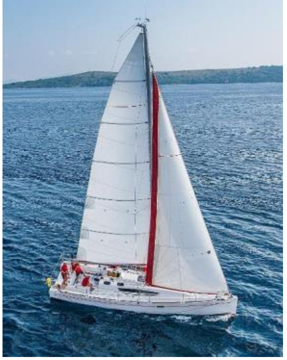
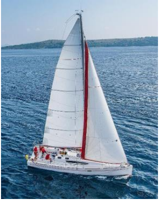


Part 2:

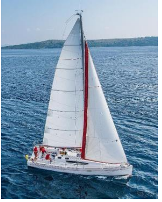
Introduction to the Relational Model

Database System Concepts, 7th Ed.



Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages
- The Relational Algebra



Example of a *Instructor* 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)

Mathematical Definition of Relation

- Consider two arbitrary sets A and B. The set of all ordered pairs *(a, b)* where $a \in A$ and $b \in B$ is called the **Cartesian product**, of A and B

$$A \times B = \{(a, b) : a \in A \text{ and } b \in B\}$$

  If $A = B$, then we often write $A \times A$ as A^2.

  - E.g., if $A = \mathbb{R}$ = the set of real numbers, then $\mathbb{R}^2$ is the two-dimensional Cartesian plane

- Denoting the cardinality of A by $|A|$, then

$$|A \times B| = |A| \times |B|$$

- A *binary relation* R from A to B is a subset of $A \times B$, i.e. $R \subseteq A \times B$
  - E.g., Let $A = B$ = set of all countries, and R indicates that two countries are adjacent; thus

    (Italy, Switzerland) $\in R$,

    (Canada, Mexico) $\notin R$

Mathematical Definition of Relation

- In general, an *n*-ary *relation R* on sets $D_1, D_2, \ldots, D_n$ is a subset of $D_1 \times D_2 \times \ldots \times D_n$; i.e.

$$R \subseteq \prod_{k=1}^{n} D_k$$

and $D_1, D_2, \ldots, D_n$ are often called domains, and

$$\left|\prod_{k=1}^{n} D_k\right| = \prod_{k=1}^{n} |D_k|$$

  - For example

    let D_1= student# , D_2= Department, D_3= City of origin. Then

    (10234, Physics, Hong Kong) $\in R$

    means that student 10234 belongs to the Physics Department and comes from Hong Kong, but

    (10234, Music, Hong Kong) $\notin R$

    means that it is not true that student 10234 belongs to the Music Department and comes from Hong Kong

- The presence of a particular tuple in *R* means that the information of that tuple represents valid information at the time
  - We may list all elements of *R* in a 3-column table

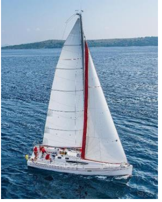
Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are **attributes**
- $R(A_1, A_2, \ldots, A_n)$ is a **relation schema**, sometimes also called **relation scheme**, it consists of a relation name and a list of attributes, which correspond to columns in a table

  Example:

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

- A **relation instance** (also called **relation state**) *r* defined over schema *R* is denoted by *r*(*R*), which consists of actual data values
  - The current values a relation are specified by a table
- An element ***t*** of relation ***r*** is called a **tuple** and is represented by a row in a table
- A schema is the logical structure, and an instance is a snapshot of the data at a given instant of time



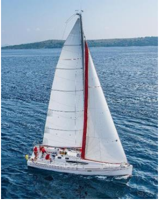
Attributes

- The set of allowable 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, indicating that the value is "unknown"
- The null value causes complications in the definition of many operations

Relations are Unordered

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

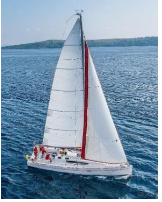
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music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



Database Schema

- Database schema -- is the logical structure of the database.
- Database instance -- is a snapshot of the data in the database at a given instant in time.
- Example:
  - schema: *instructor* (*ID, name, dept_name, salary*)
  - Instance:

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 $K \subseteq R$
- *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**
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example: *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Relational Algebra

- Relational algebra is a procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result
- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ

Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- *p* is called the **selection predicate**
- Example: select those tuples of the *instructor* relation where the instructor is in the “Physics” department.
  - Query

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

  - Result

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

Select Operation

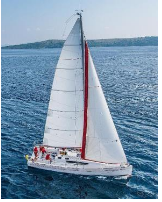
- We allow comparisons using

  $=, \neq, >, \geq. <. \leq$

  in the selection predicate
- We can combine several predicates into a larger predicate by using the connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
- Example: Find the instructors in Physics with a salary greater $90,000, we write:

  $\sigma_{\textit{dept\_name}=\text{“Physics”} \wedge \textit{salary} > 90{,}000}\ (\textit{instructor})$
- The select predicate may include comparisons between two attributes.
  - Example, find all departments whose name is the same as their building name:
  - $\sigma_{\textit{dept\_name}=\textit{building}}\ (\textit{department})$

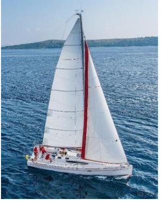


Project Operation

- A unary operation that returns its argument relation, with certain attributes left out
- Notation:

$$\Pi_{A_1, A_2, A_3 \ldots A_k}(r)$$

  where $A_1, A_2, \ldots, A_k$ 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



