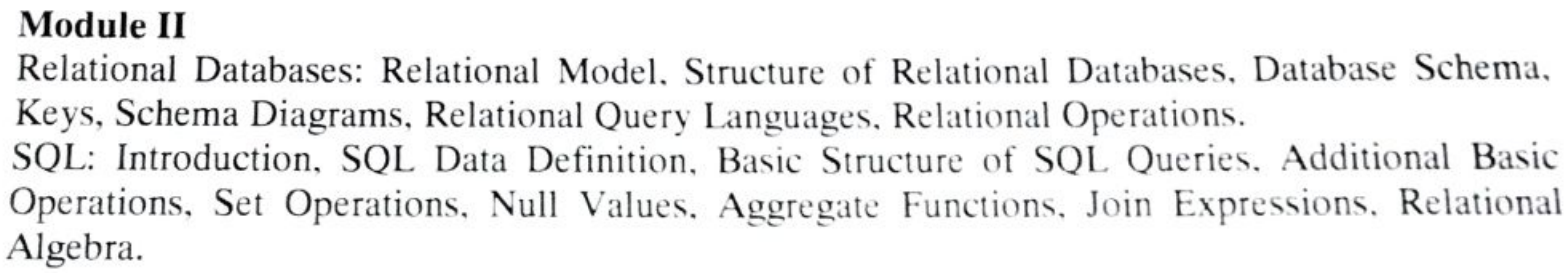


Module II

Relational Databases: Relational Model, Structure of Relational Databases, Database Schema, Keys, Schema Diagrams, Relational Query Languages, Relational Operations.

SQL: Introduction, SQL Data Definition, Basic Structure of SQL Queries, Additional Basic Operations, Set Operations, Null Values, Aggregate Functions, Join Expressions, Relational Algebra.

Database System Concepts, 6th Ed.

Introduction to the Relational Model

Database System Concepts, 6th Ed.

Relational Model

- The relational model remains the primary data model for commercial data-processing applications
- simple, which eases the job of the programmer,
- incorporating various new features and capabilities
- object-relational features such as complex data types and stored procedures, support for XML data, and

Structure of Relational Databases

- A relational database consists of a collection of tables, each of which is assigned a unique name.

- In general, a row in a table represents a relationship among a set of values.
- In mathematical terminology, a tuple is simply a sequence (or list) of values. A relationship between n values is represented mathematically by an n-tuple of values, that is, a tuple with n values, which corresponds to a row in a table.

Structure of Relational Databases

- Thus, in the relational model the term relation is used to refer to a table, while the term tuple is used to refer to a row.
- Similarly, the term attribute refers to a column of a table.

- We use the term relation instance to refer to a specific instance of a relation, that is, containing a specific set of rows.
- For each attribute of a relation, there is a set of permitted values, called the domain
- of that attribute. Thus, the domain of the salary attribute of the instructor relation is the set of all possible salary values, while the domain of the name attribute is the set of all possible instructor names

Example of a Relation

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)

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**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is “unknown”
- The null value causes complications in the definition of many operations

The null value is a special value that signifies that the value is unknown or does not

exist. For example, suppose as before that we include the attribute phone number in the

instructor relation. It may be that an instructor does not have a phone number at all,

or that the telephone number is unlisted

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*)**

- 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 ***r*** is a *tuple*, represented by a *row* in a table

