

# ***Introduction to Database Systems***

## **The Relational Data Model**

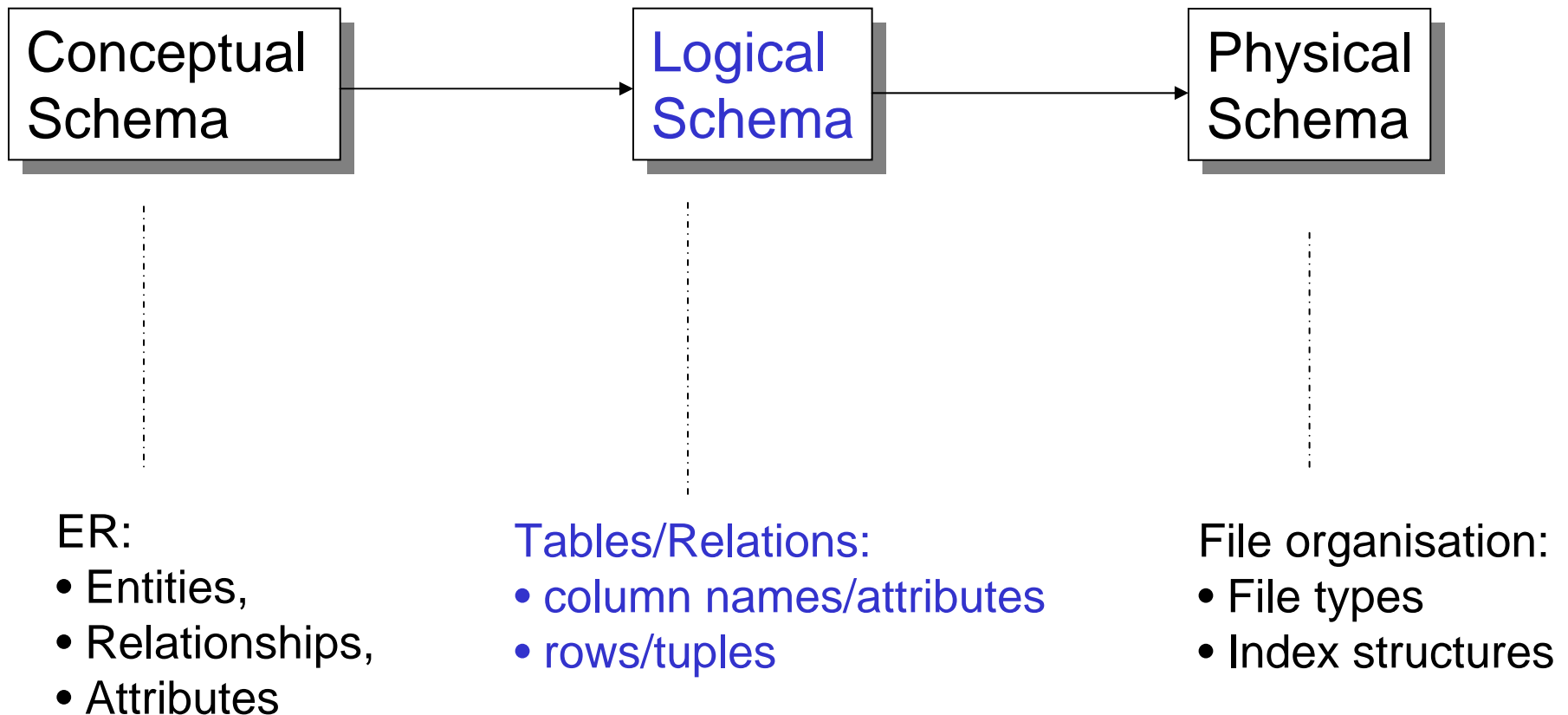
Werner Nutt

# 4. The Relational Data Model

## 4.1 Schemas

1. **Schemas**
2. Instances
3. Integrity Constraints

# Different Schemas are Based on Different Concepts



# A Table

*Table name*

*Column names*

Product:

