

# The Relational Model

Mahmoud El-Haj

# The Relational Model

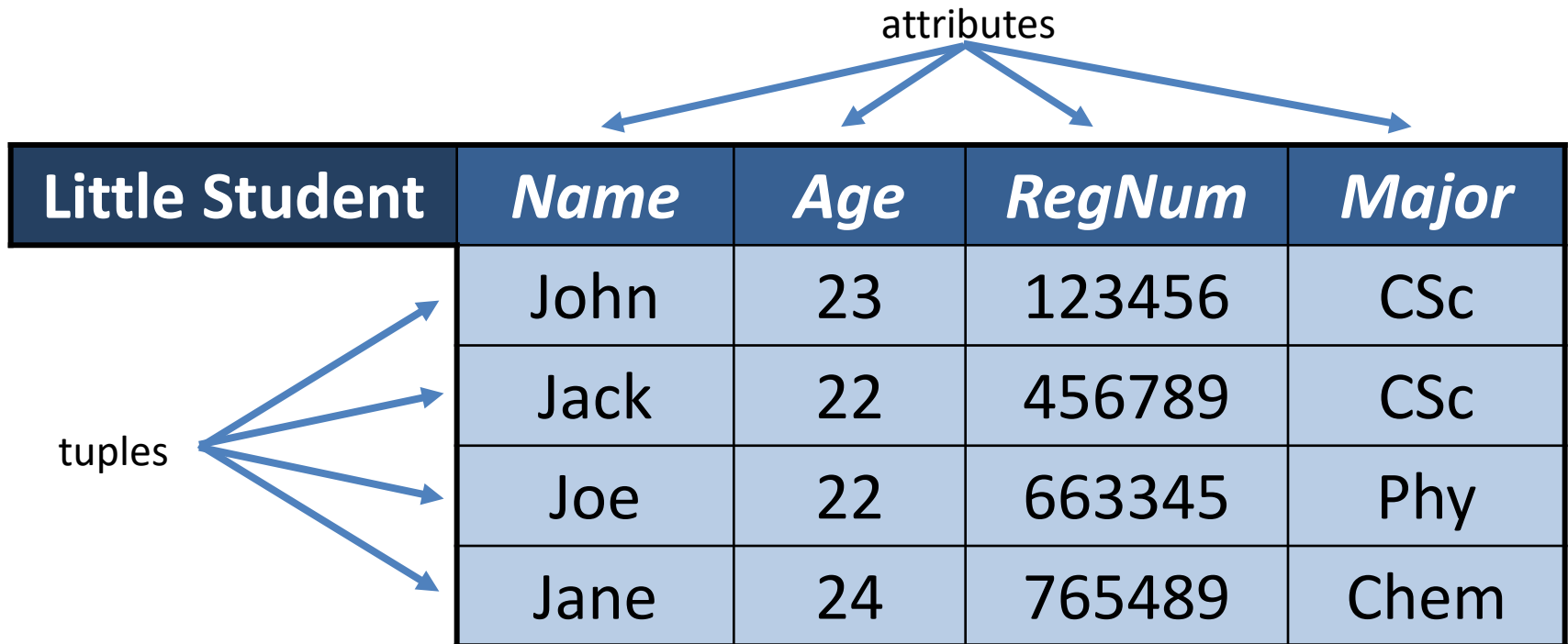
- Introduced by Edgar Codd of IBM in 1970 <sup>1</sup>.
- It's a mathematical concept known as relation.
  - Like a table of values as its basic building block
  - Implemented in a large number of commercial systems over the last twenty years or so.
- An attempt to describe data by its natural structure only
  - i.e. without imposing any additional structure for machine representation purposes
- See earlier database models: hierarchical and network (both displaced by relational database).

<sup>1</sup> [http://en.wikipedia.org/wiki/Edgar\\_F.\\_Codd](http://en.wikipedia.org/wiki/Edgar_F._Codd)

# Relations (1)

- May be thought of as table of values
  - Merely for discussion purposes
  - In a relational database, all data is held in **tables**, which are made up of **rows** and **columns**.
- Each row in table represents collection of related data values (e.g. an entity or relationship)
- Table name and column names are used to help interpret meaning of values in each row
  - e.g. a Little Students table with columns Name, Age, RegNum (Registration Number), Major

# Relations (2)



- A row is called a **tuple**.
- A column header is called an **attribute**.
- The **data type** describing the types of values that can appear in each column is called a **domain (type)**