Database Schema

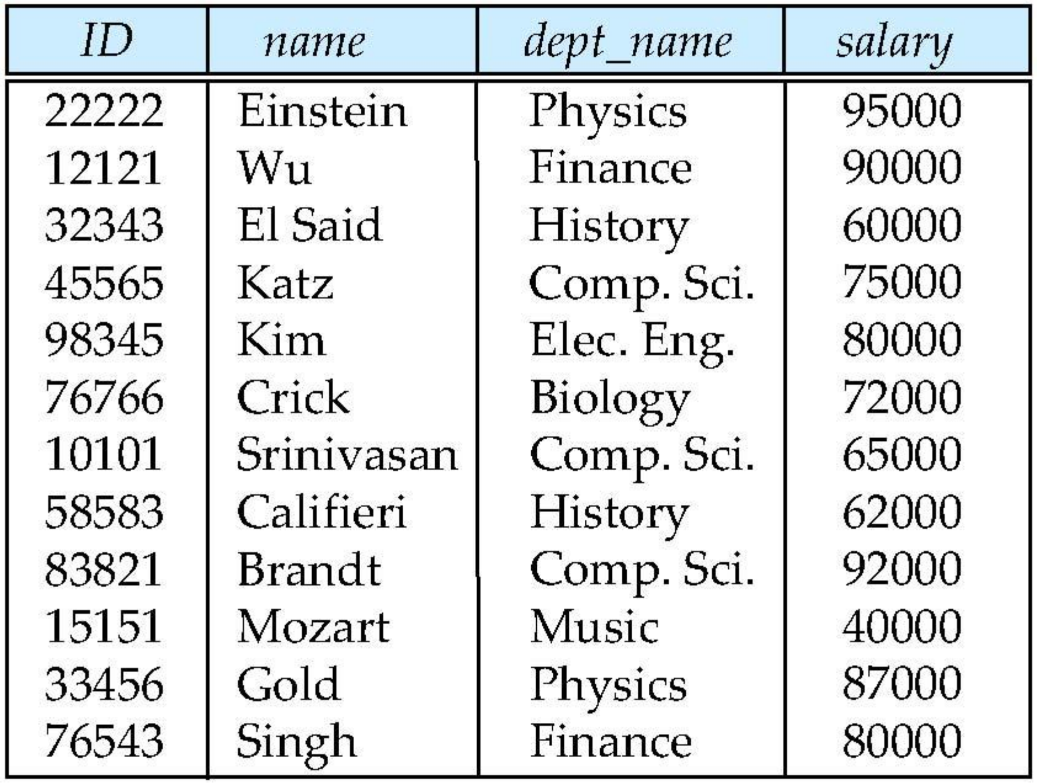
- the **database schema(normally it wont change)**,which is the logical design of the database,
- and the **database instance(it can change)**, which is a snapshot of the data in the database at a given instant in time.
- a relation schema consists of a list of attributes and their corresponding domains.
- The concept of a relation instance corresponds to the programming-language notion of a value of a variable.
- The value of a given variable may change with time; similarly the contents of a relation instance may change with time as the relation is updated.
- In contrast, the schema of a relation does not generally change.

Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, where each relation storing one part of the information
- The university database example:
  - *instructor* (*ID, name, dept_name, salary*)
  - *department (dept_name, building, budget)*
  - *student (ID, name, dept_name, tot_cred)*
  - *course (course_id, title, dept_name, credits)*
  - *prereq (course_id, prereq_id)*

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

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**.

It is possible that several distinct sets of attributes could serve as a candidate key.Suppose that a combination of name and dept name is sufficient to distinguish among members of the instructor relation. Then, both { ID } and {name, dept name} are candidate keys. Although the attributes ID and name together can distinguish instructor tuples,their combination, { ID , name}, does not form a candidate key, since the attribute ID alone is a candidate key.

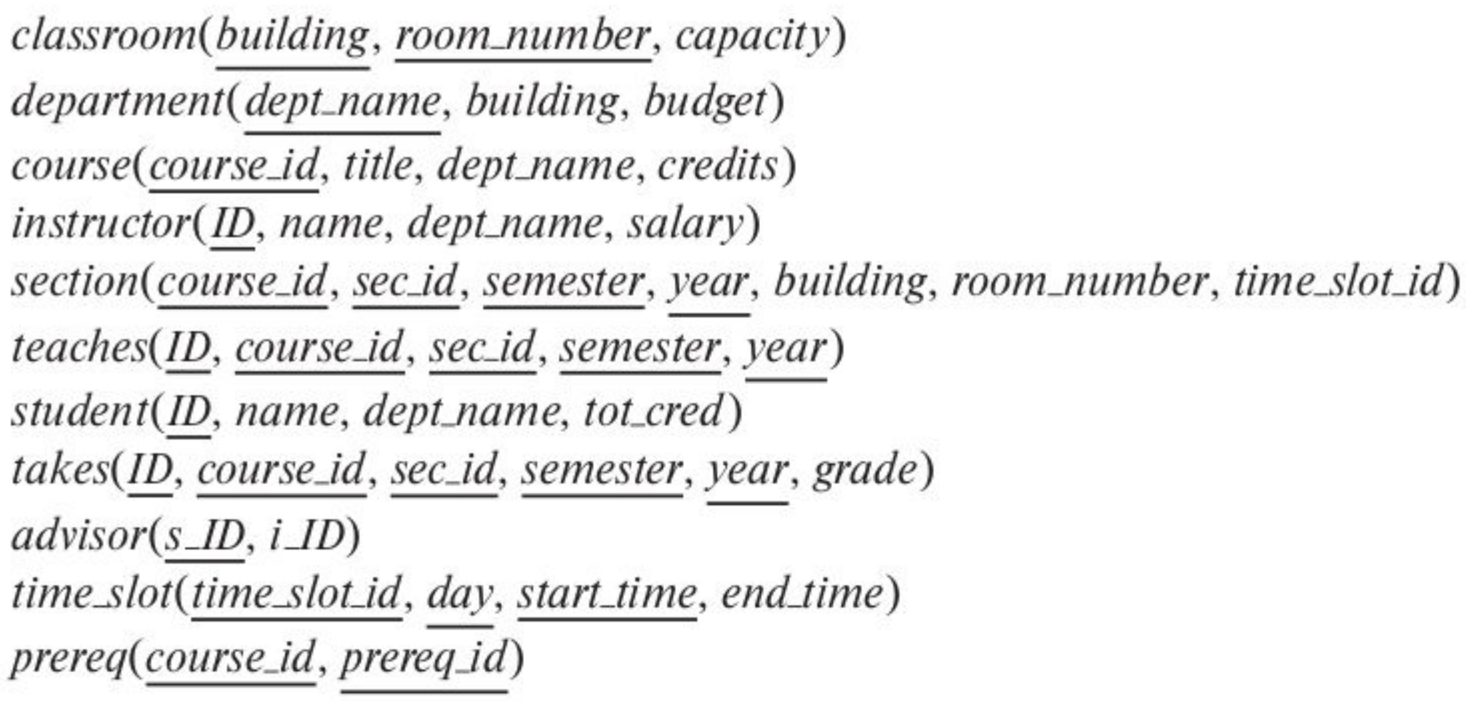
primary key constraints.