Name	Price	Category	Manufacturer
gizmo	\$19.99	gadgets	GizmoWorks
Power gizmo	\$29.99	gadgets	GizmoWorks
SingleTouch	\$149.99	photography	Canon
MultiTouch	\$203.99	household	Hitachi

*Rows*

# Review of Mathematical Concepts (1)

- **Sets:**  $S, T, S_1, \dots, S_n, T_1, \dots, T_n, \{ \}$   
Cardinality of a set  $S$  denoted as  $|S|$
- **Cartesian Product** of sets (also *cross product*):  
 $S \times T$  set of all pairs  $(s,t)$  where  $s \in S$  and  $t \in T$   
 $S_1 \times \dots \times S_n$  set of all  $n$ -tuples  $(s_1, \dots, s_n)$   
where  $s_i \in S_i$
- **Relation  $R$**  over  $S, T$ :  
subset of  $S \times T$ , written  $R \subseteq S \times T$   
We write  $(s,t) \in R$  or, equivalently,  $sRt$

# Review of Mathematical Concepts (2)

- **Relation**  $R$  over  $S_1, \dots, S_n$ :

subset  $R \subseteq S_1 \times \dots \times S_n$

The number  $n$  is the **arity** of  $R$

( $R$  is **binary** if  $n=2$  and **ternary** if  $n=3$ )

- **Function**  $f$  from  $S$  to  $T$ , written  $f: S \rightarrow T$

associates to every element  $s \in S$

**exactly** one element of  $T$ , denoted as  $f(s)$

# Review of Mathematical Concepts (3)

- **Partial function**  $f$  from  $S$  to  $T$ , written  $f: S \rightsquigarrow T$   
associates to **some** element  $s \in S$   
**exactly** one element of  $T$ , still denoted as  $f(s)$

We write  $f(s) = \perp$  (read “bottom”)

if  $f$  does not associate any element of  $T$  to  $s$

- A relation  $R$  over  $S_1, \dots, S_n$  is **total** on  $S_i$   
if for every  $s_i \in S_i$   
there are  $s_j \in S_j$ ,  $j \neq i$ , such that  $(s_1, \dots, s_n) \in R$

*In other words:*

*every element of  $S_i$  occurs in some tuple of  $R$*

# Review of Mathematical Concepts (4)

- A relation  $R$  over  $S$  and  $T$  is **functional** in its **first argument** if

$$sRt_1 \text{ and } sRt_2 \text{ implies that } t_1 = t_2$$

for all  $s \in S$ ,  $t_1, t_2 \in T$ .

In other words, for every  $s \in S$ ,

there is at most one  $t \in T$  related by  $R$  to  $s$

- Analogously, a relation  $R$  over  $S_1, \dots, S_n$  can be **functional**
  - in an **argument**  $i$ , or
  - in a **tuple of arguments**, say  $(i, j, k)$