# Domain

- A set of *atomic* values
  - Atomic means each value in domain is indivisible (Colour="red" but not Colour="read","blue") as far as relational model is concerned (e.g. values that are strings, integers, etc.)
- The domain of an attribute  $A$  is denoted by  $dom(A)$ . For example,
  - $dom(Major) = \{ CSc, Phy, Chem, Bio, Env \}$
  - $dom(Age) = \{ N \}$  where  $16 \leq n \leq 80$  and  $n \in N$

Little Student	Name	Age	RegNum	Major
	John	23	123456	CSc
	Jack	22	456789	CSc
	Joe	22	663345	Phy
	Jane	24	765489	Chem

# Data Types for Domains (1)

- A domain has a data type associated with it, e.g.
  - Student age must be an **Integer**
  - Student name must be a **String**
- A format can also be specified for a domain e.g.
  - UK post codes must be in the format  $\$ \$ n n n \$ \$$   
(where  $\$$  represents characters and  $n$  integer numbers) <sup>1</sup>
- Standard numeric data types
  - Integers: short integer, integer, long integer
  - Real numbers: float, double-precision float

<sup>1</sup> *A little more complex in reality*

# Data Types for Domains (2)

- Characters
- Strings: fixed length, variable length
- Date, time, timestamp (includes fractions of seconds) and money
- Others may be described as:
  - a sub range of values from a data type
  - an enumerated type where all possible values are explicitly listed

# Relations and Attributes

- Relation schema is used to describe a relation
- Relation schema  $R$  denoted by  $R ( A_1, A_2, \dots, A_n )$ 
  - Relation name  $R$
  - List of attributes  $A_1, A_2, \dots, A_n$
- Each attribute  $A_i$  is a name of role played by some domain  $D$  in relation schema  $R$ 
  - i.e. subset of integers used as domain for Age;  
role being played is age of student in question
- So we have...      Students ( Name, Age, RegNum, Major )

<b>Students</b>	<b>Name</b>	<b>Age</b>	<b>RegNum</b>	<b>Major</b>
-----------------	-------------	------------	---------------	--------------



# Degree

- The degree of a relation is the number of attributes  $n$  of its relation schema
  - A relation schema for a relation of degree 4 (describing university students) is:

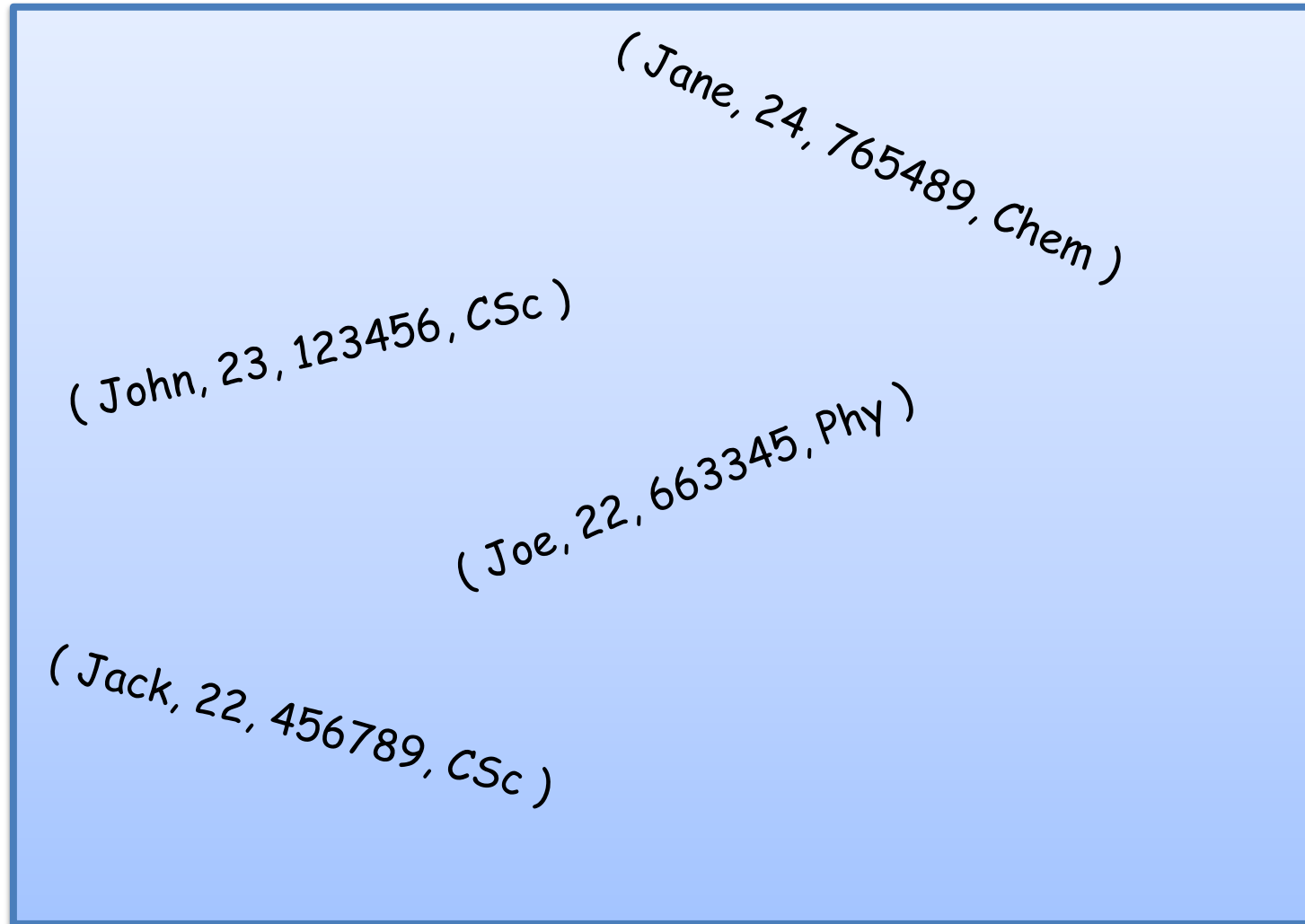
STUDENTS ( Name, Age, RegNum, Major )