- A key (whether primary, candidate, or super) is a property of the entire relation, rather than of the individual tuples.
- Any two individual tuples in the relation are prohibited from having the same value on the key attributes at the same time.
- The designation of a key represents a constraint in the real-world enterprise being modeled.
- Thus, primary keys are also referred to as primary key constraints.
- Primary key attributes are also underlined

$$\mathit{classroom}\ (\underline{\mathit{building}}, \underline{\mathit{room\_number}}, \mathit{capacity})$$

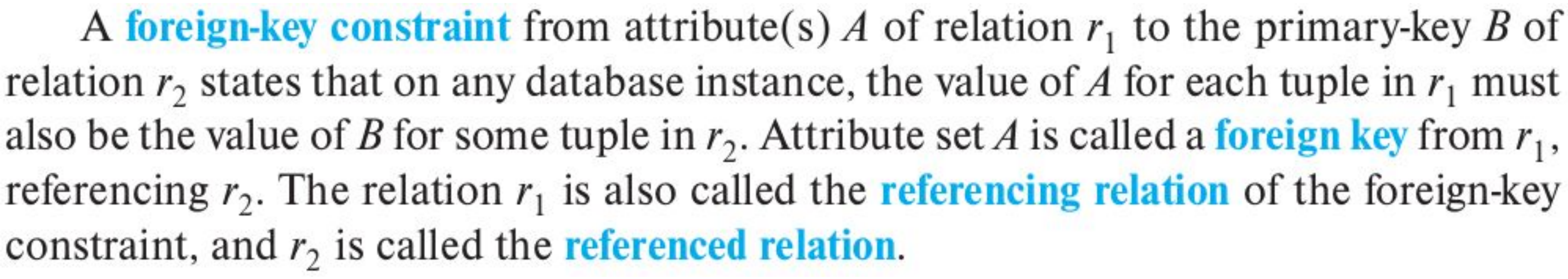
primary key constraints.

- The primary key should be chosen such that its attribute values are never, or are very rarely, changed.



classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

foreign-key constraints



A **foreign-key constraint** from attribute(s) A of relation r_1 to the primary-key B of relation r_2 states that on any database instance, the value of A for each tuple in r_1 must also be the value of B for some tuple in r_2. Attribute set A is called a **foreign key** from r_1, referencing r_2. The relation r_1 is also called the **referencing relation** of the foreign-key constraint, and r_2 is called the **referenced relation**.

foreign-key constraints

- *Note that in a foreign-key constraint, the referenced attribute(s) must be the primary key of the referenced relation.*

Referential integrity constraint

A referential integrity constraint requires that the values appearing in specified attributes of any tuple in the referencing relation also appear in specified attributes of ***at least one*** tuple in the referenced relation.

In fact, foreign-key constraints are a ***special case of referential integrity constraints***

The referenced attributes form the ***primary key*** of the referenced relation.

