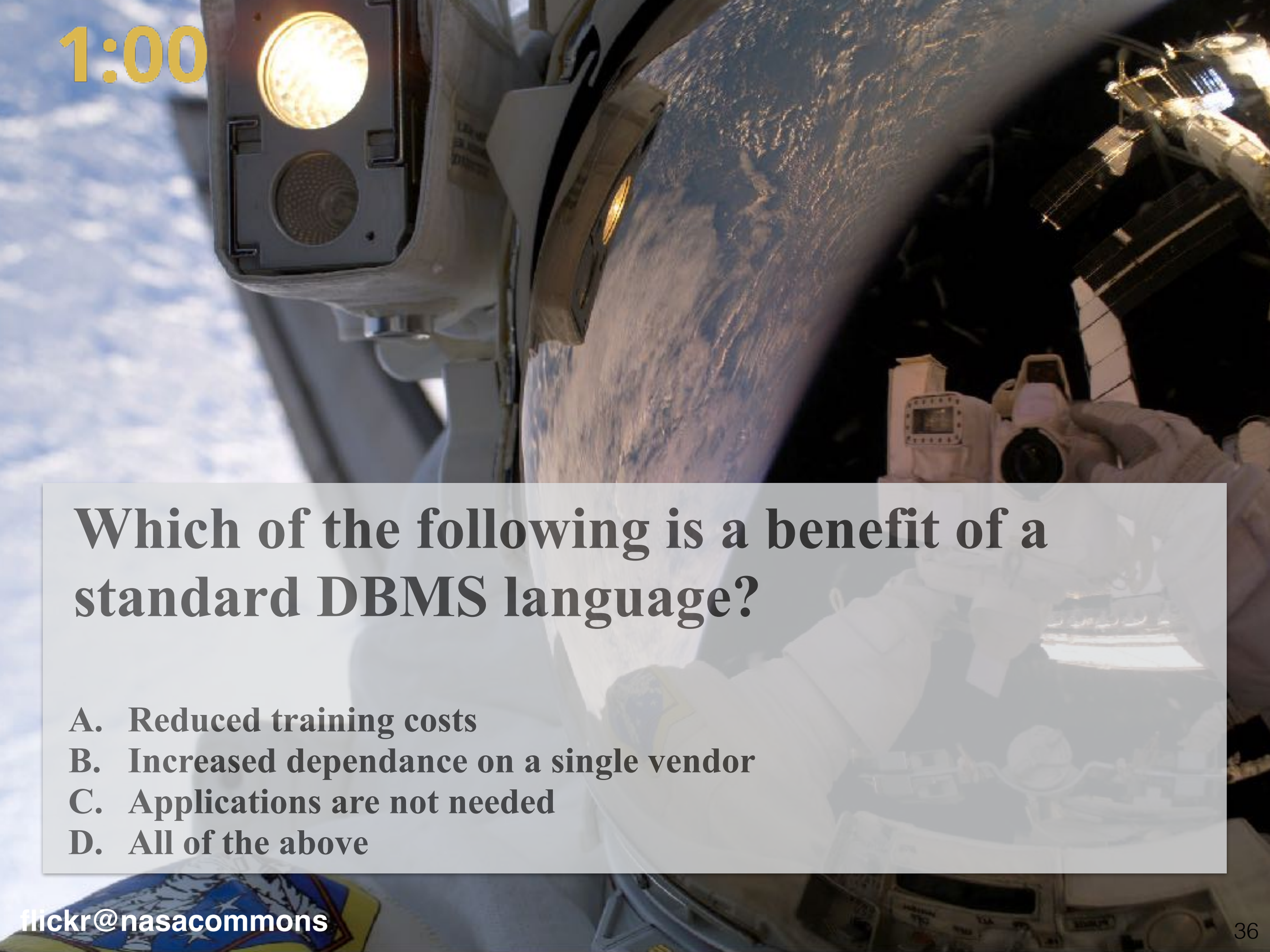


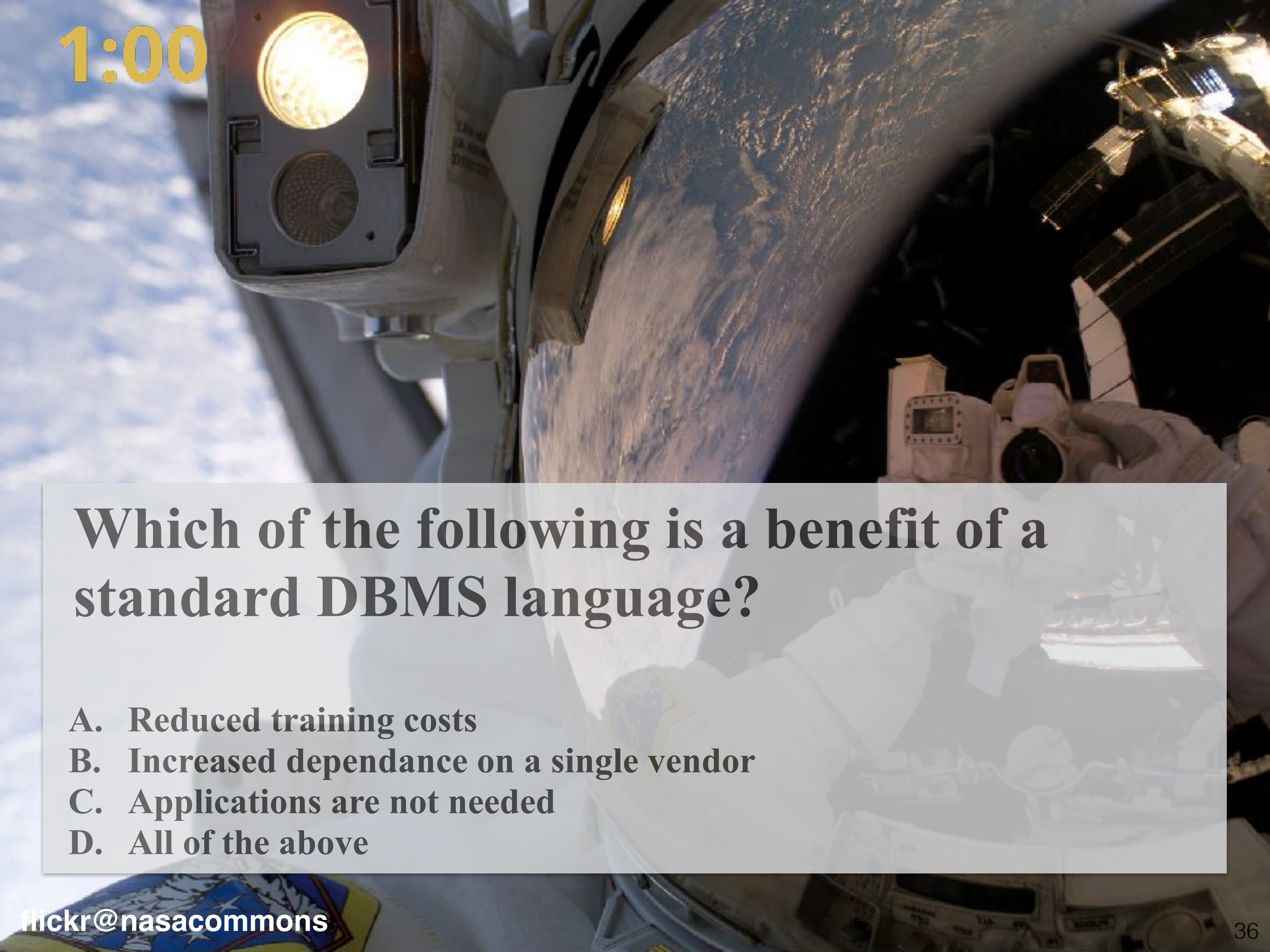
Figure 2.3
Component modules of a DBMS and their interactions.