# How Many ... ?

Consider sets

$S, T$  with  $|S| = N$  and  $|T| = M$

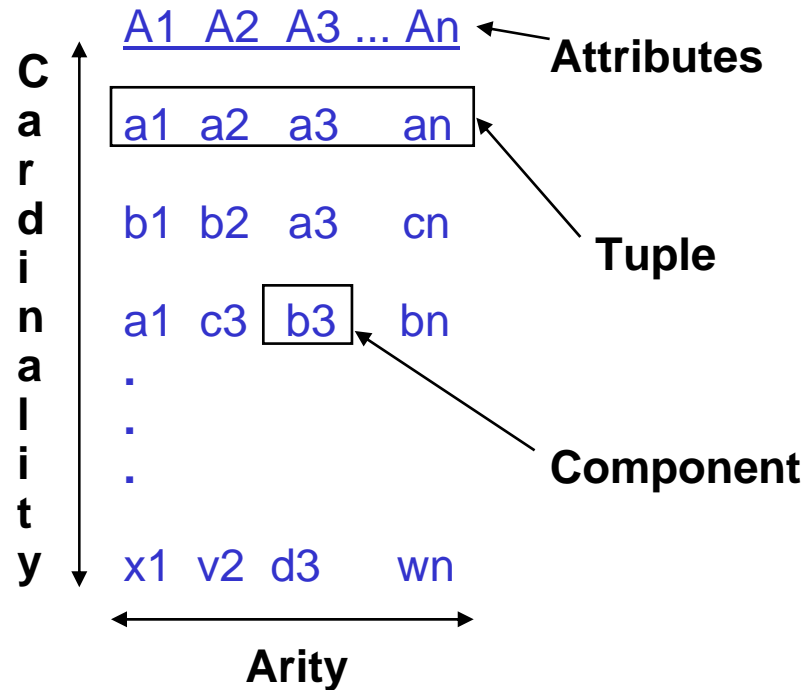
$S_1, \dots, S_n$  with  $|S_i| = N_i$

- How many elements can a relation over  $S_1, \dots, S_n$  have?  
At least? At most?
- How many relations over  $S, T$  are there?  
How many over  $S_1, \dots, S_n$ ?
- How many functions from  $S$  to  $T$  are there?
- How many partial functions from  $S$  to  $T$  are there?

## How Many ... ? (Cntd.)

- How many relations are there over  $S$  and  $T$  that are functional in the first argument?
- How many relations are there over  $S$  and  $T$  that are total on  $S$ ?

# Tables Look Like Relations ...



$\{(a1, a2, a3, \dots, an),$   
 $(b1, b2, a3, \dots, cn),$   
 $(a1, c3, b3, \dots, bn),$   
 $\dots$   
 $(x1, v2, d3, \dots, wn)\}$

*Over which sets does this relation range?*

- In Databases:** Distinguish between
- *Schema* (structure ) and
  - *Instance* (content)

# Relation Schemas

A **relation schema**

$$R(A_1:D_1, \dots, A_n:D_n)$$

consists of

- a *name*, say  $R$
- a nonempty set of *attributes*, say  $A_1, \dots, A_n$
- a *type* or *domain*, say  $D_i = \text{dom}(A_i)$ , for each attribute  $A_i$

**Example:** Product (Prodname: Name,  
Price: DollarPrice,  
Category: Name,  
Manufacturer: Name)

# Types and Domains

**Type:** Class of atomic values, e.g.,

- integers, reals, strings
- integers between 15 and 80, strings of (up to) 50 characters

**Domain:** Set of atomic values, that have a specific meaning in an application, e.g.,

- Name, EmployeeAge
- Domains have a **type**, e.g.,
  - EmployeeAge = Int[15,80]
- Domains may have **default values**

*Domains allow for an additional layer of abstraction*

# 4. The Relational Data Model

## 4.2 Instances

1. Schemas
- 2. Instances**
3. Integrity Constraints

# Relation Schema and Instance

- A tuple of values  $(v_1, \dots, v_n)$  **satisfies** the relation schema  $R(A_1:D_1, \dots, A_n:D_n)$  if  $v_i \in D_i$  ( $i=1, \dots, n$ )
- An **instance** of  $R$  is a set of tuples that satisfy the schema of  $R$  (i.e., a relation over  $D_1, \dots, D_n$ )
- **Analogy** with programming languages:
  - schema = type
  - instance = value
- Important **distinction**:
  - Relation Schema = stable over long periods of time
  - Relation Instance = changes constantly, as data is inserted/updated/deleted

# Example

## Domain declaration:

Name=String(30), DollarPrice=Decimal (10,2),

## Relation schema:

Product(Prodname: Name, Price: DollarPrice,  
Category: Name, Manufacturer: Name)

## Instance:

Prodname	Price	Category	Manufacturer
gizmo	19.99	gadgets	GizmoWorks
Power gizmo	29.99	gadgets	GizmoWorks
SingleTouch	149.99	photography	Canon
MultiTouch	203.99	household	Hitachi



# Database Schema and Instance

## Database Schema

Set of relation schemas, e.g.,

Product (Productname, Price, Category, Manufacturer),

Vendor (Vendorname, Address, Phone), ...