<b>Students</b>	<b><i>Name</i></b>	<b><i>Age</i></b>	<b><i>RegNum</i></b>	<b><i>Major</i></b>
-----------------	--------------------	-------------------	----------------------	---------------------

# Tuples (1)

- A **relation instance**  $r$  of the relation schema  $R ( A_1, A_2, \dots, A_n )$ , also denoted by  $r(R)$ , is a set of  $n$ -tuples:  $r = \{ t_1, t_2, \dots, t_m \}$ 
  - For example, in STUDENTS  $t_1$  might be ( John, 23, 123456, CSc ), which is a quadruple

# This is a relation instance $r$ of LITTLE STUDENT



A set containing four quadruples

Type in a tuple for a mixed gender  
football team (TP)

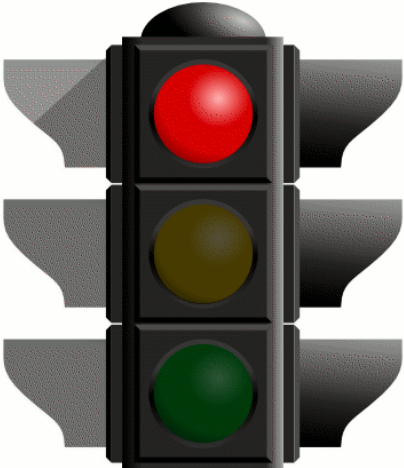
MAHMOUDINHO, 24, FORWARD, ARSENAL, 20M

# Tuples

- Ordering of tuples is unimportant
  - May be logically ordered  
e.g. by Name, Age, RegNum, etc.
  - Could be randomly ordered
  - In practice, when using a relational DBMS, do not make any assumption about order of tuples in relation
  - In mathematical sets, ordering is irrelevant

# Tuples: *this never happens...*

- Note: each tuple is distinct (**unique**)



Little Student	Name	Age	RegNum	Major
	John	23	123456	CSc
	Jack	22	456789	CSc
	Joe	22	663345	Phy
	<del>Jack</del>	<del>22</del>	<del>456789</del>	<del>CSc</del>
	Jane	24	765489	Chem

**NO DUPLICATES!!!**

# Tuples (2)

- Ordering of values in tuple (or columns in relation) is **significant** (*position under correct column heading clearly important*)
  - Each  $n$ -tuple  $t$  is an ordered list of  $n$  values
$$t = \langle v_1, v_2, \dots, v_n \rangle$$
where each value  $v_i$ ,  $1 \leq i \leq n$ , is an element of  $dom(A_i)$  or is a special *null* value
  - The  $i^{th}$  value in tuple  $t$ , which corresponds to attribute  $A_i$ , is referred to as  $t [ A_i ]$

Little Student	Name	Age	RegNum	Major
$t [ Age ] = 23$	John	23	123456	CSc

# Tuples (3)

Little Student	Name	Age	RegNum	Major
	John	23	123456	CSc

- A tuple could alternatively be considered as a set of (<attribute>, <value>) pairs, where each pair gives mapping from an attribute  $A_i$  to a value  $v_i$  from  $dom(A_i)$

- In this case, ordering of attributes within tuples is not important as attribute name appears with its value in the pair
- Makes sense as there is no reason to prefer having one attribute value appear before another in a tuple
- But please don't get confused between this slide and the previous one!