Which of the following is a benefit of a standard DBMS language?

A. Reduced training costs
B. Increased dependance on a single vendor
C. Applications are not needed
D. All of the above

flickr@nasacommons



Which of the following is a benefit of a standard DBMS language?

A. Reduced training costs
B. Increased dependance on a single vendor
C. Applications are not needed
D. All of the above

flickr@nasacommons

The Relational Model

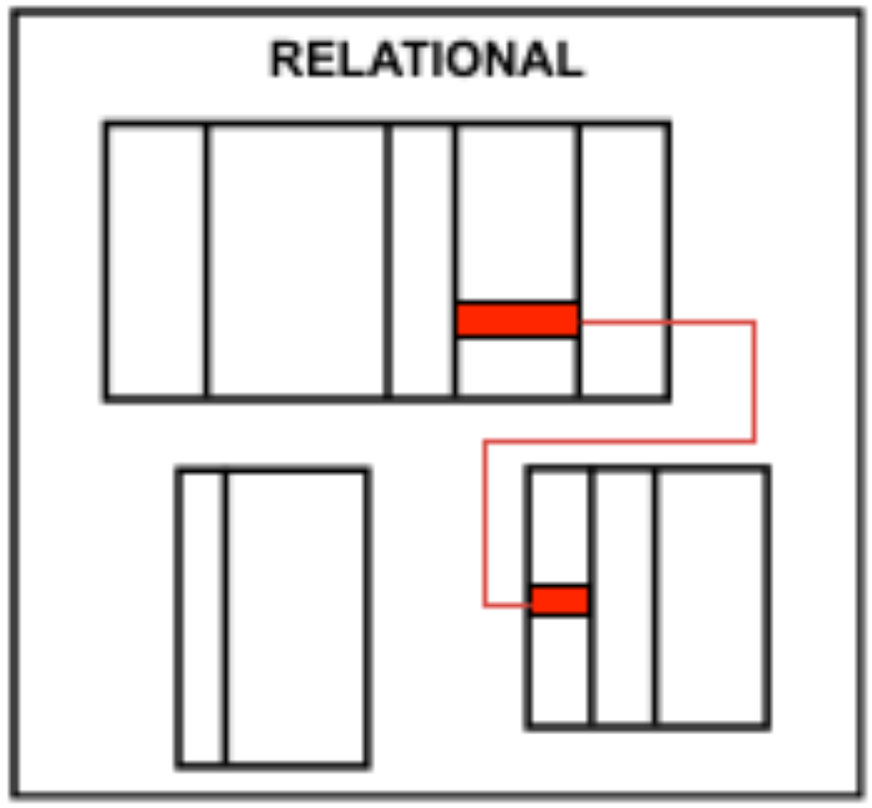
The Relational Model

- Proposed by E. F. Codd in 1970 in order to support data independence

- It is based on (a variant of) the mathematical notion of **relation**
  - Relations are naturally represented by means of **tables**

- The relational model satisfies data independence requirements

- Made available in commercial DBMSs in 1981
  - It is not easy to implement data independence efficiently and reliably!



Domain

Domain *D*: **set** of ***atomic*** values having coherent *data types*, a *logical definition*, and a *name*

- **Atomic**: indivisible, as far as the data model is concerned (more later)
- **Data Type (and format)**: e.g. String, Integer, Date, Timestamp, etc.
- **Logical Definition**: meaning of the domain in the context of the data model

- **Examples**
  - *NetID*: a set of alphanumeric characters without punctuation
  - *USA Phone Number*: the set of ten-digits numbers valid in the United States
  - *Name*: the set of character strings that represent names of persons
  - *Grade Point Average*: possible values of computed grade points averages: each must be a real (floating point) number between 0 and 10

Relation Schema (or Relation Scheme)

Relation Schema *R* is denoted by
$R(A_1, A_2, \ldots, A_n)$

- A Relation Schema *describes* a relation
- R —> Relation name
- $(A_1, A_2, \ldots, A_n)$ —> List of attributes
  - Each A_i is the **role** played by some domain D in the relation schema R
    - Several attributes can *have the same domain*
  - $dom(A_i)$ —> the domain of A_i

- **Degree** (or **-arity**) of R —> is the number of attributes in R

Relation Schema (or Relation Scheme)

Relation Schema *R* is denoted by

$R(A_1, A_2, \ldots, A_n)$

Notation used from now on

- A Relation Schema *describes* a relation
- R —> Relation name
- $(A_1, A_2, \ldots, A_n)$ —> List of attributes
  - Each A_i is the **role** played by some domain D in the relation schema R
    - Several attributes can *have the same domain*
  - $dom(A_i)$ —> the domain of A_i

- **Degree** (or **-arity**) of R —> is the number of attributes in R

Relation Schema Example

Relation Schema

STUDENT(Name, NetID, Home_Phone, Mobile_Phone, Age)

Relation Name

Attribute
Name: Home_Phone
Domain: 10 Digits Number

Attribute
Name: Mobile_Phone
Domain: 10 Digits Number

Relation Schema Example

Relation Schema

STUDENT(Name, NetID, Home_Phone, Mobile_Phone, Age)

Relation Name

Attribute
Name: Home_Phone
Domain: 10 Digits Number

Attribute
Name: Mobile_Phone
Domain: 10 Digits Number

Using this notation, order of attributes matters!