Schema Diagrams

A database schema, along with primary key and foreign-key constraints, can be depicted by **schema diagrams**.

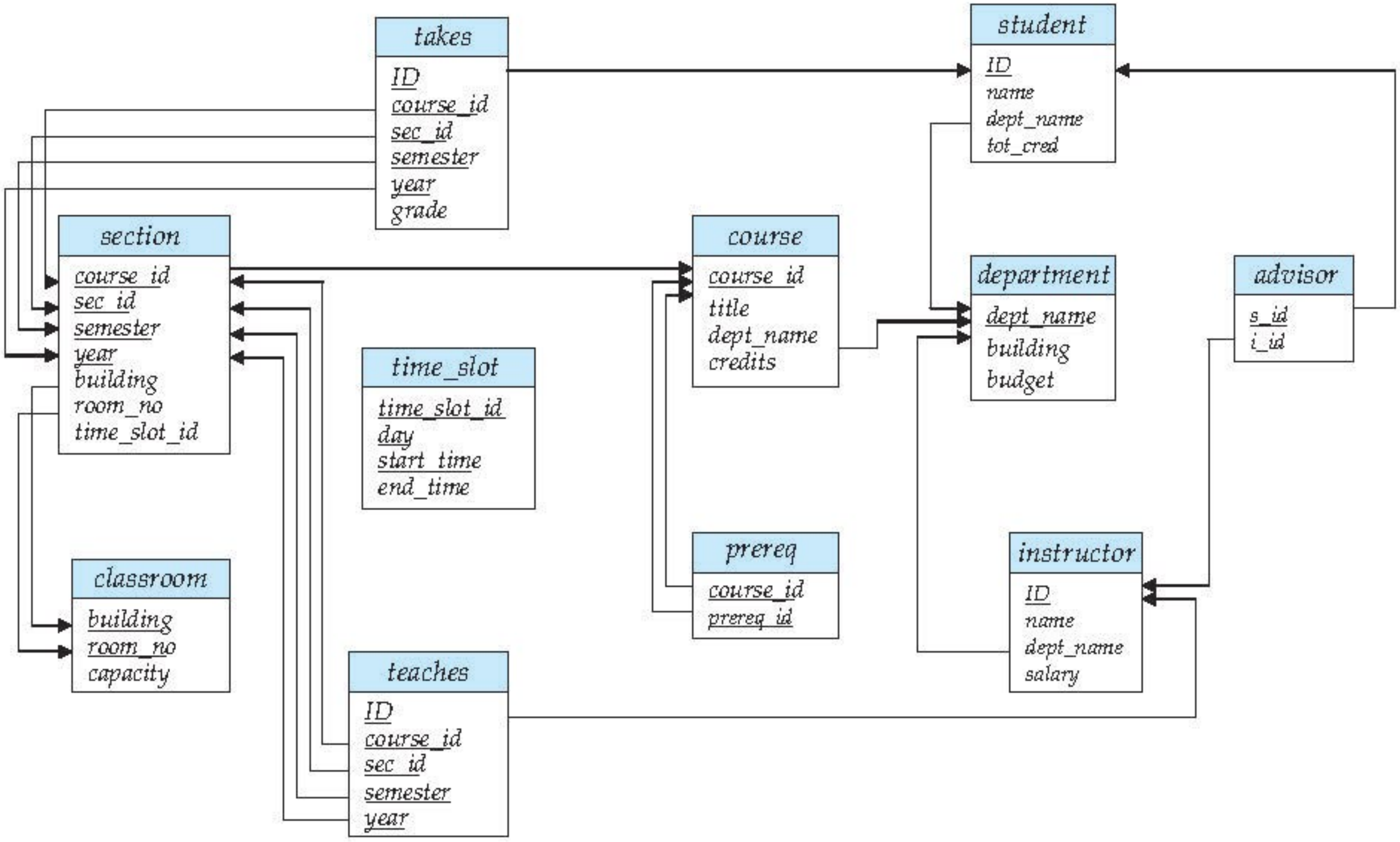
Each relation appears as a box, with the relation name at the top in blue and the attributes listed inside the box.

Primary-key attributes are shown underlined.

Foreign-key constraints appear as arrows from the foreign-key attributes of the referencing relation to the primary key of the referenced relation

Two-headed arrow to indicate a referential integrity constraint that is not a foreign-key constraints

Schema Diagram for University Database



Relational Query Languages

A query language is a language in which a user requests information from the database.

