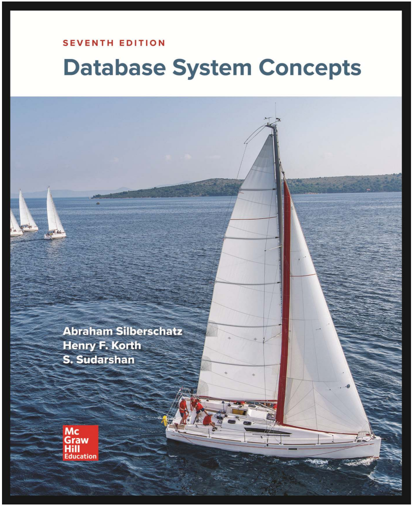
Database System Concepts, 7^{th} Edition
Chapter 2: Introduction to Relational Model

Silberschatz, Korth and Sudarshan

July 27, 2025

Database System Concepts



Content has been extracted from *Database System Concepts*, Seventh Edition, by Silberschatz, Korth and Sudarshan. Mc Graw Hill Education. 2019.
Visit `https://db-book.com/`.

Example of a Instructor Relation

attributes (or columns)

tuples (or rows)

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

Attributes

- 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. It indicates that the value is “unknown”.
- the `null` value causes complications in the definition of many operations.

Relation 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

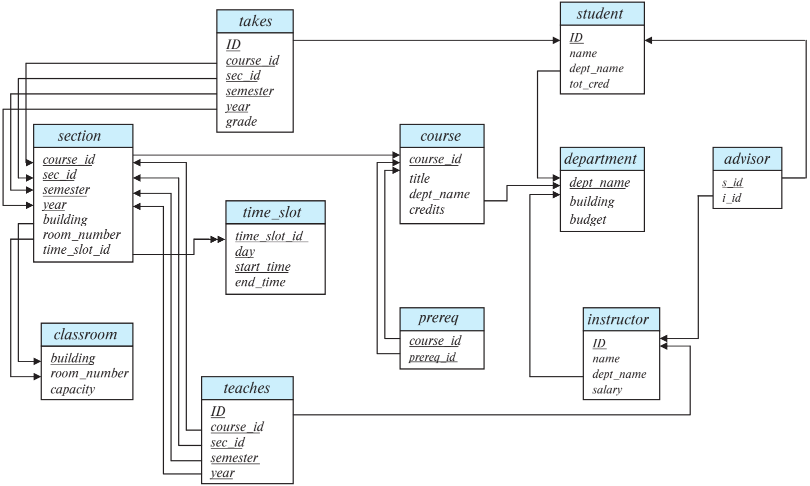
Keys

- Let $K \subseteq R$.
- K is a **superkey** of R if values of 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 candidate key for *instructor*.
- One of the candidate keys is selected to be the **primary key**.
  - ¿which one?
- **Foreign key constraint:** Value in one relation must appear in another.
  - **Referencing** relation.
  - **Referenced** relation.
  - Example: *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*.

Schema of the University Database

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

Schema Diagram for University Database



Relational Query Languages

- "Pure" languages:
  - Relational algebra.
  - Tuple relational calculus.
  - Domain relational calculus.
- The above 3 pure languages are equivalent in computing power.
- We will concentrate in this chapter on relational algebra:
  - Not turing-machine equivalent.
  - Consist of 6 basic operations.

Relational Algebra

- A functional 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: Π
  - union: $\cup$
  - difference: $-$
  - cartesian product: $\times$
  - 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:
  - Statement: "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 (Cont.)

- We allow comparisons using $=$, $\neq$, $>$, $\geq$, $<$, $\leq$ in the selection predicate.
- We can combine several predicates into a larger predicate by using connectives: $\wedge$ **and**, $\vee$ **or**, $\neg$ **not**.
- Example: "Find the instructors in Physics with a salary greater than \$90000."
- Solution:

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

- Then select predicate may include comparisons between two attributes.
  - Example: "Find all departments whose name is the same as their building name."
  - Solution:

$$\sigma_{\text{dept\_name} = \text{building}}(department)$$

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 (Cont.)

