Lic. Informatique, Univ Lille 1, 2010-11

- **Introduction aux bases de données relationnelles**
  - 2ème séance: **Introduction au modèle relationnel. Algèbre relationnelle.**
- **Enseignante**: C. Kuttler
- **Biblio**: chapitre 2 de *Database Systems Concepts* de Silberschatz et al, McGraw-Hill (6ème edition, 2010)
- Ces transparents sont une adaptation de ceux disponibles sur le site du livre: **www.db-book.com**

Chapter 2: Intro to Relational Model

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use

Example of a relation: instructors

The relation's **attributes** (or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples (or rows)

A row represents a **relationship** among a set of values.

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; i.e. indivisible
- The special value ***null*** is a member of every domain
- The null value causes complications in the definition of many operations

©Silberschatz, Korth and Sudarshan

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are *attributes*

- $R = (A_1, A_2, \ldots, A_n)$ is a *relation* **schema**

  Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)

  - Each A_i can take values from a set D_i, called its domain.

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation** ***r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

- The current values (**relation instance**) of a relation are specified by a table

- An element ***t*** of the relation ***r*** is a **tuple**, represented by a *row* in a table

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Database design

- A database consists of **multiple relations**
- Information about an enterprise is broken up into parts

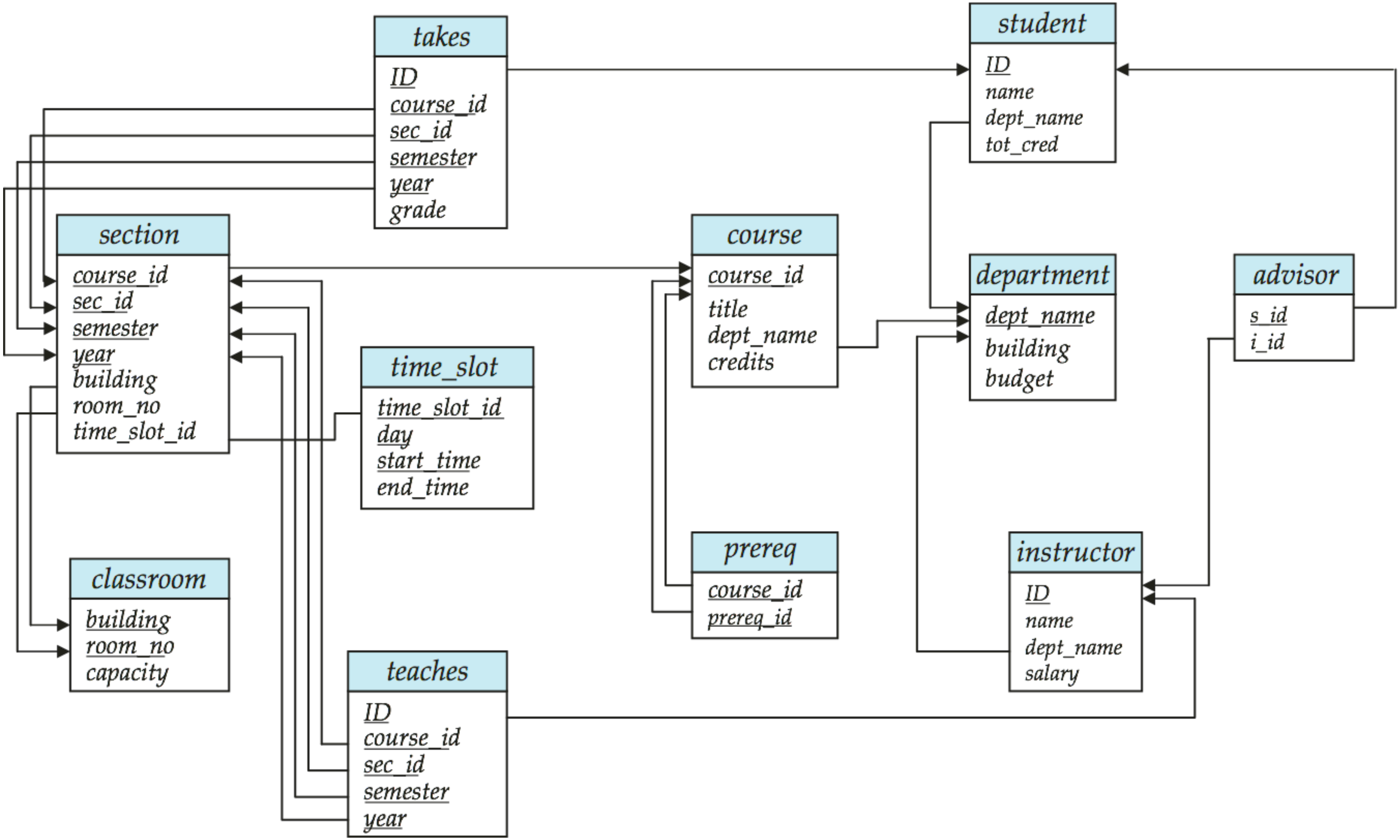
    instructor
    student
    advisor

- **Bad design**: storing all information as a single relation
    univ (*instructor_ID, name, dept_name, salary, student_ID*, ..)

    this results in

    - **repetition** of information (e.g., two students have the same instructor)
    - the need for **null** values (e.g., represent an student with no advisor)
- Normalization theory (Chapter 7) deals with how to design "good" relational schemas