Query languages can be categorized as

1)imperative

the user instructs the system to perform a specific sequence of operations on the database to compute the desired result; such languages usually have a notion of state variables, which are updated in the course of the computation.

2) functional

the computation is expressed as the evaluation of functions that may operate on data in the database or on the results of other functions;

functions are side-effect free, and they do not update the program state

e.g relational algebra

Relational Query Languages

3)declarative query language

the user describes the desired information without giving a specific sequence of steps or function calls for obtaining that information; the desired information is typically described using some form of mathematical logic. It is the job of the database system to figure out how to obtain the desired information.

In a declarative query language, like SQL you just specify the pattern of the data you want - what conditions the results must meet, and how you want the data to be transformed (e.g., sorted, grouped, and aggregated) - **but not how to achieve that goal**. It is up to the database system's query optimizer to decide which indexes and which join methods to use, and in which order to execute various parts of the query.

e.g tuple relational calculus ,sql

The Relational Algebra

The relational algebra consists of a set of operations that take one or two relations as input and produce a new relation as their result.

1)unary operations

- operate on one relation

  e.g **select, project, and rename operations**

2)binary operations

- operate on pairs of relations e.g **union, Cartesian product, and set difference**

Select Operation

The select operation selects tuples that satisfy a given predicate. We use the lowercase Greek letter **sigma (σ)** to denote selection.

The predicate appears as a subscript to **σ.**

The argument relation is in parentheses after the σ.

Thus, to select those tuples of the instructor relation where the instructor is in the "Physics" department, we write:

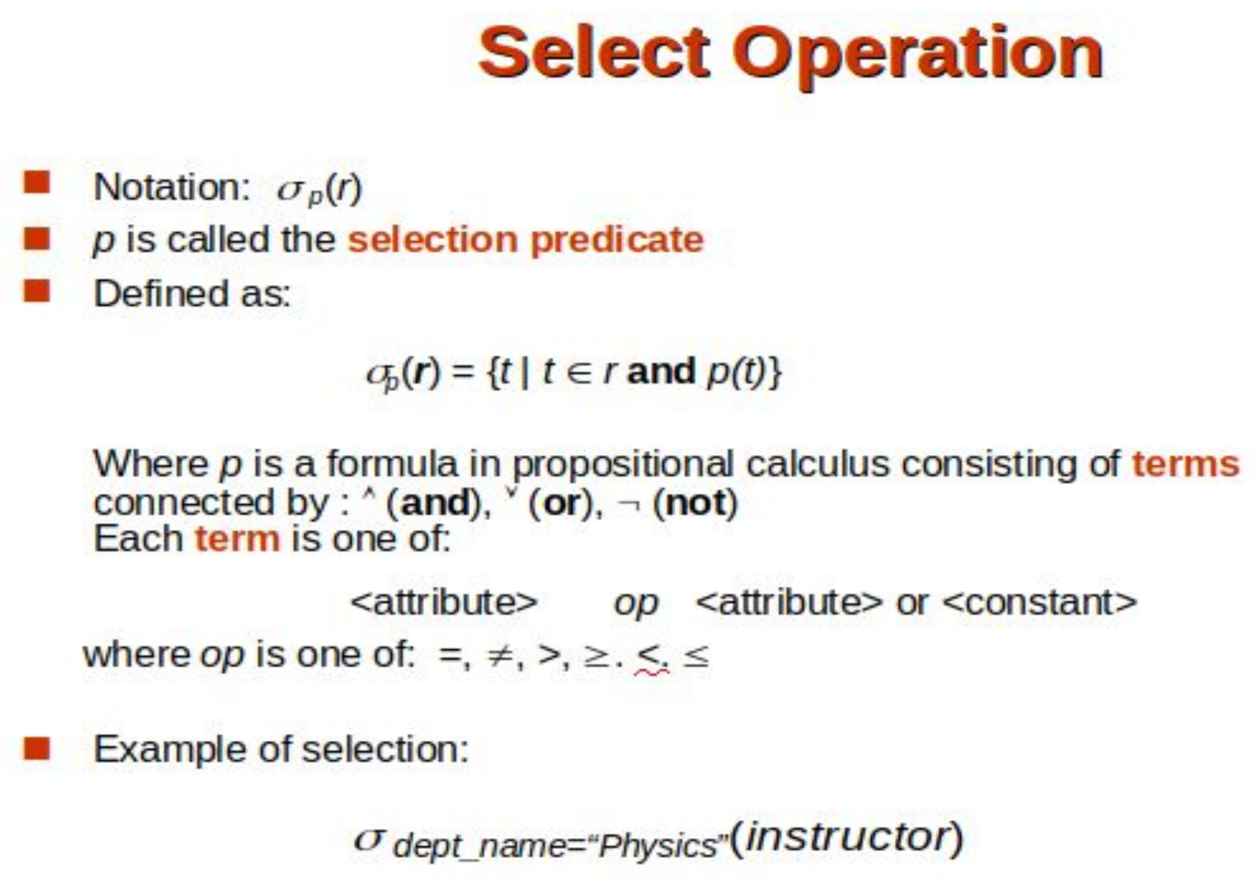
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Figure 2.10 Result of $\sigma_{dept\_name = \text{"Physics"}}(instructor)$.