- Example: "Eliminate the dept_name attribute of instructor."
- Query:

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

- 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 a relation itself and therefore 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."

$$\Pi_{\text{name}}(\sigma_{\text{dept\_name} = \text{`Physics'}}(instructor))$$

- 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 $\times$) allows us to combine information from any two relations.
- Example: the cartesian product of the relations *instructor* and *teaches* is written as:

$$instructor \times teaches$$

- 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 × 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 Cartesian-Product

$$instructor \times teaches$$

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 "$instructor \times teaches$" that pertain to instructors and the courses that they taught, we write:

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

- The result of this expression is shown in the next slide.

Join Operation (Cont.)

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 (Cont.)

- 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

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

- Can equivalent be written as:

$$instructor \bowtie_{\text{instructor.ID = teaches.ID}} teaches$$

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 attributes domain must be **compatible** (i.e. 2^{nd} column of r deals with the same type of values as does the 2^{nd} column of s).
- Example: "Find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both".
- Query:

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

Union Operation (Cont.)

- Result of:

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

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

Intersection Operation

- The intersection operation allows us to find tuples that are in both the input relations.
- Notation: $r \cap s$.
- For $r \cap s$ to be valid:
  1. r,s must have the same *arity*.
  2. The attributes domain must be *compatible*.
- Example: "Find the set of all courses taught in the Fall 2017 and the Spring 2018 semesters".
- Query:

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

course_id
CS-101

Difference Operation

- The difference operation allows us to find tuples that are in one relation but are not in another.
- Notation: $r - s$.
- For $r - s$ to be valid:
  1. r,s must have the same *arity*.
  2. The attributes domain must be *compatible*.
- Example: "Find the set of all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester".
- Query:

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

course_id
CS-347
PHY-101

The Assignment Operation

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

$$\begin{aligned} physics &\leftarrow \sigma_{\text{dept\_name} = \text{'Physics'}}(instructor) \\ music &\leftarrow \sigma_{\text{dept\_name} = \text{'Music'}}(instructor) \\ physics &\cap music \end{aligned}$$

- With the assignment operation, a query can be written as a sequential program 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.
- Another form of the rename operation:

$$\rho_{x(A_1,A_2,\ldots,A_n)}(E)$$

Equivalent Queries

- There are 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:

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

- Query 2:

$$\sigma_{\text{dept\_name} = \text{`Physics'}}\,(\sigma_{\text{ salary} > 90000}(instructor))$$

- The two queries are not identical; they are, however, equivalent – they give the same result on any database.

Equivalent Queries

- There are more than one way to write a query in relational algebra.
- Example: "Find information about courses taught by instructors in the Physics department."
- Query 1:

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

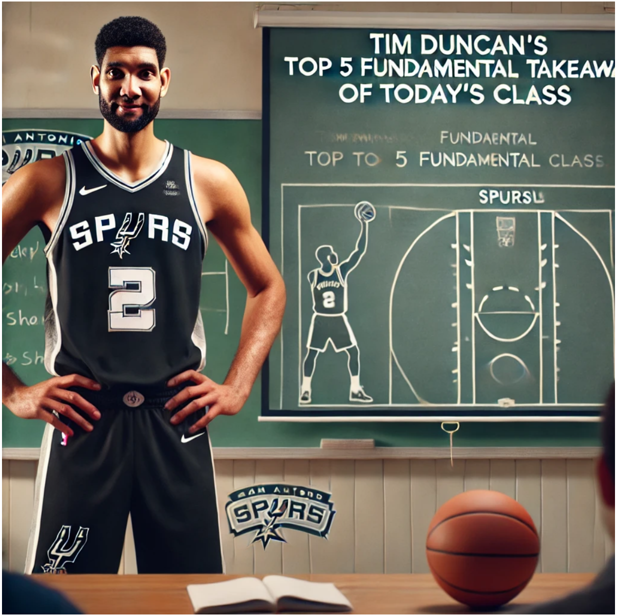
- Query 2:

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

- The two queries are not identical; they are, however, equivalent – they give the same result on any database.

End of Chapter 2.