( RegNum, 123456 )  
( Age, 23 )  
( Major, CSc )  
( Name, John )

**AN ALTERNATIVE  
WAY TO THINK OF  
A TUPLE**



# Summary of notation so far (1)

- A relation schema  $R$  of degree  $n$  is denoted by  
$$R ( A_1, A_2, \dots, A_n )$$

- An  $n$ -tuple  $t$  in a relation is denoted by

$$t = \langle v_1, v_2, \dots, v_n \rangle$$

where  $v_i$  is value corresponding to attribute  $A_i$

–  $t [ A_i ]$  refers to the value  $v_i$  in  $t$  for attribute  $A_i$

# Summary of notation so far (2)

- $t [ A_u, A_w, \dots, A_z ]$ , where  $A_u, A_w, \dots, A_z$  is a list of attributes from  $R$ , refers to the sub-tuple of values  $\langle v_u, v_w, \dots, v_z \rangle$  from  $t$  corresponding to the attributes specified in the list

Little Student	<i>Name</i>	<i>Age</i>	<i>RegNum</i>	<i>Major</i>
	John	23	123456	CSc

$\dagger [ \text{Major}, \text{Age} ] = \langle \text{CSc}, 23 \rangle$

# Superkey

- Remember: all tuples in a relation are **distinct**
  - No two tuples can have the same combination of values for all their attributes
- What defines this uniqueness?  $r = \{ t_1, t_2, \dots, t_m \}$
- Usually there are **subsets of attributes** of a relation schema  $R$  with the property that no two tuples in any relation state  $r$  of  $R$  can have the same combination of values for these attributes
  - Such a subset of attributes is called a superkey

A relation state  $r$  of  $R$  can be thought of as an instance of  $R$

# Superkey (2)

- More formally, a superkey  $SK$  of a relation schema  $R$  is a subset of attributes so that for any two distinct tuples  $t_1$  and  $t_2$  in a relation state  $r$  of  $R$

$$t_1 [ SK ] \neq t_2 [ SK ]$$

**THIS IS BECAUSE  
ALL TUPLES MUST  
BE UNIQUE**

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

The set of attribute values within a superkey uniquely identifies a tuple

# Querying a Relation

...In its broadest sense, if we know both:

- The domains (and their ordering) of a relation (i.e., the generic field names at the top of the columns – the “schema” for the table)
- The values of the domains (or fields) for a specific tuple (or set of tuples) then we can examine the relation for any tuples which have those specific values in those domains.

Let us take as our example an extended version of the Students relation

# Students relation

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapeemapetilon	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

First Name (FName), Last Name (LName), Department Name (DName)

# Query Template

- If we know the last name of a student (say Burns) we can find out his associated *Dname*
- We can write down the values we know (and don't know!) as a *Query Template*
  - We use question marks to represent the values we do not know (and may wish to find out)

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
?	Burns	?	?	?

A possible text representation: (Lname = "Burns")

## Example One

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
?	Burns	?	?	?

- We examine the table for any tuple having “Burns” as its *Lname* and read the values for *Fname*, ..., *Dname*
- We see the eighth tuple has “Burns” in the appropriate domain and the associated *Dname* is Physics.

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics



<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
?	<b>Burns</b>	?	?	?

- We talk about the query template matching a tuple; where question marks match any value in a field (act as a wild card)
- When the fields in the template match the tuple's fields, we have a match or a hit

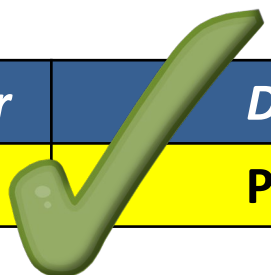
<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

# Tuple Matching

- All fields must match to be a *tuple match*
  - Taking each row (tuple) in turn...

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
?	Burns	?	?	?
YES	NO	YES	YES	YES

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Montgomery	Burns	8888	M	Physics
?	Burns	?	?	?
YES	YES	YES	YES	YES



<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
?	<b>Simpson</b>	?	?	?

## Example Two

- The result of our question  
 “What *Dname* does Simpson have?” involves two tuples
  - Therefore two answers : **Computer Science** and **Physics**.