Relation (or Relation Instance)

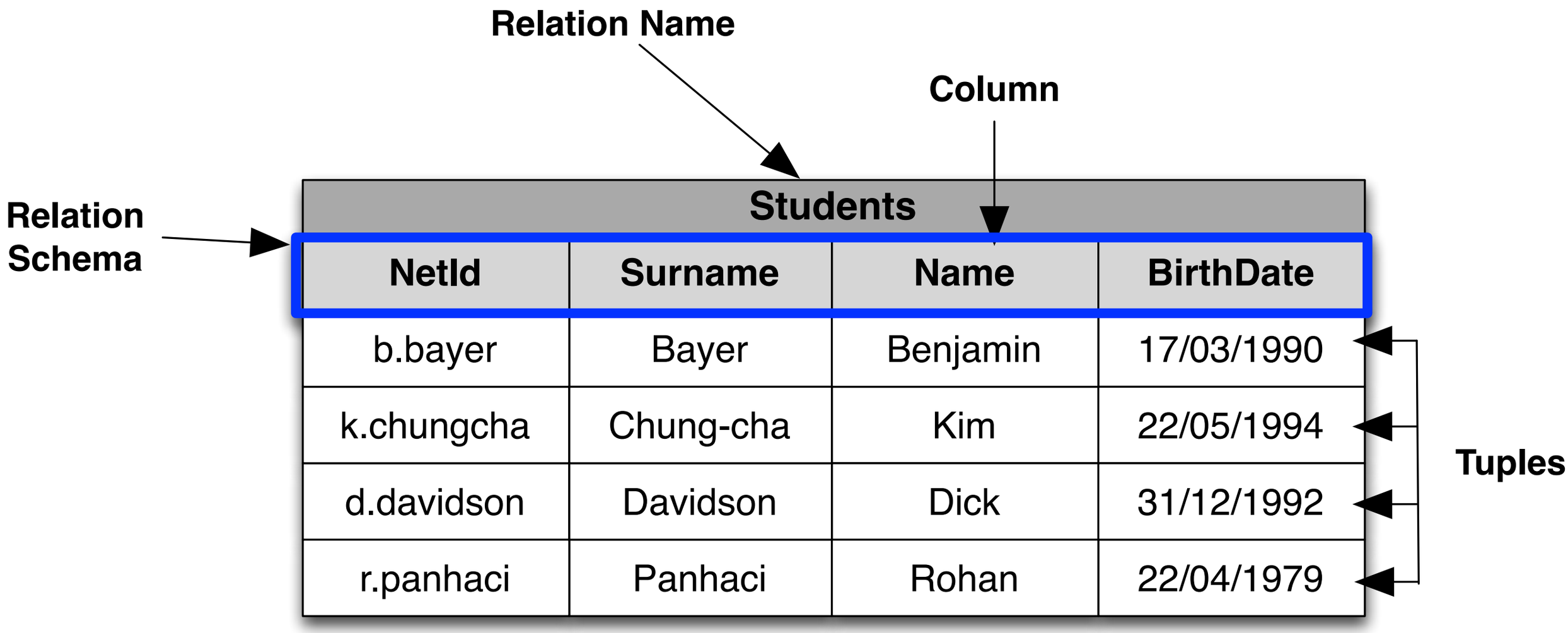
Relation *r* of the **Relation Schema** $R(A_1, A_2, \ldots, A_n)$ is denoted by $r(R)$

- A **set** of **n-tuple** $r = \{t_1, t_2, \ldots, t_m\}$
- Each **n-tuple** t_i is an ordered list of *n* values

$$t = <v_1, v_2, \ldots, V_n>$$

- Each value $v_i,\ 1 <= i <= n$ is
  - an element of $dom(A_i)$
  - a special **NULL** value (we will cover later)

Example of Relation



Characteristics of Relations

Relation *!=* Table

- A relation is defined as a **set** of tuples
  - Tuples have no specified order
  - Many orders could be specified for the same relation
  - But files on disk are (typically by insertion)

Why? A relation represents facts at a logical level

- A relation **has no duplicate tuples**
  - **No two tuples can gave the same combination of values for *all* the attributes**

Why? Sets have no duplicates

Characteristics of Relations

- Each value in a tuple is **atomic**
  - Not divisible into components

- Multi-valued attributes are not allowed
  - They **must become relations**

- Composite attributes are not allowed
  - Split into simple attributes

- E.g. *Name*:
  - “Alessandro”, “Jan”, “Rob” (1 value for each tuple) —> **YES!**
  - “Alessandro;Jan;Rob” (many values in single tuple) —> **NO!**

Why?
First Normal Form
assumption

Basis of flat relational model

1:00

Which of the following relations are identical?

A

Receipts		
Number	Date	Total
10	01/01/2014	100
15	03/01/2014	300
20	05/01/2014	50
25	05/01/2014	10
30	10/01/2014	400

B

Receipts		
Number	Total	Date
10	100	01/01/2014
15	300	03/01/2014
20	50	05/01/2014
25	10	05/01/2014
30	400	10/01/2014

C

Receipts		
Number	Date	Total
30	10/01/2014	400
15	03/01/2014	300
10	01/01/2014	100
20	05/01/2014	50
25	05/01/2014	10

A. *None*
B. *A and B*
C. *A and C*
D. *B and C*

flickr@nasacommons

1:00

Which of the following relations are identical?