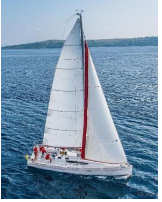
Project Operation Example

- Example: eliminate the *dept_name* attribute of *instructor*
- Query*:*

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

- Result:

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

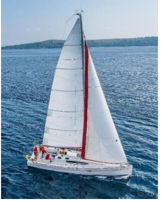


Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**
- Consider the query -- Find the names of all instructors in the Physics department

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

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation

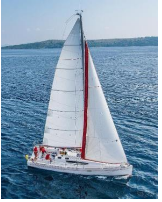


Cartesian Product Operation

- The Cartesian product operation (denoted by X) allows us to combine information from any two relations
- Example: the Cartesian product of the relations *instructor* and *teaches* is written as:

  instructor X *teaches*

- We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide)
- Since the instructor *ID* appears in both relations, we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came
  - *instructor.ID*
  - *teaches.ID*



The *instructor* x *teaches* table

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Join Operation

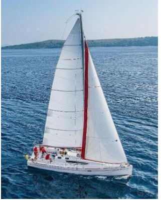
- The join operation may be derived from the Cartesian-Product, followed by selection, and sometimes projection
- The Cartesian-Product

  instructor X *teaches*

  associates every tuple of instructor with every tuple of teaches.
  - Most of the resulting rows have information about instructors who did NOT teach a particular course.
- To get only those tuples of "*instructor* X *teaches* " that pertain to instructors and the courses that they taught, we write:

  $\sigma_{instructor.id = teaches.id}$ (*instructor* x *teaches*)

  - We get only those tuples of "*instructor* X *teaches*" that pertain to instructors and the courses that they taught



Join Operation

- The table corresponding to:

$$\sigma_{instructor.id = teaches.id}(instructor \times teaches)$$

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

Join Operation

- The **join** operation allows us to combine a select operation and a Cartesian-Product operation into a single operation
- Consider relations *r*(*R*) and *s*(*S*)
- Let "theta" be a predicate on attributes in the schema *R* "union" *S*. The join operation $r \bowtie_\theta s$ is defined as follows:

$$r \bowtie_\theta s = \sigma_\theta (r \times s)$$

- Thus

$$\sigma_{instructor.id = teaches.id} (instructor \times teaches)$$

- Can equivalently be written as

$$instructor \bowtie_{Instructor.id = teaches.id} teaches$$

Union Operation

- The union operation allows us to combine two relations
- Notation: $r \cup s$
- For $r \cup s$ to be valid

  1. *r, s* must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \land year=2017} (section)) \cup$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \land year=2018} (section))$$

Union Operation

- Result of:

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \wedge year=2017} (section)) \cup$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \wedge year=2018} (section))$$

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

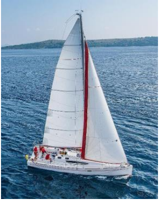
Set-Intersection Operation

- The set-intersection operation allows us to find tuples that are in both the input relations
- Notation: $r \cap s$
- Assume:
  - r, s have the same arity
  - attributes of r and s are compatible
- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \ \wedge \ year=2017} (section)) \cap$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \ \wedge \ year=2018} (section))$$

  - Result

course_id
CS-101

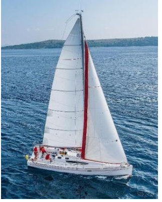


Set Difference Operation

- The set-difference operation allows us to find tuples that are in one relation but are not in another
- Notation $r - s$
- Set differences must be taken between **compatible** relations
  - r and s must have the same arity
  - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$$\Pi_{course\_id}\ (\sigma_{semester=\text{"Fall"}\ \wedge\ year=2017}\ (section))\ -$$
$$\Pi_{course\_id}\ (\sigma_{semester=\text{"Spring"}\ \wedge\ year=2018}\ (section))$$

course_id
CS-347
PHY-101



The Assignment Operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables
- The assignment operation is denoted by $\leftarrow$ and works like assignment in a programming language
- Example: Find all instructors in the "Physics" and "Music" departments

$$Physics \leftarrow \sigma_{dept\_name=\text{"Physics"}}(instructor)$$

$$Music \leftarrow \sigma_{dept\_name=\text{"Music"}}(instructor)$$

$$Physics \cup Music$$

- With the assignment operation, a query can be written as sequential operations, consisting of a series of assignments followed by an expression whose value is displayed as the result of the query

The Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them. The rename operator, ρ, is provided for that purpose
- The expression:

$$\rho_x (E)$$

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

Equivalent Queries

- There is in general more than one way to write a query in relational algebra
- Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000
- Query 1

$$\sigma_{\textit{dept\_name}=\text{"Physics"} \wedge \textit{salary} > 90{,}000}\,(\textit{instructor})$$

- Query 2

$$\sigma_{\textit{dept\_name}=\text{"Physics"}}(\sigma_{\textit{salary} > 90.000}\,(\textit{instructor}))$$

- The two queries are not identical; they are, however, equivalent -- they give the same results

Equivalent Queries

- Example: Find information about courses taught by instructors in the Physics department
- Query 1

  $\sigma_{dept\_name=\text{``Physics''}}(instructor \bowtie_{instructor.ID\, =\, teaches.ID} teaches)$

- Query 2

  $(\sigma_{dept\_name=\text{``Physics''}}(instructor)) \bowtie_{instructor.ID\, =\, teaches.ID} teaches$