- Another question with multiple answers would be  
 “What students are male?”
  - Query template  
 (? , ? , ? , M , ?)

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

# Looking for Superkeys

- Remember, a superkey identifies a single tuple
  - Let's try "Lname". The query (Lname = "Simpson") we hit two tuples
  - Similarly, (Lname = "Flanders") will hit two tuples

- So "Lname"  
**is not**  
a superkey

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

# Looking for Superkeys

- Let's try "Fname". The query (Fname = "Bart") hits one tuple
- (Fname = "Rod"), again one tuple
  - This is looking good, for the example data we have

- **BEWARE:**  
How likely is it that a realistic database would contain only students with unique first names?
  - i.e. "John"

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

# Looking for Superkeys

- Let's try "Fname" and "Lname". The queries
  - (Fname = "Bart", Lname = "Simpson") will hit one tuple
  - (Fname = "Rod", Lname = "Flanders"), again one tuple

- Looks good, for the example data here. But will this be true for all time and for all possible data?

*...at one point there were 7 James Taylor at Lancaster University*

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

# Looking for Superkeys

- Let's try "Lname" and "RegNum". The queries
  - (Lname = "Simpson", RegNum = 1111) that will hit one tuple
  - (Lname = "Flanders", RegNum = 6666), again one tuple

- Looks good for the example data here. But will this be true for all time and all possible data?

<i>FName</i>	<i>Lname</i>	<u><i>RegNum</i></u>	<i>Gender</i>	<i>Dname</i>
Bart	Simpson	1111	M	Computer Science
Lisa	Simpson	2222	F	Physics
Milhouse	Van Houten	3333	M	History
Ralph	Wiggum	4444	M	History
Todd	Flanders	5555	M	Religious Studies
Rod	Flanders	6666	M	Religious Studies
Apu	Nahasapee...	7777	M	Computer Science
Montgomery	Burns	8888	M	Physics

# Key

- A superkey can have redundant attributes; attributes that do not contribute to the property of unique identification
- A key  $K$  of a relation schema  $R$  is a superkey of  $R$  with the additional property that **removing any attribute  $A$  from  $K$  leaves a set of attributes  $K'$  ( $K$  prime) that is not a superkey of  $R$** 
  - A key is a minimal superkey, for example
  - { RegNum } is a **key** of STUDENT
  - { FName, RegNum, Dname } is a superkey but not a key  
...if we get rid of either “Fname” or “Dname” or both, what’s left will still act as a superkey



# Candidate Key

- If a relation schema has more than one key, each of the keys is called a *candidate key*
  - If relation schema for STUDENT had additional attribute NI\_Num then both { RegNum } and { NI\_Num } would be candidate keys of STUDENT
    - NI\_Num: UK National Insurance Number

Think of it as:

RegNum and NI\_Num are “artificial” or “generated” so we can ensure they are unique

# Primary Key

- One of the candidate keys of a relation is chosen as the *primary key*
  - This is the candidate key whose values are used to identify tuples in a relation

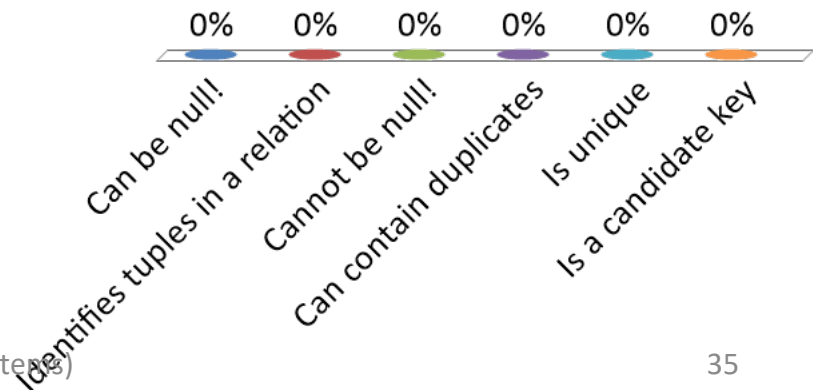
- Attributes that define the primary key of a relation schema are underlined, for example

LITTLE STUDENT ( Name, Age, RegNum, Major )

- No primary key value can be **NULL**
  - The value is used to identify individual tuples in a relation

Which of the following are correct about a primary key, choose more than one? (TP)

- A. Can be null!
- B. Identifies tuples in a relation
- C. Cannot be null!
- D. Can contain duplicates
- E. Is unique
- F. Is a candidate key



# For you to come back to

Self  
learning

BASIS FOR COMPARISON	SUPERKEY	CANDIDATE KEY
Basic	A single attribute or a set of attributes that uniquely identifies all attributes in a relation is Superkey.	A proper subset of a Superkey, which is also a Superkey, is a candidate key (remember if a subset of the candidate key is a Superkey then it is not a candidate key).
One in other	It is not compulsory that all Superkeys will be candidate keys.	All candidate keys are Superkeys.
Selection	The set of Superkeys forms the base for selection of <b>candidate keys</b> .	The set of candidate keys form the base for selection of <b>a single primary key</b> .
Count	There are comparatively more Superkeys in a relation.	There are comparatively less candidate keys in a relation.