A

Receipts		
Number	Date	Total
10	01/01/2014	100
15	03/01/2014	300
20	05/01/2014	50
25	05/01/2014	10
30	10/01/2014	400

B

Receipts		
Number	Total	Date
10	100	01/01/2014
15	300	03/01/2014
20	50	05/01/2014
25	10	05/01/2014
30	400	10/01/2014

C

Receipts		
Number	Date	Total
30	10/01/2014	400
15	03/01/2014	300
10	01/01/2014	100
20	05/01/2014	50
25	05/01/2014	10

A. *None*
B. *A and B*
C. *A and C*
D. *B and C*

flickr@nasacommons

NULL Values

- The relational model impose a rigid structure to data:
  - Information is represented by means of tuples
  - Tuples have to conform to relation schemas
- In practice, the available data might not conform to the required formats
- Sometimes information is missing or unknown

NULL values are used to represent values of attributes that 1) may be unknown, or 2) may not apply to a tuple

Example Complete Information

- (A *Geemente* has municipality office, a *Dorp* doesn't)
  - *Den Haag* has a municipality office, but we do not know its address
  - *Delfgauw* has no municipality office
  - *Rijswijk* may have a municipality office, but we don't know

City	
Name	**OfficeAddress**
Amsterdam	Amstel 1, 1011 PN
Den Haag	
Delfgauw	
Rijswijk	

Why having a special value?

- We should not (despite what often happens) use domain values (zero, 99, empty string, etc.) to represent lack of information:
  - there need not be “unused”values
  - “unused”values could become meaningful
  - in programs, we should be able to distinguish between actual values and placeholders
    - e.g. calculate the average age of a set of people, where 0 is used for unknown ages!

Types of NULL Value

- At least two:
  - a value is **unknown** (exists but it is not known)
    - E.g. a person's birth date is not known
  - a value is **not available** (exists but it is purposely withheld)
    - e.g a person has a home phone but does not want it to be listed
  - a value is **not applicable** (undefined for this tuple)
    - e.g. an attribute *LastCollegeDegree* would be `NULL` for a person with no college degree

- DBMS do not distinguish between the types: they implicitly adopt the non-available value

- We could (and often should) put restrictions on the presence of NULL values in tuples (we will see later with SQL)

Introducing Multiple Relations

- A database has several relations
- Tuples in such relations are often related in various ways

The state of the whole database corresponds to the states of all its relations at a particular point in time

Examples

Students			
NetId	**Surname**	**Name**	**BirthDate**
b.bayer	Bayer	Benjamin	17/03/1990
k.chungcha	Chung-cha	Kim	22/05/1994
d.davidson	Davidson	Dick	31/12/1992
r.panhaci	Panhaci	Rohan	22/04/1979

Exams		
Student	**Grade**	**Course**
b.bayer	10	TI2735-A
k.chungcha	8	TI1205
d.davidson	9	TI1505
r.panhaci	7.5	TI1505

Courses		
Code	**Title**	**Lecturer**
TI1205	OOP	Zaidman
TI1505	Web and DB	Bozzon
TI2735-A	Computational Intelligence	Redi

1:00

What makes the following database instance meaningless (in the context of a Dutch university)? [Free text answer]

Exams			
Student	**Grade**	**Course**	**Honours**
b.bayer	11	TI2735-A	
k.chungcha	7	TI1205	
d.davidson	9	TI1505	honours
d.davidson	10	TI1500	honours

Courses	
Code	**Title**
TI1205	OOP
TI1505	Web and DB
TI2735-A	Computational Intelligence

flickr@nasacommons

1:00

What makes the following database instance meaningless (in the context of a Dutch university)? [Free text answer]

Exams			
Student	**Grade**	**Course**	**Honours**
b.bayer	11	TI2735-A	
k.chungcha	7	TI1205	
d.davidson	9	TI1505	honours
d.davidson	10	TI1500	honours

Courses	
Code	**Title**
TI1205	OOP
TI1505	Web and DB
TI2735-A	Computational Intelligence

flickr@nasacommons

A Meaningless Database Instance

Exams			
Student	**Grade**	**Course**	**Honours**
b.bayer	11	TI2735-A	
k.chungcha	7	TI1205	
d.davidson	9	TI1505	honours
d.davidson	10	TI1500	honours

Courses	
Code	**Title**
TI1205	OOP
TI1505	Web and DB
TI2735-A	Computational Intelligence

- Grades are between 0 and 10
- **Honours** can be awarded only if grade is **A**
- Different students must have different NetID
- Exams must refer to **existing** courses

Constraint

Properties in the real world to be modelled by our database

A property that must be satisfied by all meaningful database instances

- Motivations:
  - Useful to describe the application in greater detail
  - A contribution to “data quality”
  - An element in the design process (more on “normal forms”)
  - Used by the system in choosing the strategy for query processing

Constraints

- There are **constraints** that restrict the values in the database
  - Model-based (**implicit**) constraints (e.g. no duplicates)
  - Schema-based (**explicit**) constraints —> SEE NEXT

Constraints

- There are **constraints** that restrict the values in the database
  - Model-based (**implicit**) constraints (e.g. no duplicates)
  - Schema-based (**explicit**) constraints —> SEE NEXT

A RDBMS system makes sure that all the constraints are satisfied

Constraints

- There are **constraints** that restrict the values in the database
  - Model-based (**implicit**) constraints (e.g. no duplicates)
  - Schema-based (**explicit**) constraints —> SEE NEXT

A RDBMS system makes sure that all the constraints are satisfied

- But also
  - Application-based (**semantic, business**) constraints
    - Difficult to express or enforce in the data model
  - Also **data dependencies** (functional and multivalued)
    - covered in another course

Relational Database Schemas

A **Relational Database Schema** $S = \{R_1, R_2, \ldots, R_m\}$ is a set of relation schemas and a set of **Integrity Constraints** *IC*

A **Relational Database State** *DB* of *S* is a set of relation states $DB = \{r_1, r_2, \ldots, r_m\}$ such that

- r_i is **a** state of R_i
- r_i states **satisfy** the integrity constraints *IC*

A Database State that does not obey all the integrity constraints is called an **invalid state**

Type of Explicit Constraints

- **Intra-Relational Constraints**
  - tuple constraints
  - domain constraints
  - uniqueness constraints

- **Inter-Relational Constraints**
  - integrity constraints
  - referential constraints