$$\sigma_{dept\_name = \text{"Physics"} \wedge salary > 90000}(instructor)$$

????

Select Operation



Select Operation

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

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

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>

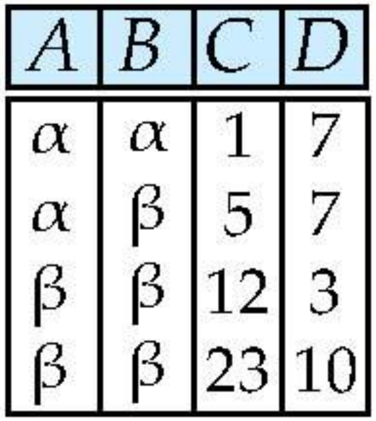
where *op* is one of: $=, \neq, >, \geq. \leq. \leq$

- Example of selection:

$$\sigma_{dept\_name=\text{“Physics”}}(instructor)$$

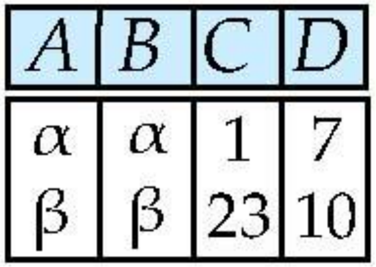
Select Operation – selection of rows (tuples)

- Relation r



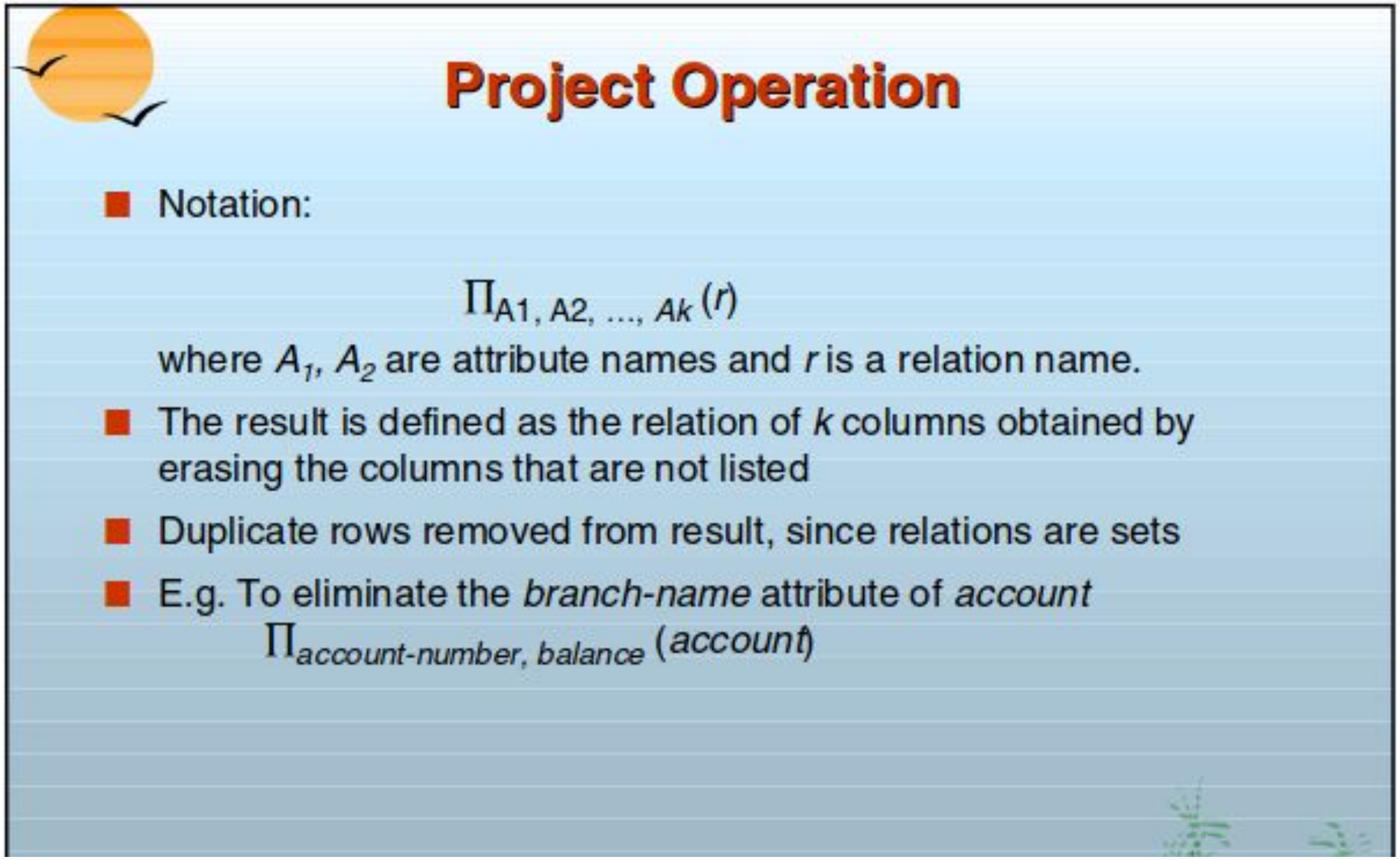
A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$