*To keep things simple,  
we have dropped types/domains*

## Database Instance

Set of relation instances,  
one for each relation in the schema

Important *distinction*:

- Database Schema = stable over long periods of time
- Database Instance = changes constantly

# Updates

A database reflects the state of an aspect of the real world:  
The world changes → the database has to change

Updates to an instance:

- 1) **adding** a tuple
- 2) **deleting** a tuple
- 3) **modifying** an attribute of a tuple

*What could be updates to a schema?*

- Updates to the **data** happen very **frequently**.
- Updates to the **schema**: relatively **rare**, rather **painful**.

*Why?*

# Null Values

## Attribute values

- are atomic
- have a known domain
- can sometimes be “*null*”

## Three meanings of null values

1. not applicable
2. not known
3. absent (not recorded)

## Student

studno	name	hons	tutor	year	thesis title
s1	jones	ca	bush	2	<i>null</i>
s2	brown	cis	kahn	2	<i>null</i>
s3	smith	<i>null</i>	goble	2	<i>null</i>
s4	bloggs	ca	goble	1	<i>null</i>
s5	jones	cs	zobel	1	<i>null</i>
s6	peters	ca	kahn	3	“A CS Survey”

# Order and Duplication

## In tables:

- *Order of attributes* is fixed
- *Order of rows* is fixed (i.e., tables with different order of rows are different)
- *Duplicate rows* matter

## In mathematical relations:

- Order of tuples and duplicate tuples do not matter
- Order of attributes is still fixed

## Question:

*Can we model relations so that we get rid of attribute order?*

## Reminder: Relations as Subsets of Cartesian Products

- Tuple as elements of  $\text{String} \times \text{Int} \times \text{String} \times \text{String}$   
E.g.,  $t = (\text{gizmo}, 19.99, \text{gadgets}, \text{GizmoWorks})$
- Relation = subset of  $\text{String} \times \text{Int} \times \text{String} \times \text{String}$
- Order in the tuple is important !
  - $(\text{gizmo}, 19.99, \text{gadgets}, \text{GizmoWorks})$
  - $(\text{gizmo}, 19.99, \text{GizmoWorks}, \text{gadgets})$
- No explicit attributes, hidden behind positions

# Alternative Definition: Relations as Sets of Functions

- Fix the set  $A$  of attributes, e.g.

$$A = \{\text{Name, Price, Category, Manufacturer}\}$$

- Fix  $D$  as the union of the attribute domains, e.g.,

$$D = \text{dom}(\text{Name}) \cup \text{dom}(\text{Price}) \cup \text{dom}(\text{Category}) \\ \cup \text{dom}(\text{Manufacturer})$$

- A tuple is a function  $t: A \rightarrow D$

- E.g.

{	Prodname	→	gizmo,
	Price	→	19.99,
	Category	→	gadgets,
	Manufacturer	→	GizmoWorks}

*This is the model  
underlying SQL*

- Order in a tuple is not important,  
attribute names are important!

# Notation

Schema  $R(A_1, \dots, A_n)$ , tuple  $t$  that satisfies the schema

Then:

- $t[A_i]$  = value of  $t$  for attribute  $A_i$
- $t[A_i, A_j, A_k]$   
= subtuple of  $t$ , with values for  $A_i, A_j, A_k$

**Example:**  $t = (\text{gizmo}, 19.99, \text{gadgets}, \text{GizmoWorks})$

–  $t[\text{Price}] = 19.99$

–  $t[\text{ProdName}, \text{Manufacturer}] = (\text{gizmo}, \text{GizmoWorks})$

# Two Definitions of Relations

- **Positional tuples**, without attribute names
- **Tuples as mappings/functions** of attributes