Tuple Constraints

Define conditions on the values of each tuple, independently from other tuples

- A possible syntax: boolean expressions with atoms that compare attributes, constants or expressions over them
  - Example: *NOT(Honours = ”honours”) OR (Grade = ”10”)*
  - Example (on another schema): *Net = Amount – Deductions*

Domain Constraints

A domain constraint that involve a **single attribute**

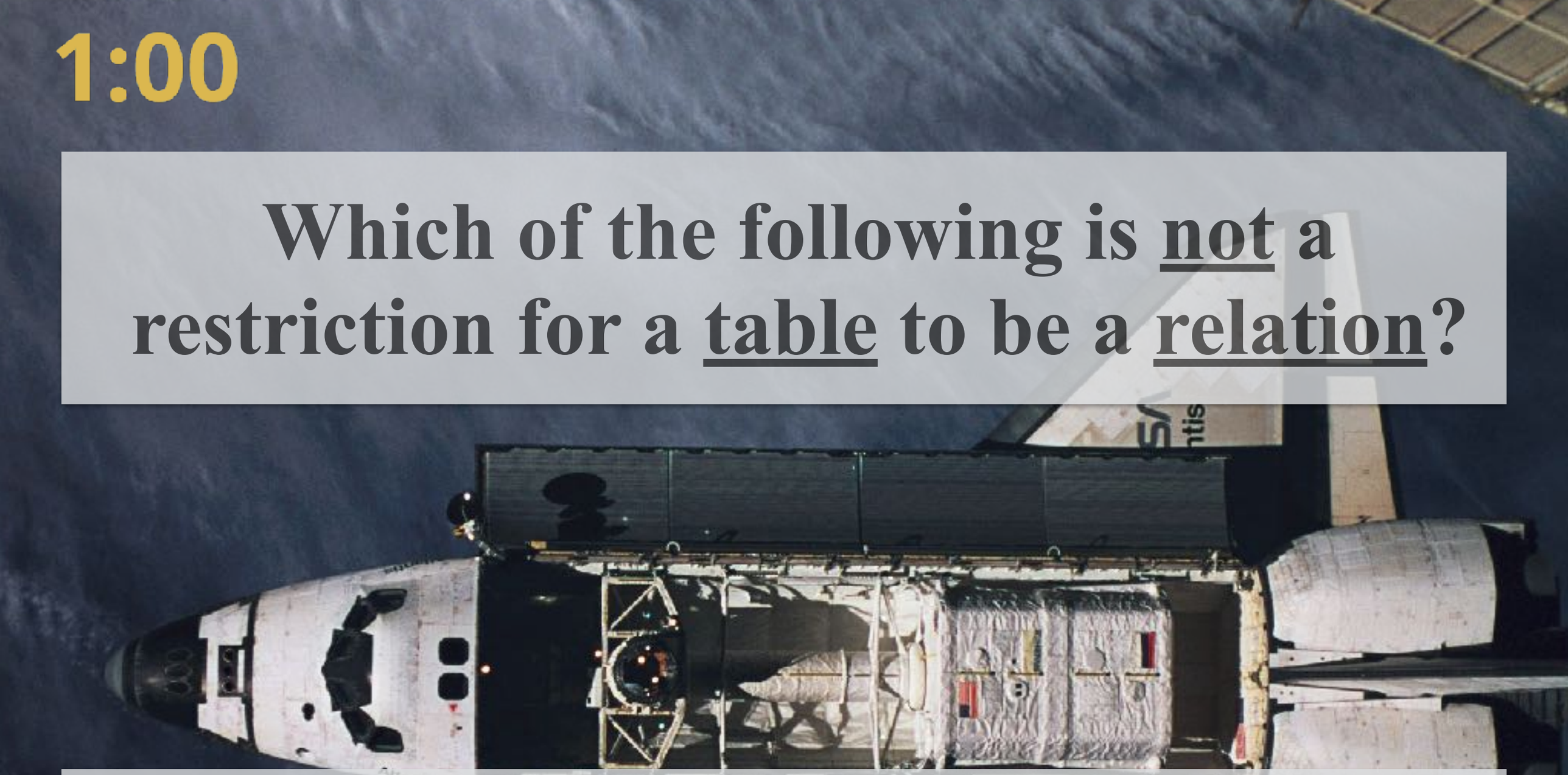
- On data type
  - Numeric for integers and real numbers, Characters, Booleans, Fixed-length strings, Variable-length strings, Date, Time, Timestamp, Money, etc.

- On value ranges
  - Example: *(Grade ≥ 0) AND (Grade ≤ 10)*

- Value within an enumeration

1:00

Which of the following is <u>not</u> a restriction for a <u>table</u> to be a <u>relation</u>?