# Which key is which?

StudentID	REG_ID	Name	Major	Email
1	CS-2019-55	Amy	Computer Science	amyOne@xyz.com
2	CS-2018-12	Salma	Computer Science	salma@xyz.com
3	Phy-2019-65	Amy	Physics	amyTwo@xyz.com
4	Eng-2019-23	Angelina	Engineering	angelina@xyz.com

Give this a try later given the explanation in the previous slide:

- List all possible **Super Keys**
- From those find which ones can be **candidate keys**
- And finally which of the candidate keys can be chosen as the **primary key**

# Relational Database Schema (1)

- So far we have discussed single relations and single relation schemas
- A relational database schema is a set of relation schemas  $S = \{ R_1, R_2, \dots, R_m \}$  and a set of integrity constraints
  - We will come back to integrity constraints later

# Relational Database Schema (2)

STUDENT

<i>FName</i>	<i>MName</i>	<i>LName</i>	<u><i>RegNum</i></u>	<i>BDate</i>	<i>Address</i>	<i>Gender</i>	<i>DName</i>
--------------	--------------	--------------	----------------------	--------------	----------------	---------------	--------------

DEPARTMENT

<u><i>DName</i></u>	<i>HoD</i>	<i>NoOfEmps</i>
---------------------	------------	-----------------

DEPT\_LOCATIONS

<u><i>DName</i></u>	<u><i>DLocation</i></u>
---------------------	-------------------------

Note two attributes in Key

We underline all Primary Key attributes

# Foreign Key (1)

- Sometimes a tuple in one relation refers to a tuple in another relation. Such references are maintained using the concept of a *Foreign Key*
- A set of attributes FK in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies following:
  - **Rule 1:** Attributes in FK have **same domain(s)** as *primary key* attributes PK of  $R_2$  (basically a primary key of R2 becomes a foreign key in R1)
  - **Rule 2:** A **value** of FK in tuple  $t_1$  of current state  $r_1(R_1)$  either occurs as a value of PK for some tuple  $t_2$  in current state  $r_2(R_2)$  or is NULL

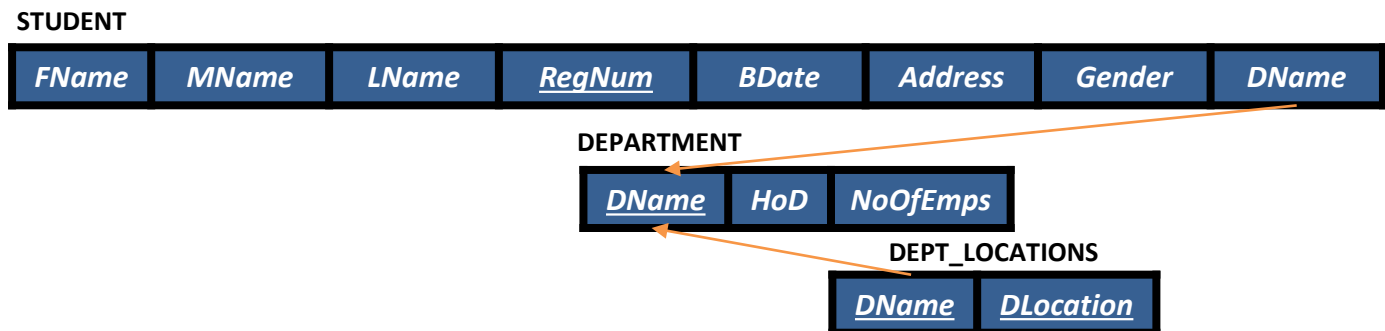
In former case  $t_1 [ FK ] = t_2 [ PK ]$

The tuple  $t_1$  references tuple  $t_2$   
 $R_1$  is a referencing relation and  
 $R_2$  is a referenced relation

