CSE1500 - WEB AND DATABASE TECHNOLOGY

DB LECTURE 2

THE RELATIONAL MODEL

Alessandro Bozzon

cse1500-ewi@tudelft.nl

A STORY OF PAIN AND SUFFERING…

https://www.theguardian.com/info/2018/nov/30/bye-bye-mongo-hello-postgres

AT THE END OF THIS LECTURE, YOU SHOULD BE ABLE TO....

- **Enumerate** and **define** the main elements of the Relational Model

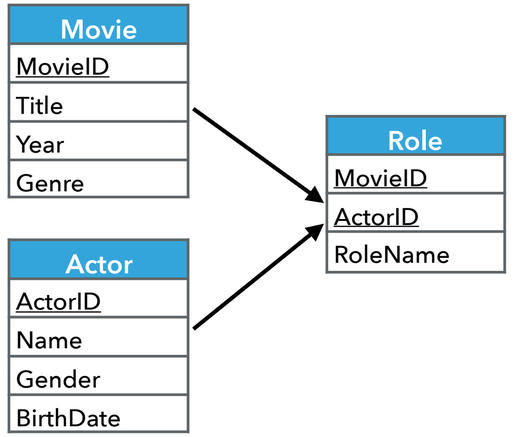
RECAP

1970s – RELATIONAL MODEL

- Ted Codd, mathematician at IBM (Turing Award 1981)
  - A declarative model based on the mathematical notion of *relation*, a relational algebra, and a relational calculus – to express extremely complex queries
  - The relational model satisfies data independence requirements
    - Programmer "declared" data entities and relationships, the computer would do the rest
- Abstraction: logical and physical structure of the database are decoupled
  - Data stored in simple structures
  - Access data through high-level language
  - Physical storage left up to implementation
- Made available in commercial DBMSs in early 1980s
  - It is not easy to implement data independence efficiently and reliably!

Relational

Schema



Instance

MovieID	Title	Year	Genre
1	The Matrix	1999	Action
2	The Devil's Advocate	1997	Drama

ActorID	Name	Gender	BirthDate
1	Keanu Reeves	M	02-09-1964
2	Al Pacino	M	25-04-1940