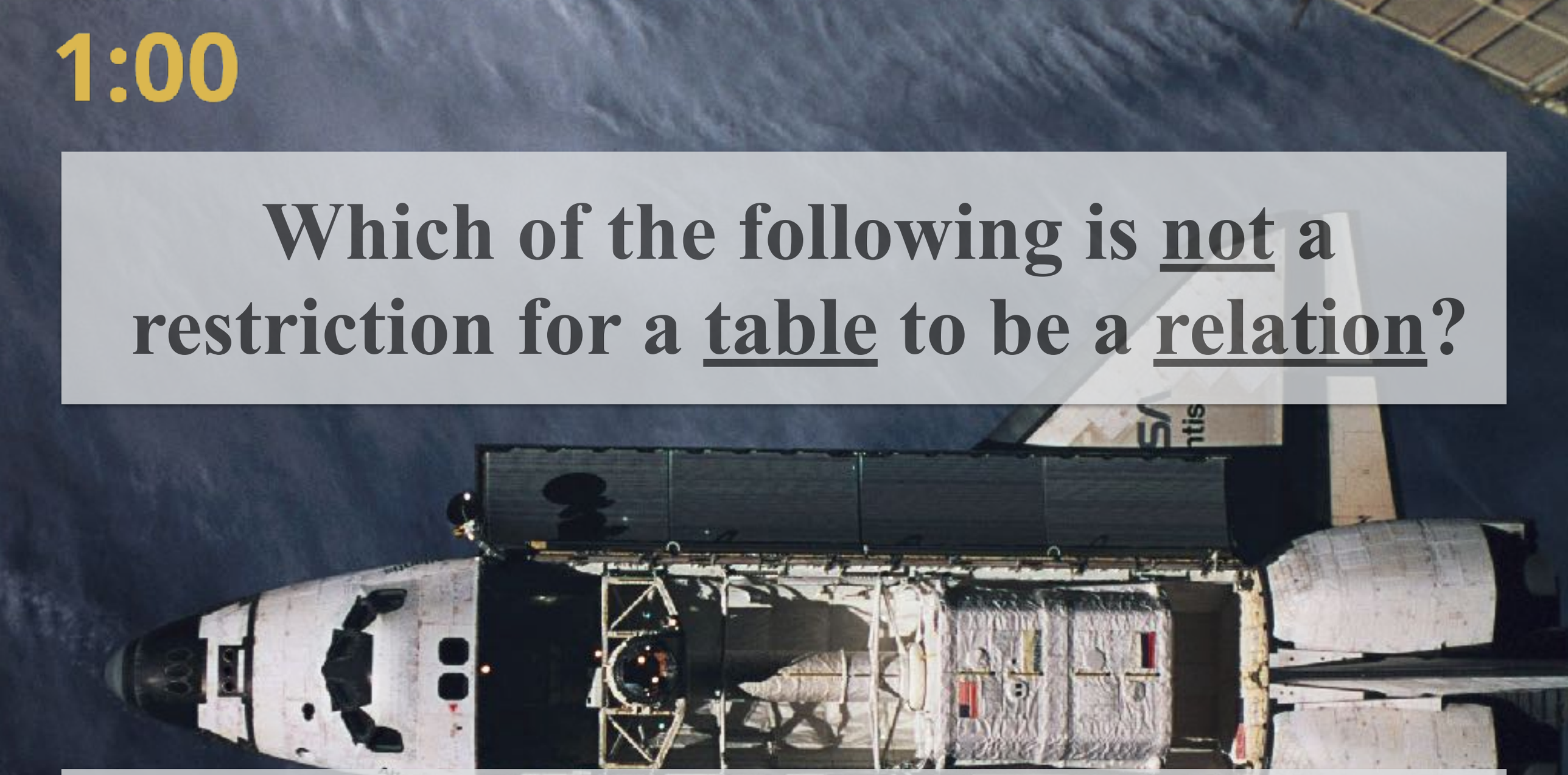


A. *The cells of the table must contain a single value.*
B. *All of the entries in any column must be of the same kind.*
C. *The columns must be ordered.*
D. *No two rows in a table may be identical.*

flickr@nasacommons

1:00

Which of the following is <u>not</u> a restriction for a <u>table</u> to be a <u>relation</u>?



A. *The cells of the table must contain a single value.*
B. *All of the entries in any column must be of the same kind.*
C. *The columns must be ordered.*
D. *No two rows in a table may be identical.*

flickr@nasacommons

Unique Identification of Tuples

- Tuples in a relation instance are **unique**
- But there are other **subsets of attributes** of a relation schema *R* with the same uniqueness property

Students				
NetId	**Surname**	**Name**	**BirthDate**	**DegreeProg**
b.bayer	Bayer	Benjamin	17/03/1990	Electrical
k.bayer	Bayer	Kim	17/03/1990	Electrical
d.bayer	Bayer	Dick	31/12/1992	Informatica
d.panhaci	Panhaci	Dick	31/12/1992	Wiskunde
d.panhaci2	Panhaci	Dick	13/08/1981	Wiskunde

- The registration number identifies students
  - There is no pair of tuples with the same value for NetId
- Personal data identifies students
  - There is no pair of tuples with the same values on each of Surname, FirstName, BirthDate

Key Constraints

A set of attributes K is a **superkey** for a relation *R* if in any state *r* of *R* no two distinct tuples t_1 and t_2 have $t_1[K] = t_2[K]$ (the same values for the attributes in K)

- Every relation has at least one default **superkey**

WHICH ONE?

Key Constraints

A set of attributes K is a **superkey** for a relation *R* if in any state *r* of *R* no two distinct tuples t_1 and t_2 have $t_1[K] = t_2[K]$ (the same values for the attributes in K)

- Every relation has at least one default **superkey**
  - The set of all its attributes

WHICH ONE?

Key Constraints

A set of attributes K is a **superkey** for a relation *R* if in any state *r* of *R* no two distinct tuples t_1 and t_2 have $t_1[K] = t_2[K]$ (the same values for the attributes in K)

- Every relation has at least one default **superkey**
  - The set of all its attributes

WHICH ONE?

- A **superkey** can have redundant attributes
  - i.e. attributes in *K* that, even when removed, does not influence the property of *K* being a superkey

Key

A **key** *K* of a relation schema *R* is a **minimal superkey** of *R*

- There exists no other superkey *K'* of *R* that is contained in *K* as proper subset
- Removing any attribute *A* from *K* leaves a set of attributes *K'* that is not a **superkey** of *R* anymore

Properties of Keys

- A key is a **superkey**, but not vice-versa

- A **superkey** formed by a single attribute is also a **Key**

- The value of a **key** attribute can be used to **uniquely identify** (**and access**) each tuple in the relation
  - i.e. key values are unique

- A set of attributes constituting a **key** is a **time-invariant** property of the relation schema valid for all its states

1:00

Considering the relation in the picture, which of the following sets of attributes are valid superkeys?

1. NetId
2. {NetId, BirthDate}
3. {Name, Surname}
4. {Name, Surname, Birthdate}

Students				
NetId	Surname	Name	BirthDate	DegreeProg
b.bayer	Bayer	Benjamin	17/03/1990	Electrical
k.bayer	Bayer	Kim	17/03/1990	Electrical
d.bayer	Bayer	Dick	31/12/1992	Informatica
d.panhaci	Panhaci	Dick	31/12/1992	Wiskunde
d.panhaci2	Panhaci	Dick	13/08/1981	Wiskunde

A. *1*
B. *1 and 2*
C. *1, 2, and 4*
D. *All*

1:00

Considering the relation in the picture, which of the following sets of attributes are valid superkeys?