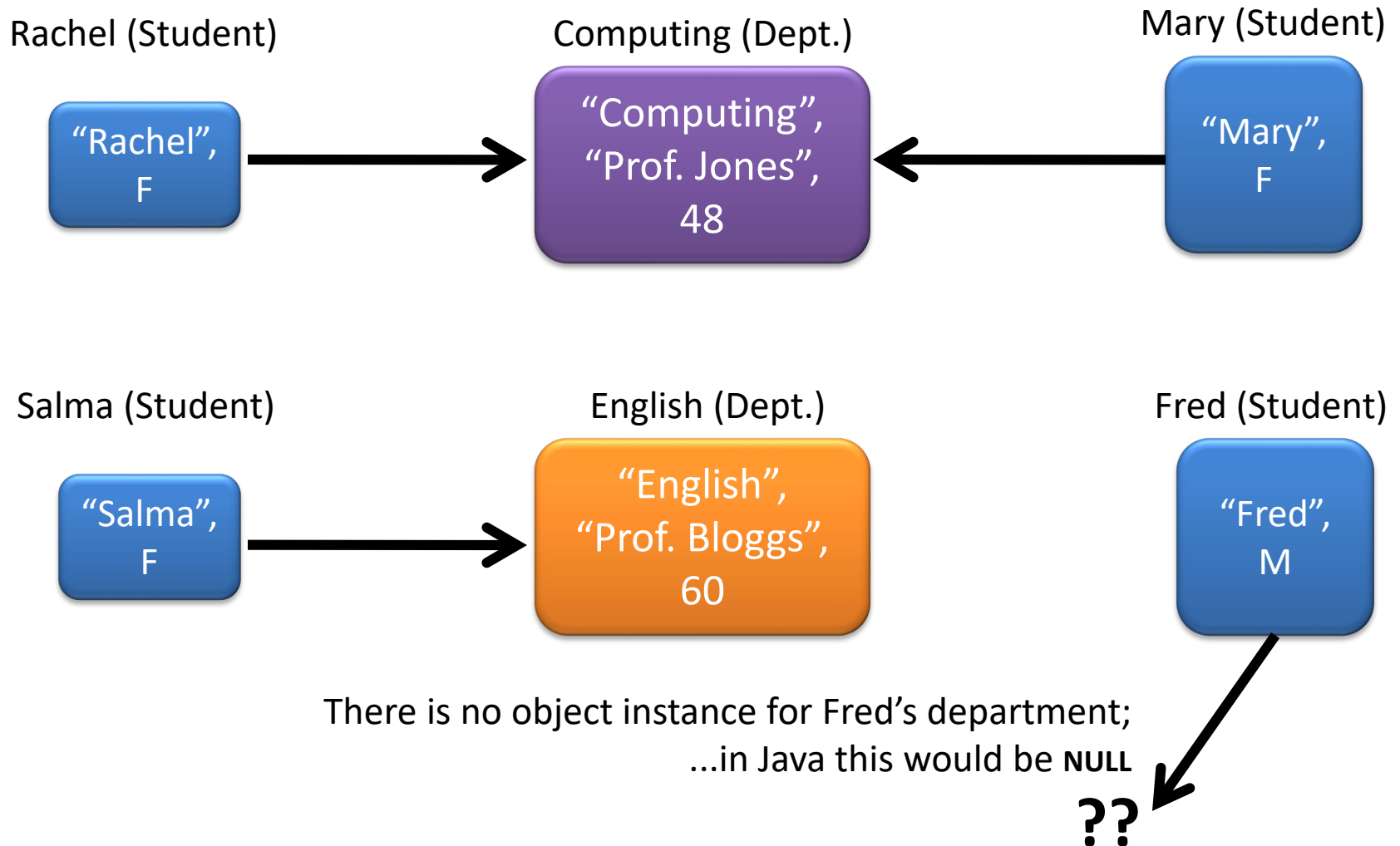


# Foreign Key (2)

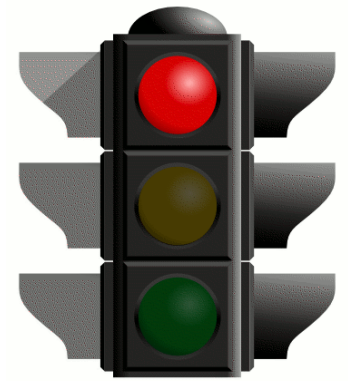
- DName is foreign key for relation STUDENT
- Dname is foreign key for relation DEPT\_LOCATIONS
- Rule 2 on previous slide ensures *referential integrity*
  - Ensuring a tuple in one relation that refers to another relation **must refer to an existing tuple** in that relation