MovieID	ActorID	Role
2	1	Kevin Lomax
2	2	John Milton

DEFINITIONS

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

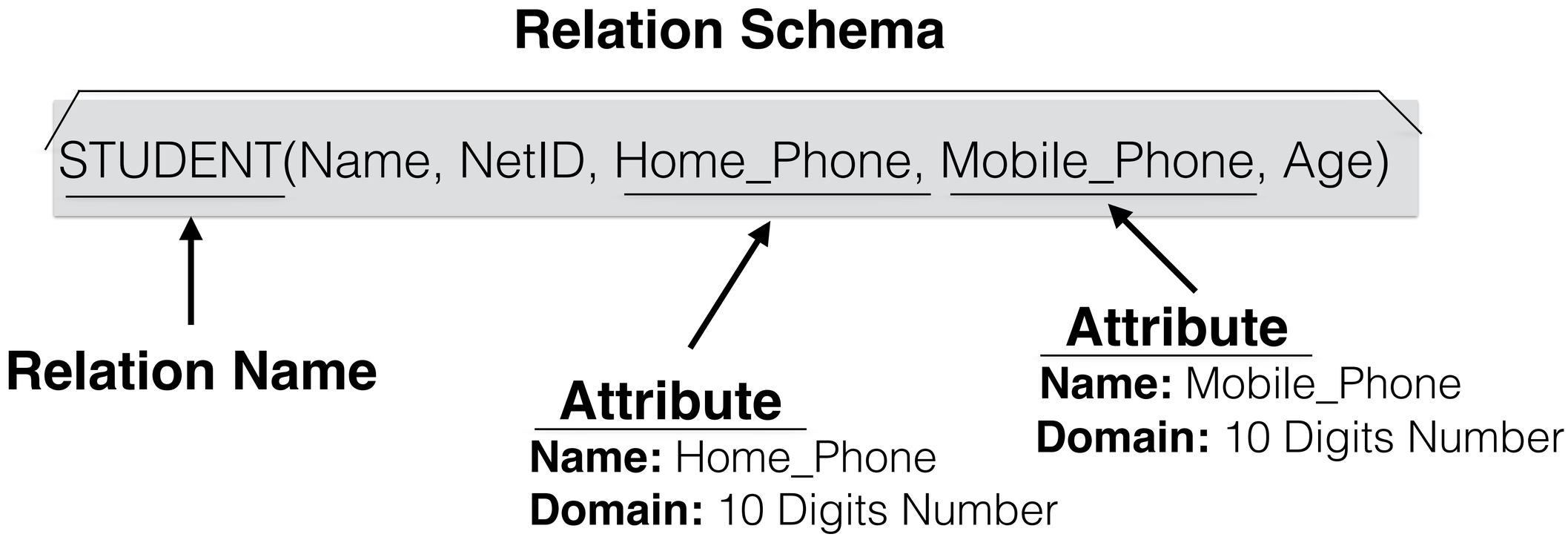
$R(A_1, A_2, \ldots, A_n)$

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



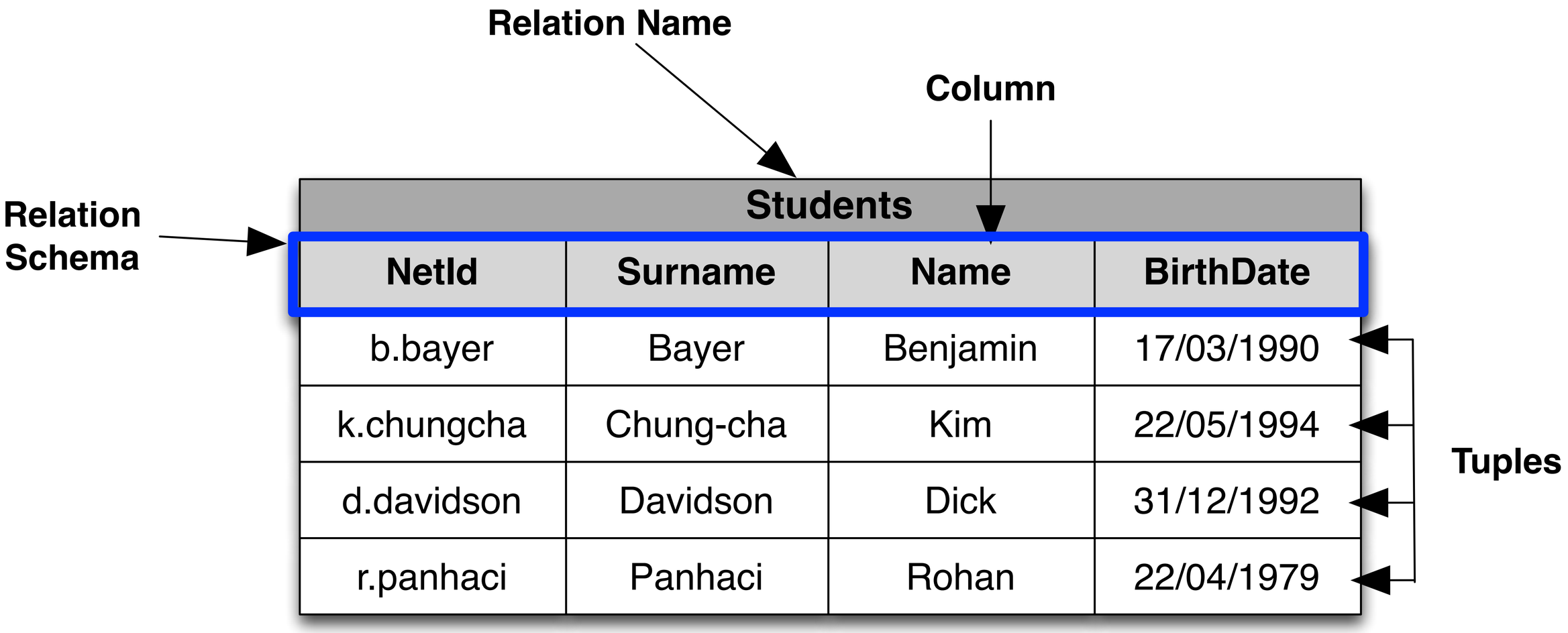
Using this notation, order of attributes matters!