TDT5FTOTTC


TIM DUNCAN'S
TOP 5 FUNDAMENTAL TAKEAW
OF TODAY'S CLASS
FUNDAENTAL
TOP TO 5 FUNDAMENTAL CLASS
SPURSL
SPURS
2
ANTONI
SPURS

Top 5 Fundamental Takeaways

Top 5 Fundamental Takeaways

5. **Structure of Relational Databases** – Data is organized in tables (relations) with attributes (columns) and tuples (rows), following the relational model.

Top 5 Fundamental Takeaways

5. **Structure of Relational Databases** – Data is organized in tables (relations) with attributes (columns) and tuples (rows), following the relational model.
4. **Database Schema and Keys** – A schema defines the database structure, while keys (superkeys, candidate keys, primary keys, and foreign keys) ensure data integrity and uniqueness.

Top 5 Fundamental Takeaways

5 **Structure of Relational Databases** – Data is organized in tables (relations) with attributes (columns) and tuples (rows), following the relational model.

4 **Database Schema and Keys** – A schema defines the database structure, while keys (superkeys, candidate keys, primary keys, and foreign keys) ensure data integrity and uniqueness.

3 **Equivalent Queries and Query Composition** – Multiple relational algebra expressions can yield the same result, allowing optimization through query rewriting and composition.

Top 5 Fundamental Takeaways

5 **Structure of Relational Databases** – Data is organized in tables (relations) with attributes (columns) and tuples (rows), following the relational model.

4 **Database Schema and Keys** – A schema defines the database structure, while keys (superkeys, candidate keys, primary keys, and foreign keys) ensure data integrity and uniqueness.

3 **Equivalent Queries and Query Composition** – Multiple relational algebra expressions can yield the same result, allowing optimization through query rewriting and composition.

2 **Join and Set Operations** – Joins combine related tuples from different tables, while set operations (union, intersection, and difference) manipulate relational data effectively.

Top 5 Fundamental Takeaways

5 **Structure of Relational Databases** – Data is organized in tables (relations) with attributes (columns) and tuples (rows), following the relational model.

4 **Database Schema and Keys** – A schema defines the database structure, while keys (superkeys, candidate keys, primary keys, and foreign keys) ensure data integrity and uniqueness.

3 **Equivalent Queries and Query Composition** – Multiple relational algebra expressions can yield the same result, allowing optimization through query rewriting and composition.

2 **Join and Set Operations** – Joins combine related tuples from different tables, while set operations (union, intersection, and difference) manipulate relational data effectively.

1 **Relational Query Languages** – Relational algebra and relational calculus provide formal methods to retrieve and manipulate data, with operations like selection (σ), projection (Π), and join ($\bowtie$).

Top 5 Fundamental Takeaways

5 **Structure of Relational Databases** – Data is organized in tables (relations) with attributes (columns) and tuples (rows), following the relational model.

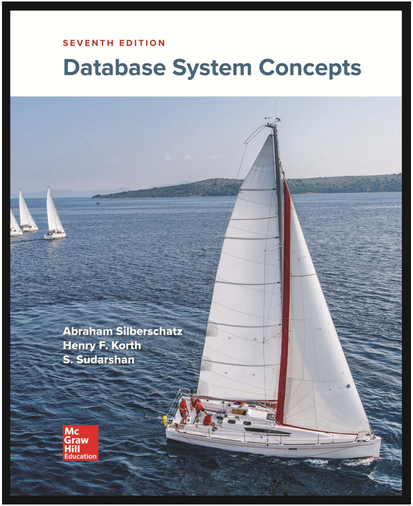
4 **Database Schema and Keys** – A schema defines the database structure, while keys (superkeys, candidate keys, primary keys, and foreign keys) ensure data integrity and uniqueness.

3 **Equivalent Queries and Query Composition** – Multiple relational algebra expressions can yield the same result, allowing optimization through query rewriting and composition.

2 **Join and Set Operations** – Joins combine related tuples from different tables, while set operations (union, intersection, and difference) manipulate relational data effectively.

1 **Relational Query Languages** – Relational algebra and relational calculus provide formal methods to retrieve and manipulate data, with operations like selection (σ), projection (Π), and join ($\bowtie$).

Database System Concepts



Content has been extracted from *Database System Concepts*, Seventh Edition, by Silberschatz, Korth and Sudarshan. Mc Graw Hill Education. 2019.
Visit `https://db-book.com/`.