# Symbolic Links

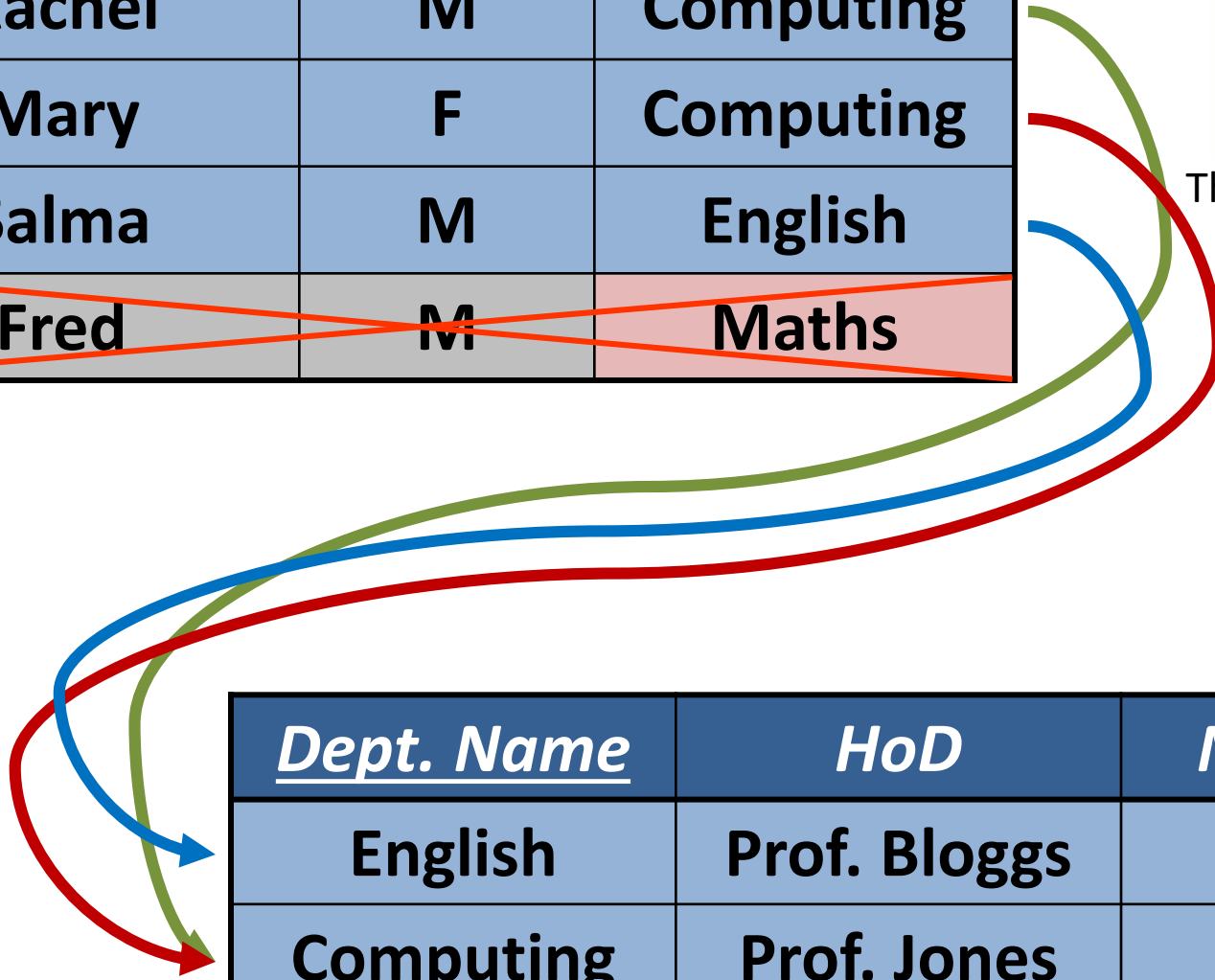


<u>Stud. Name</u>	Gender	DName
Rachel	M	Computing
Mary	F	Computing
Salma	M	English
<del>Fred</del>	<del>M</del>	<del>Maths</del>



This never happens!

Can't have a reference to a non-existent foreign key value



<u>Dept. Name</u>	HoD	NoOfStaff
English	Prof. Bloggs	60
Computing	Prof. Jones	48

# Integrity Constraints

- You might have noticed we have already been discussing integrity constraints; we have seen a number...
- Domain constraints
  - Data types
  - Data Formats
  - Range of values
- Keys
  - Uniqueness
  - Non-null primary keys
- Foreign Keys
  - Referential integrity