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** $\mathbf{r = \{t_1, t_2, \ldots, t_m\}}$
- Each **n-tuple** t_i is an ordered list of *n* values $\mathbf{t = <v_1, v_2, \ldots, V_n>}$
- 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

- 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

Why?
First Normal Form
assumption

Basis of flat relational model

- For instance, *Name*:
  - “Alessandro”, “Jan”, “Rob” (1 value for each tuple) –> **YES!**
  - “Alessandro;Jan;Rob” (many values in single tuple) –> **NO!**

MULTIPLE RELATIONS

- A database has several relations
- Tuples in such relations are often related in various ways
- Relations can represent facts about entities, or about relationships

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

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

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*

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*

1. Same relation name
2. Attributes in the same order

NULL VALUES

UNKNOWN VALUES

- The relational model imposes 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

Amusing example of difference between '0' and ‘null’
Source: unknown

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

EXAMPLE

- (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 computer 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!

Amusing example of difference between '0' and ‘null’
Source: unknown

TYPES OF NULL VALUES

At least three:

1. A value is **unknown** (exists but it is not known)
   - E.g. a person's birth date is not known

2. 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

3. 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 **not available** interpretation
- We could (and often should) put restrictions on the presence of NULL values in tuples
  - we will see later with SQL)

CONSTRAINTS

WHAT MAKES THE FOLLOWING DATABASE INSTANCE MEANINGLESS (IN THE CONTEXT OF A DUTCH UNIVERSITY)?

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

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

A property in the real world to be modelled by a 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

TYPE OF CONSTRAINTS

There are 4 types of **constraints** that restrict the values in the database

- Model-based (**implicit**) constraints (e.g. no duplicates)
- Schema-based (**explicit**) constraints –> SEE NEXT
- Application-based (**semantic**, **business**) constraints
  - Difficult to express or enforce in the data model
- **Data dependencies** (functional and multivalued)
  - covered in another course

A RDBMS system makes sure that all the constraints are satisfied

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 (key) 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

- 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

WHICH OF THE FOLLOWING IS NOT AN IMPLICIT RELATIONAL CONSTRAINT?

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

WHICH OF THE FOLLOWING IS NOT AN IMPLICIT RELATIONAL CONSTRAINT?

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

KEY CONSTRAINTS

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

CONSIDERING THE RELATION IN THE PICTURE, WHICH OF THE FOLLOWING SETS OF ATTRIBUTES ARE VALID SUPERKEYS?

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

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

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

CONSIDERING THE RELATION IN THE PICTURE, WHICH OF THE FOLLOWING SETS OF ATTRIBUTES ARE VALID SUPERKEYS?

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

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

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

1. **All** *NetIds* are unique
2. As all *NetIds* are unique, any set of attributes with *NetId* is a superkey
3. There are duplicate values for *{Name, Surname}*
4. There are no duplicate values for *{Name, Surname, Birthdate}*, so **4.** could be considered a superkey (though a bad one at schema level)

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 underlined
  - Other candidate keys might be designated as unique key

Students				
NetId	**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

EXAMPLE (FROM BOOK)

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 3.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

ENTITY INTEGRITY CONSTRAINT

- Primary key value **cannot** 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?