A	B	C	D
α	α	1	7
β	β	23	10

Project Operation – selection of columns (Attributes)



Project Operation

- Notation:

$$\Pi_{A1,\, A2,\, \ldots,\, Ak}\,(r)$$

  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- E.g. To eliminate the *branch-name* attribute of *account*

$$\Pi_{account\text{-}number,\ balance}\,(account)$$

Project Operation – selection of columns (Attributes)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

$\Pi_{ID,\ name,\ salary}(instructor)$

$\Pi_{ID, name, salary/12}(instructor)$

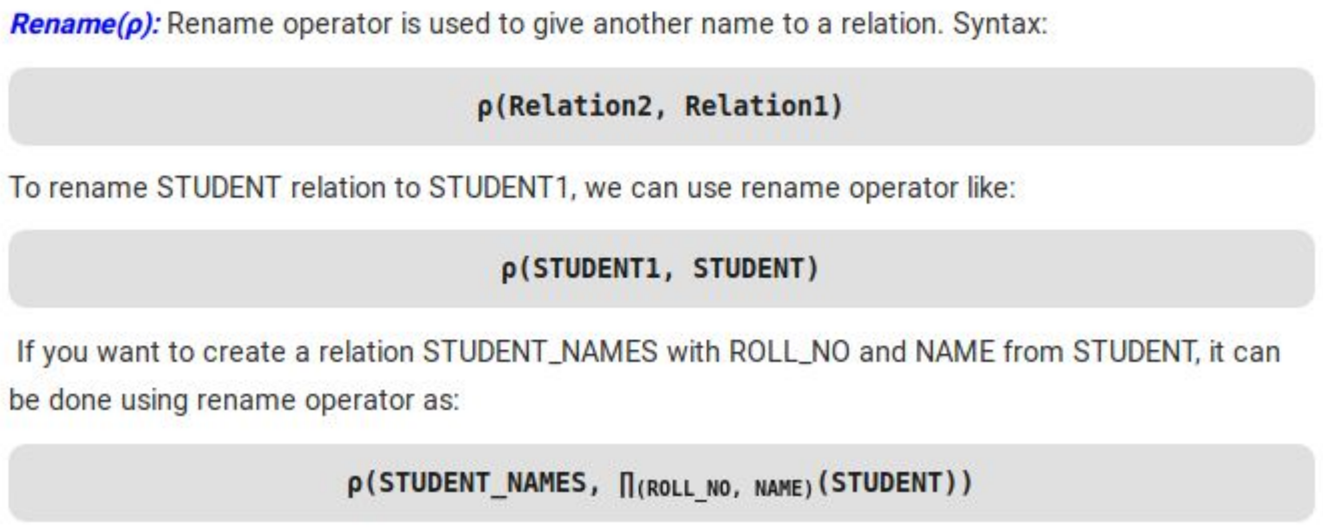
ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Composition of Relational Operations

"Find the names of all instructors in the Physics department."

$$\Pi_{name}\,(\sigma_{dept\_name = \text{“Physics”}}\,(instructor))$$

relational-algebra expression.:relational-algebra operations can be composed together