In theory and practice, both are used, e.g.,

- SQL: tuples as functions
- QBE (query by example): positional tuples

*We will switch back and forth between the two...*



# Why Relations?

- Very simple model
- *Often* a good match for the way we think about our data
- Foundations in logic and set theory
- Abstract model that underlies SQL, the most important language in DBMSs today
  - But SQL uses “bags” while the abstract relational model is set-oriented

# 4. The Relational Data Model

## 4.3 Integrity Constraints

1. Schemas
2. Instances
- 3. Integrity Constraints**

# Integrity Constraints

**Ideal:** DB instance reflects the real world

**In real life:** This is not always the case

**Goal:** Find out, when DB is out of sync

**Observation:**

Not all mathematically possible instances make sense

**Idea:**

- Formulate conditions that hold for all plausible instances
- Check whether the condition holds after an update

*Such conditions are called integrity constraints!*

# Common Types of Integrity Constraints

- **Functional Dependencies (FDs)**
  - “Employees in the same department have the same boss”
- **Superkeys and keys** (special case of FDs)
  - “Employees with the same tax code are identical”
- **Referential Integrity** (also “foreign key constraints”)
  - “Employees can only belong to a department that is mentioned in the Department relation”
- **Domain Constraints**
  - “No employee is younger than 15 or older than 80”

Integrity constraints (ICs) are part of the schema  
We allow only instances that satisfy the ICs

# Functional Dependencies (Example)

Emp (Name, taxCode, Dept, DeptHead)

A state of Emp that contains two tuples with

- the same Dept, but different DeptHead
- the same taxCode, but different Name, Dept, or DeptHead

is definitely out of sync.

We write the desired conditions symbolically as

- Dept  $\rightarrow$  DeptHead
- taxCode  $\rightarrow$  Name, Dept, DeptHead.

We read:

- “Dept functionally determines DeptHead”, or
- “Name, Dept, and DeptHead functionally depend on taxCode”

# Functional Dependencies

DB relation  $R$ .

A **functional dependency** on  $R$  is an expression

$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n$$

where  $A_1, \dots, A_m$  and  $B_1, \dots, B_n$  are attributes of  $R$ .

An instance  $r$  of  $R$  satisfies the FD if for all tuples  $t_1, t_2$  in  $R$

$$\begin{aligned} t_1[A_1, \dots, A_m] = t_2[A_1, \dots, A_m] \\ \text{implies} \\ t_1[B_1, \dots, B_n] = t_2[B_1, \dots, B_n] \end{aligned}$$

*How many FDs are there on a given relation?*

# FDs: Example

Emp (EmpID, Name, Phone, Position)

with instance

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	Jones	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Brown	1234	Lawyer

Which FDs does this instance satisfy?

- EmpID → Name, Phone, Position
- Position → Phone
- Phone → Position

# General Approach for Checking FDs

To check  $A \rightarrow B$  on an instance,

- erase all other columns

...	A	...	B	
	X1		Y1	
	X2		Y2	
	...		...	

- check if the remaining relation is functional in  $A$

*Why is that correct?*



## FDs: Example (Cntd.)

Check Position → Phone !

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	Jones	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Brown	1234	Lawyer

*Is the white relation functional in Position?*

# FDs, Superkeys and Keys

Person (SSN, Name, DOB)  
SSN  $\rightarrow$  Name, DOB

Product (Name, Price, Manufacturer)  
Name  $\rightarrow$  Price, Manufacturer  
Name  $\rightarrow$  Name, Price, Manufacturer  
Name, Price  $\rightarrow$  Name, Price, Manufacturer

Book (Author, Title, Edition, Price)  
Author, Title, Edition  $\rightarrow$  Price

- A set of attributes of a relation is a **superkey** if it functionally determines all the attributes of the relation
- A superkey is a **candidate key** if none of its subsets is a superkey

*Candidate keys are minimal superkeys*

# Keys: Definitions

- **Superkey**
  - a *set* of attributes whose values together *uniquely* identify a tuple in a relation
- **Candidate Key**
  - a superkey for which no proper subset is a superkey:  
a superkey that is *minimal*
  - *Can be more than one for a relation*
- **Primary Key**
  - a candidate key chosen to be the main key
  - *One for each relation,*  
indicated by underlining the key attributes