A PRIMARY KEY MUST BE…

1. Not NULL
2. Unique
3. Not NULL OR Unique
4. Not NULL AND Unique

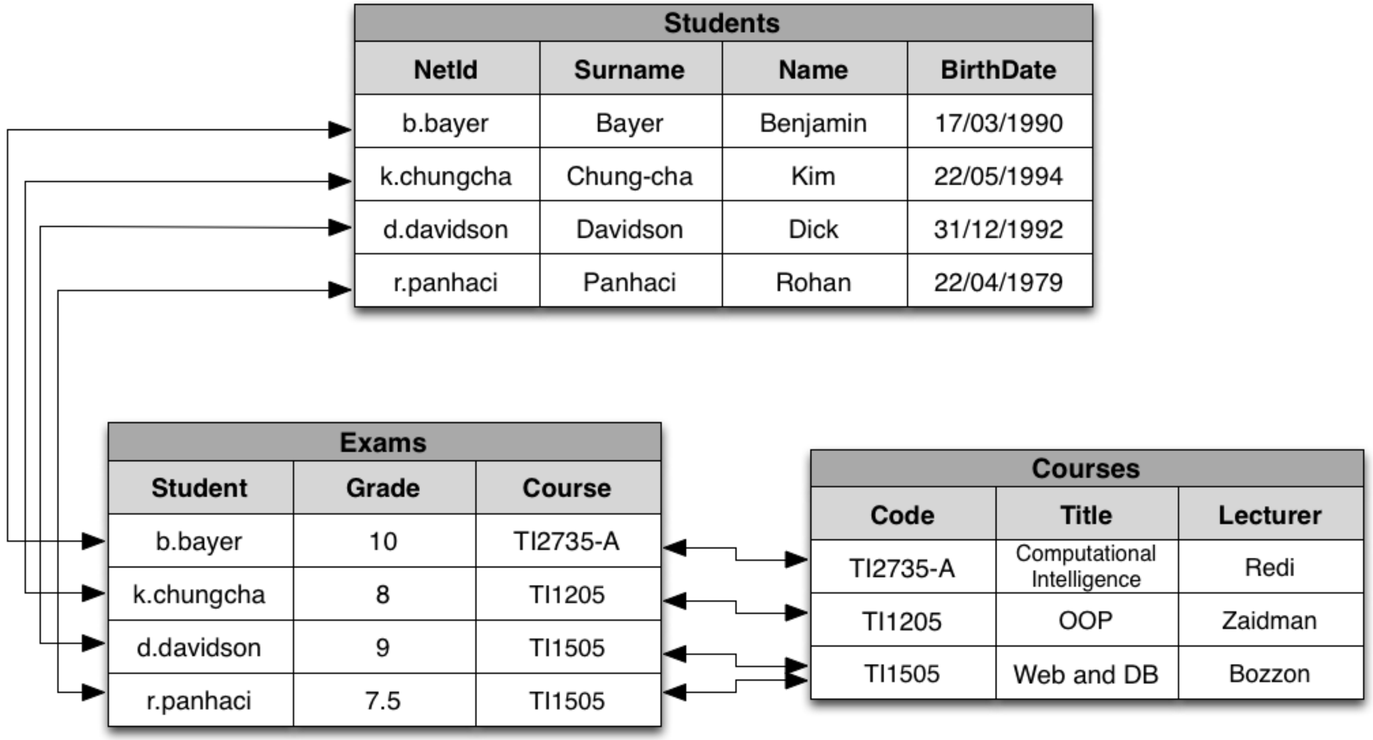
A PRIMARY KEY MUST BE…

1. Not NULL
2. Unique
3. Not NULL OR Unique
4. Not NULL AND Unique

REFERENTIAL INTEGRITY CONSTRAINTS

VALUE-BASED REFERENCES

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



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
TI2735-A	Computational Intelligence	Redi
TI1205	OOP	Zaidman
TI1505	Web and DB	Bozzon

How to enforce logical relationships between data?

VALUE-BASED CONSTRAINTS

- Pieces of data in different relations are related by means of values of (primary) keys
- Referential integrity constraints are imposed in order to guarantee that the values refer to actual 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

A DATABASE WITH REFERENTIAL INTEGRITY 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** *specifies* a referential integrity constraint between two relation schemas R_1 and R_2

A set of attributes *FK* in relation 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
- It holds 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

WHICH TYPE OF CONSTRAINTS ALLOWS THE ENFORCEMENT OF THE FOLLOWING REAL-WORLD CONSTRAINT?

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

A. ***Domain Constraint***
B. ***Referential Constraint***
C. ***Cross-relation Constraint***
D. ***None of the above***

WHICH TYPE OF CONSTRAINTS ALLOWS THE ENFORCEMENT OF THE FOLLOWING REAL-WORLD CONSTRAINT?

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

A. *Domain Constraint*
B. *Referential Constraint*
C. *Cross-relation Constraint*
D. *None of the above*

Such constrain can be specified only within the application program that updates the database, by using a general purpose constraint specification language.

CRUD AND CONSTRAINTS

VIOLATING (REFERENTIAL) CONSTRAINTS

- Operations of the relational model can be categorised into retrievals and transactions
- 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

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, the *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

WRAPPING UP

TODAY WE COVERED

- The Relational Model

END OF LECTURE