1. NetId
2. {NetId, BirthDate}
3. {Name, Surname}
4. {Name, Surname, Birthdate}

Students				
NetId	Surname	Name	BirthDate	DegreeProg
b.bayer	Bayer	Benjamin	17/03/1990	Electrical
k.bayer	Bayer	Kim	17/03/1990	Electrical
d.bayer	Bayer	Dick	31/12/1992	Informatica
d.panhaci	Panhaci	Dick	31/12/1992	Wiskunde
d.panhaci2	Panhaci	Dick	13/08/1981	Wiskunde

A. *1*
B. *1 and 2*
C. *1, 2, and 4*
D. *All*

Primary Key

- A relation schema may have more than one **key**
  - Each key is called **candidate key**
- One **candidate key** is designated as **primary key**
  - Notation: the attributes in the primary key are <u>underlined</u>
- Other candidate keys might be designated as **unique key**

Students				
<u>NetId</u>	**Surname**	**Name**	**BirthDate**	**DegreeProg**
b.bayer	Bayer	Benjamin	NULL	Informatica
k.chungcha	Bayer	Kim	17/03/1990	Electrical
k.panhaci	Panhaci	Kim	NULL	NULL
k.panhaci2	Panhaci	Kim	31/12/1992	Electrical

Entity Integrity Constraint

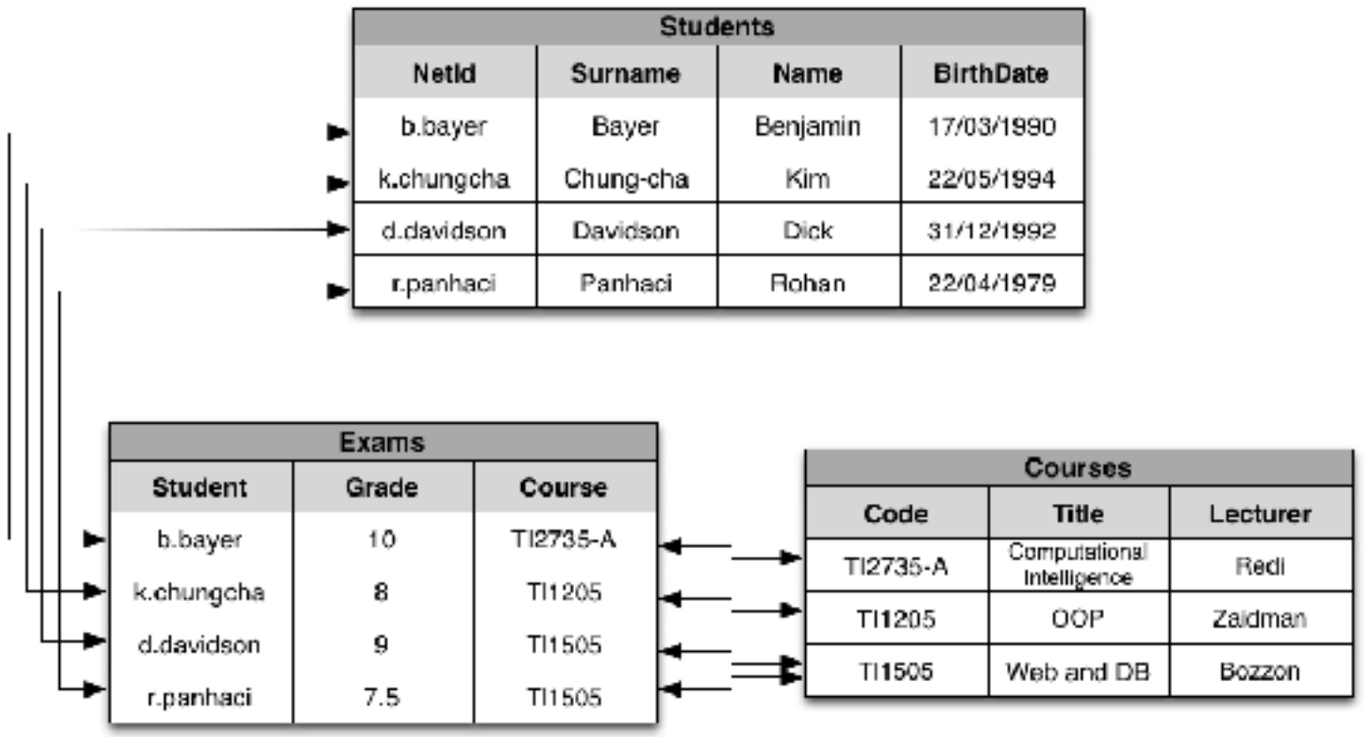
- No primary key value can be NULL
  - No guarantee for unique identification

Students				
NetId	**Surname**	**Name**	**BirthDate**	**DegreeProg**
NULL	Bayer	Benjamin	NULL	Informatica
k.chungcha	Bayer	Kim	17/03/1990	Electrical
k.panhaci	Panhaci	Kim	NULL	NULL
NULL	Panhaci	Kim	31/12/1992	Electrical

- How do we access the first tuple? Are the third and fourth tuple the same?

Value-based References

- The relational model is value-based
- References between data in different relations are represented by means of values of the domains



How to enforce logical relationships between data?

(Referential) Integrity Constraints

- Pieces of data in different relations are correlated by means of values of (primary) keys

- **Referential integrity constraints** are imposed in order to guarantee that the **values refer to <u>actual</u> values** in the referenced relation

A **Referential Integrity Constraint** imposes to the values on a set X of attributes of a relation R_1 to appear as values for the primary key of another relation R_2

1:00

A primary key must be…

A. *Not NULL*
B. *Unique*
C. *Not NULL <u>OR</u> Unique*
D. *Not NULL <u>AND</u> Unique*

flickr@nasacommons

1:00

A primary key must be…

A. *Not NULL*
B. *Unique*
C. *Not NULL <u>OR</u> Unique*
D. *Not NULL <u>AND</u> Unique*

flickr@nasacommons

A Database with Referential Constraints

Offences				
Code	**Date**	**Officer**	**Dept**	**Registration**
143256	25/10/2012	567	75	5694 FR
987554	26/10/2012	456	75	5694 FR
987557	25/10/2012	456	75	6544 XY
630876	15/10/2012	456	47	6544 XY
539856	12/10/2012	567	75	6544 XY

Officers		
RegNum	**Surname**	**FirstName**
567	Brun	Jean
456	Larue	Henri
638	Larue	Jacques

Cars			
Registration	**Dept**	**Owner**	**...**
6544 XY	75	John Doe	...
7122 HT	75	John Doe	...
5694 FR	75	Jane Smith	...
6544 XY	47	J.J. Wilde	...