Student(studno,name,tutor,year)

# Example: Multiple Keys

Student (Lastname, Firstname,

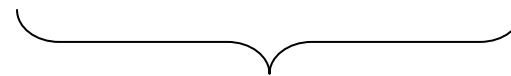
MatriculationNo, Major)



Candidate key  
(2 attributes)



Candidate key



Superkey

Note: There are alternate candidate keys

- Candidate keys are  
    {Lastname, Firstname} and  
    {StudentID}

# Foreign Keys

A **set of attributes** in a relation that exactly matches the **primary key** in another relation

- the names of the attributes don't have to be the same but must be of the same domain

Student (studno, name, hons, **tutor**, **tutorroom**, year)

Staff (lecturer, roomno, **appraiser**, **approom**)



## Notation:

FK1: Student (**tutor**, **tutorroom**) references Staff (**lecturer**, **roomno**)

FK2: Staff (**appraiser**, **approom**) references Staff (**lecturer**, **roomno**)

# Satisfaction of Foreign Key Constraints

“FK:  $R(A)$  references  $S(B)$ ”

is **satisfied** by an instance of  $R$  and  $S$  if  
for every  $t1$  in  $R$  there is a  $t2$  in  $S$  such that  
 $t1[A] = t2[B]$ ,  
provided  $t1[A]$  is not null

*To keep things simple, we assume that “lecturer” alone is the primary key of Staff*

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	null

*Foreign key constraints are also called “referential integrity constraints.”*

# Updates May Violate Constraints ...

Updates are

**Insertions, Deletions, Modifications**

of tuples

Example DB with tables as before:

Student (studno, name, hons, tutor, year)

Staff (lecturer, roomno, appraiser)

The DB has key and foreign key constraints

## **Questions:**

- *What can go wrong?*
- *How should the DBMS react?*

# Insertions (1)

If the following tuple is inserted into **Student**, what should happen? Why?

(s1, jones, cis, capon, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>



## Insertions (2)

If the following tuple is inserted into **Student**, what should happen? Why?

(null, jones, cis, capon, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

## Insertions (3)

If the following tuple is inserted into **Student**,  
what should happen? Why?

(s7, jones, cis, null, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

## Insertions (4)

If the following tuple is inserted into **Student**, what should happen? Why?

(s7, jones, cis, calvanese, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

# Deletions (1)

If the following tuple is deleted from **Student**,  
is there a problem? And what should happen?

(s2, brown, cis, kahn, 2)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

## Deletions (2)

And if this one is deleted from **Staff** ?

(kahn, IT206, watson)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

# Modifications (1)

What if we change in Student

(s1, jones, ca, bush, 2)

to

(s1, jones, ca, watson, 2) ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

## Modifications (2)

And what if we change in Student

(s2, brown, cis, kahn, 2)

to

(s1, jones, ca, bloggs, 2) ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

## Modifications (3)

And what if we change in **Staff**

(lindsey, 2.10, woods)

to

(lindsay, 2.10, woods) ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>



## Modifications (4)

Now, let's change in Staff

(goble, 2.82, capon)

to

(gobel, 2.82, capon) ...

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	<i>null</i>

# Summary: Reactions to Integrity Violations

If an update **violates an IC**, the DBMS can

- **Reject** the update
- **Repair** the violation by
  - *inserting null*
  - *inserting a default value*
  - *cascading a deletion*
  - *cascading a modification*

# Summary

- The relational model is based on concepts from set theory (and logic)
- It formalises the concept of a *table*
- Distinguish:
  - **relation schema**: relation name, attributes, domains/types
  - **relation instance**: relation over domains of attributes
- Two formalisations of **tuples**
  - **positional tuples** vs. tuples as **functions on attributes**
- **Integrity constraints**: Domain cs, FDs, Keys, FKs
- **Updates** may violate ICs
  - and the DMBS has to react