Schema Diagram for University Database

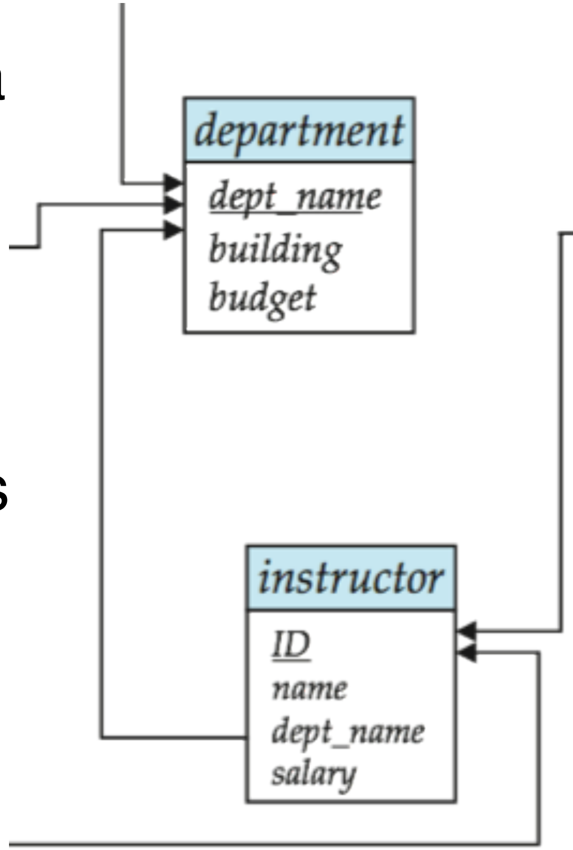


Keys

- Let K ⊆ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is **minimal**
  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - its value should never, or rarely, change!

Foreign keys

- A relation schema may have an **attribute that corresponds to the primary key of another relation**. The attribute is called a foreign key.
  - *Arrow from the foreign key attribute of the referencing relation, to the primary key of the referenced relation.*
  - E.g. *dept_name* attribute of instructor is a foreign key, referencing to *department.*
- Foreign key constraint: value in one relation must appear in another **referenced** relation



Question 1

- Consider the foreign key constraint from the *dept_name* attribute of *instructor*, to the *department* relation. Give examples of inserts and deletes to these relations, which can cause violation of the foreign key constraint.

©Silberschatz, Korth and Sudarshan

Question 2

- Consider the *time_slot* relation. Given that a particular time slot can meet more than once per week, explain why *day* and *start_time* are part of the primary key of this relation, while *end_time* is not.

Select Operation

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each **term** is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq. <. \leq$

- Example of selection:

$$\sigma_{branch\_name=\text{“}Perryridge\text{”}}(account)$$

Relational Query Languages

- Procedural vs.non-procedural, or declarative
- "Pure" languages:
  1. Relational algebra (procedural)
  - Tuple relational calculus (declarative, based on predicate logic)
  - Domain relational calculus (declarative)
- **Today**, we present the most important relational algebra operators
- **Later**, we will see how different translations of the same SQL query into relation algebra, is used in **optimization** of query execution.

Selection of tuples

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- Select tuples with A=B and D > 5
  - $\sigma_{A=B \text{ and } D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

Each **term** is one of:

<attribute> *op* (<attribute> or <constant>)

where *op* is one of: $=, \neq, >, \geq. <. \leq$

Selection of Columns

- Relation *r* :

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select columns A and C
  - Projection of r on A and C
  - $\Pi_{A,\ C}$ (r)
  - duplicates are eliminated, the result is a *set*!

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Joining two relations – Cartesian Product

- Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:
- each tuple of r is combined with each tuple of s,
- tuples are concatenated.

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Union of two relations

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- **r ∪ s**

A	B
α	1
α	2
β	1
β	3

- **Condition**: both input relations must be **compatible**: same arity, and same domains of attributes

Set difference of two relations

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r - s$

A	B
α	1
β	1

Set Intersection of two relations

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Joining two relations – Natural Join

- Let r and s be relations on schemas R and S respectively.
  Then, the "natural join" of relations R and S is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r

Natural Join Example

- Relations r and s

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

- r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Relational Algebra Operators: Summary

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}$ (*instructor*)
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}$ (*instructor*)
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$ (Natural Join)	*instructor* $\bowtie$ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian Product)	*instructor* $\times$ *department*
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.

Questions

- Consider the following expressions, which use the result of a relational algebra expression as the input to another operation. For each expression, explain in words what the example does!
  - Expressions given on board during lecture!

Questions:

- Consider the relational database
  - *employee(person_name,street,city)*
  - *works(person_name,company_name,salary)*
  - *company(company_name,city)*
- Give an expression in the relational algebra to find the names of all employees
  - who live in the city Miami
  - whose salary is greater than $100.000
  - who live in Miami and earn over $100.000

Review terms

- Table
- Relation
- Tuple
- Attribute
- Domain
- Atomic domain
- Null value
- Database schema
- Database instance
- Relation schema
- Relation instance
- Keys: super, candidate, primary
- Foreign key: referencing rleation, referenced relation
- Referential integrity constraint
- Query languages: procedural vs declarative

End of Chapter 2

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use