A Database with Referential Constraints

Offences

Code	Date	Officer	Dept	Registration
143256	25/10/2012	567	75	5694 FR
987554	26/10/2012	456	75	5694 FR
987557	25/10/2012	456	75	6544 XY
630876	15/10/2012	456	47	6544 XY
539856	12/10/2012	567	75	6544 XY

Officers

RegNum	Surname	FirstName
567	Brun	Jean
456	Larue	Henri
638	Larue	Jacques

Cars

Registration	Dept	Owner	...
6544 XY	75	John Doe	...
7122 HT	75	John Doe	...
5694 FR	75	Jane Smith	...
6544 XY	47	J.J. Wilde	...

A Database with Referential Constraints

Referencing Relation

Offences				
Code	**Date**	**Officer**	**Dept**	**Registration**
143256	25/10/2012	567	75	5694 FR
987554	26/10/2012	456	75	5694 FR
987557	25/10/2012	456	75	6544 XY
630876	15/10/2012	456	47	6544 XY
539856	12/10/2012	567	75	6544 XY

Officers		
RegNum	**Surname**	**FirstName**
567	Brun	Jean
456	Larue	Henri
638	Larue	Jacques

Cars			
Registration	**Dept**	**Owner**	**...**
6544 XY	75	John Doe	...
7122 HT	75	John Doe	...
5694 FR	75	Jane Smith	...
6544 XY	47	J.J. Wilde	...

Referenced Relation

Foreign Key

A Foreign Key specify a referential integrity constraint between two relations schemas R_1 and R_2

A set of attributes *FK* in relation scheme R_1 is a **foreign key** of R_1 that **references** relation R_2 if it satisfies the following rules:

Attributes in *FK* have the same domain(s) as the primary key attributes *PK* of R_2

A value of *FK* in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of *PK* for some tuple t_2 in the current state $r_2(R_2)$ or is *NULL*

Advantages of Value-based Structure

- Independence of physical structures
- Only data that is relevant from the application point of view
  - Pointers usually exist at the physical level, but they are not visible at the logical level
  - And, logically, they are not oriented
- Easy transferability of data between systems

Integrity Constraints Can Get Intricate

Accidents				
Code	**Dept1**	**Registration1**	**Dept2**	**Registration2**
6207	75	6544 XY	93	9775 GF
6974	93	5694 FR	93	9775 GF

Cars			
Registration	**Dept**	**Owner**	**...**
7122 HT	75	John Doe	...
5694 FR	93	John Doe	...
9775 GF	93	Jane Smith	...
6544 XY	75	J.J. Wilde	...

- In the example above we have two referential constraints
  - From Registration1, Dept1 to Cars
  - From Registration2, Dept2 to Cars

- A *Foreign Key* can also refer to its own relation

A word on Semantic Integrity Constraints

How can we express the following constraint?

The salary of an employee should not exceed the salary of the employee's supervisor

1:00

Which of the following statements is true?

A. *The relational model is value-based*
B. *The relational model never contains unspecified information*
C. *A key is a set of attributes that uniquely identifies value based* relations
D. *The same tuple can appear more than once in a relation*

flickr@nasacommons

1:00

Which of the following statements is true?

A. *The relational model is value-based*
B. *The relational model never contains unspecified information*
C. *A key is a set of attributes that uniquely identifies value based* relations
D. *The same tuple can appear more than once in a relation*

flickr@nasacommons

Read Operation

Receipts		
Number	**Date**	**Total**
2200	13/11/2013	23.50
2243	14/11/2013	24
4394	14/11/2013	25

Details			
Number	**Quantity**	**Item**	**Line**
2220	2	I001	1
2220	3	I002	2
2220	1	I003	3
2220	1	I004	4
2220	2	I006	5
2243	3	I001	1
2243	3	I002	2
2243	2	I005	3
4394	2	I001	1
4394	2	I002	2
4394	2	I003	3
4394	2	I006	4

Item		
Number	**Description**	**Cost**
I001	Covers	1.00
I002	First Course	3.00
I003	Bream	2.50
I004	Salad	1.00
I005	Steak	6.00
I006	Coffee	1.00

- Extract the Receipt of the current week with higher total
- Order Items by number of orders
- How many Coffees have been sold in October?

- **MORE IN NEXT LESSON**

Create Operation

- Provides a list of attribute values for a new tuple *t* that is to be inserted into a relation *R*

- Can **violate any** of the previously defined constraints
  - Domain, Key, Entity Integrity, Referential Integrity

- If an insertion violates one or more constraints Default option is to **reject the insertion**

Update Operation

- Necessary to specify a condition on attributes of relation
  - Select the tuple (or tuples) to be modified

- If attribute not part of a primary key nor of a foreign key
  - Usually causes no problems

- Updating a primary/foreign key
  - Similar issues as with Insert/Delete

Delete Operation

- Can violate **only referential integrity**
- If tuple being deleted is referenced by foreign keys from other tuples
- **Restrict**: reject the deletion
- **Cascade**: Propagate the deletion by deleting tuples that reference the tuple that is being deleted
- Set **NULL** or Set **DEFAULT**: Modify the referencing attribute values that cause the violation

Violating Referential Constraints

- Operations of the relational model can be categorised into **retrievals** and **updates**
- Basic operations that change the states of relations in the database:
  - **C**reate (or Insert)
  - **R**ead – **NO STATE CHANGE**
  - **U**pdate (or Modify)
  - **D**elete

Today we covered

- Main features of DBMS
- Pros & Cons of DBMS
- Data Models and Data Independence Principle
- Languages of DBMS
- The Relational Model

Readings

- **Book**
  - *Chapter 1:* Introduction to Databases
  - *Chapter 2:* Overview of Database Languages and Architectures
  - *Chapter 3:* The basic (flat) Relational Model

- **Suggested Readings on Blackboard**

End of Lecture