Rename(ρ): Rename operator is used to give another name to a relation. Syntax:

$$\rho(Relation2,\ Relation1)$$

To rename STUDENT relation to STUDENT1, we can use rename operator like:

$$\rho(STUDENT1,\ STUDENT)$$

If you want to create a relation STUDENT_NAMES with ROLL_NO and NAME from STUDENT, it can be done using rename operator as:

$$\rho(STUDENT\_NAMES,\ \Pi_{(ROLL\_NO,\ NAME)}(STUDENT))$$

- **The general RENAME operation ρ can be expressed by any of the following forms:**
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_{S}(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn

Union of two relations

- Relations *r, s:*

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

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*:

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

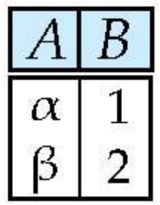
- $r \cap s$

A	B
α	2

Note: $r \cap s = r - (r - s)$

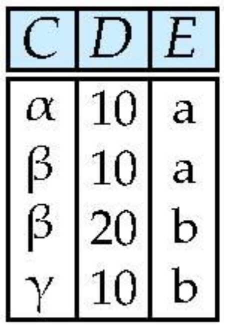
joining two relations -- Cartesian-product

- Relations *r, s*:



A	B
α	1
β	2

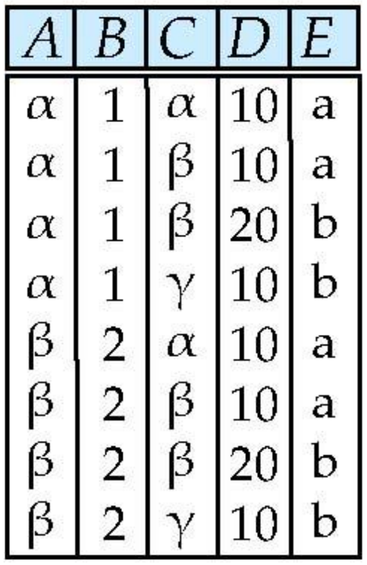
r



C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:



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

Cartesian-product – naming issue

- Relations *r, s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*r.B*	*s.B*	*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

Renaming a Table

- Allows us to refer to a relation, (say E) by more than one name.

  $\rho_{x}(E)$

  returns the expression *E* under the name *X*

- Relations *r*

A	*B*
α	1
β	2

r

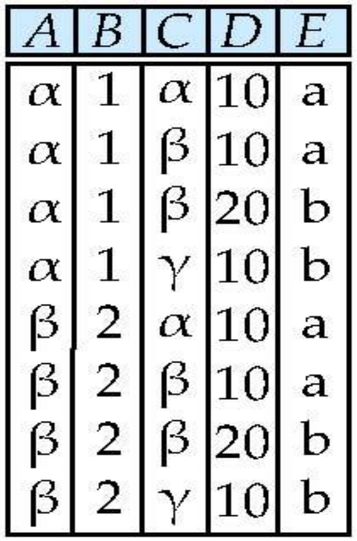
- $r \times \rho_{s}(r)$

r.A	*r.B*	*s.A*	*s.B*
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Composition of Operations

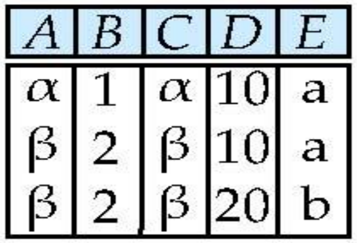
- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

- $r \times s$



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

- $\sigma_{A=C}(r \times s)$



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

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*
    - *t* has the same value as t_s on *s*

Natural Join Example

- Relations r, 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

- Natural Join
  - r $\bowtie$ 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

$$\Pi_{A,\ r.B,\ C,\ r.D,\ E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s)))$$

Notes about Relational Languages

- Each Query input is a table (or set of tables)
- Each query output is a table.
- All data in the output table appears in one of the input tables
- Relational Algebra is not Turning complete
- Can we compute:
  - SUM
  - AVG
  - MAX
  - MIN

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary} >= 85000}(\textit{instructor})$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{\textit{ID, salary}}(\textit{instructor})$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (Union)	$\Pi_{\textit{name}}(\textit{instructor}) \cup \Pi_{\textit{name}}(\textit{student})$
	Output the union of tuples from the *two* input relations.
- (Set Difference)	$\Pi_{\textit{name}}(\textit{instructor}) -- \Pi_{\textit{name}}(\textit{student})$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.