

# CSE462/562: Database Systems (Spring 22)

## Lecture 3: Relational Model

2/8/2022

# Data abstraction

---

- A revisit of the personal spending DB
- What if we want to
  - record the payment method
  - track budgets/bills
  - link entries to itemized receipts
- Or what if
  - the program/spreadsheet is slow after a while
  - you are managing the spending DB for many people (e.g., a company)
- Constant changes in data management
  - for efficiency or for new application usages
  - impractical to break existing applications for every change

Date	Amount	Description
2/1	\$20.21	Grocery
2/2	\$10.54	Fast food
2/3	\$39.22	Cell phone bill
...		
2/27	\$33.00	Clothes

# Data abstraction

---

- Data abstraction
  - View level: what and how to present data to different applications/users



Logical Data Independence: ability to change logical schema without changing the external views and upper-level applications

- Logical level: what data are stored



Physical Data Independence: ability to change physical data storage without changing the logical schema

- Physical level: how data are stored

# Data models

---

- Data models are conceptual tools for
  - describing and defining the data abstractions
  - linking user's view to the bits stored in DBMS
- Many data models exist
  - **Relational model (aka structured data model)**
  - Entity-Relationship Model
  - Semi-structured data model
  - Graph data model
  - ...

We'll focus on relational model and Relational DataBase Management Systems (RDBMS) in this course:

It's the foundation of many other data models (including semi-structured data model, graph data model and etc.).

- The survey below gives a historical view of why relational models are successful
  - Joseph M. Hellerstein and Michael Stonebraker. What Goes Around Comes Around. Readings in Database Systems, 4th Edition (2005).
  - Keep it simple and stupid!

# Relational model

- Example: student records database

student

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

enrollment

sid	semester	cno	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7
101	s21	560	3.3
102	f21	560	4.0
103	s22	460	2.7
101	f21	560	3.3
103	f21	250	4.0

# Relational model

---

- Relational database: a collection of named **relations** (aka **tables**)
- Relation: a set of **records** (aka **tuples**) – no duplicates
  - In reality: multi-set semantics are more prevalent – allow duplicates
- Record: a sequence of values
  - represents relationships among values
- Two concepts
  - **Database Schema**: names of the relations + names and types of the columns + constraints
    - e.g., student(sid: integer, name: string, login: string, major: string, adm\_year: date)
    - each named column is also called an attribute or a field
  - **Database instance**: a snapshot of the data at a time point
    - e.g., the specific data in our student record database example

# Relational model

## Database schema

student(sid: integer, name: string, login: string, major: string, adm\_year: date)  
enrollment(sid: integer, semester: string, cno: integer, grade: float)

Relation (schema)

Relation (instance)

student

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

Record

Column

Database instance

enrollment

sid	semester	cno	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7
101	s21	560	3.3
102	f21	560	4.0
103	s22	460	2.7
101	f21	560	3.3
103	f21	250	4.0

# Integrity constraints

---

- Key constraints
  - **Superkey**: a set of columns that uniquely identify a record
    - e.g.,  $\{sid\}$  is a superkey of **student** relation;  $\{sid, name\}$  is too;
    - e.g.,  $\{sid, semester, cno\}$  is a superkey of **enrollment** relation; but  $\{sid, cno\}$  is not
    - Has nothing to do with specific instances
      - $\{sid, cno\}$  is not a superkey even if no one's ever taken a course twice
      - but it will be if the university policy prohibits retaking the same course
  - **Candidate key**: a superkey  $K$  s. t.  $\nexists K' \subset K: K'$  is a superkey
    - e.g.,  $\{sid\}$  and  $\{login\}$  are both candidate keys of **student**;  $\{sid, login\}$  is not
  - **(Primary) key**: a chosen candidate key by the database designer
    - e.g., student(sid: integer, name: string, login: string, major: string, adm\_year: date)

# Integrity constraints

- Foreign-key constraints
  - from attributes  $A$  of **referencing relation  $R$**  to primary key  $A'$  of **referenced relation  $R'$** :
  - such that for any DB instance, any value of  $A$  must appear in  $A'$  of some tuple in  $R'$

$R' = \text{student}$

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

$R = \text{enrollment}$

<u>sid</u>	<u>semester</u>	<u>cno</u>	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7
101	s21	560	3.3
102	f21	560	4.0
103	s22	460	2.7
101	f21	560	3.3
103	f21	250	4.0

# Integrity constraints

---

- Referential constraints
  - from attributes  $A$  of **referencing relation  $R$**  to *attributes  $A'$*  of **referenced relation  $R'$**
  - such that for any DB instance, any value of  $A$  must appear in  $A'$  of some tuple in  $R'$
  - Foreign-key constraints as a special case where  $A'$  is the primary key of  $R'$
- Other general constraints
- These are less supported by DBMS due to efficiency reasons

# Query Language

---

- Formal query languages
  - Relational algebra
    - Functional – describes how to query
  - Relational calculus
    - Declarative – describes what to query
  - No side effects! Does not include data definition, update, integrity checks, and etc.
  - Theoretical foundation of modern RDBMS; allows for query optimization
- Query language in practice: SQL (Structured Query Language)
  - Has its root in relational algebra and relational calculus
  - Includes many more beyond queries: imperative sublanguage, data definition, etc.

# Relational algebra

---

- There are 6 basic operators:
  - Selection  $\sigma$
  - Projection  $\pi$
  - Renaming  $\rho$
  - Cartesian product  $\times$
  - Set difference  $-$
  - Union  $\cup$
- The operators takes relations as input, and outputs a relation
  - Schemas of the input/output schema are fixed
  - Operators can be composed

# Selection

- $\sigma_P R$ 
  - Selects the records in relation  $R$  that satisfy a predicate  $P$
  - Output relation has the same schema as its input

student

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

$\sigma_{major='CS'} student$

# Projection

- $\pi_A R$ 
  - Retains only the attributes  $A$  in the output (i.e., “filters” on columns)
  - Schema of the result is exactly  $A$
- Projection in relational algebra *must* eliminate duplicates
  - In practice, no for using multi-set relational algebra, unless requested by the user.

student

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

$\pi_{major, adm\_year} student$



major	adm_year
CS	2021
CE	2020
CS	2020

# Renaming operator

- $\rho_{A_1 \rightarrow A'_1, A_2 \rightarrow A'_2, \dots} R$ 
  - Renames the attributes  $A_1, A_2, \dots$  to  $A'_1, A'_2, \dots$
  - Output schema is same as  $R$  except that the attributes are renamed

student

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

<u>sid</u>	fullname	ubitname	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

$\rho_{login \rightarrow ubitname, 2 \rightarrow fullname} student$

Positional notation

# Cartesian product

- $R_1 \times R_2$ 
  - Concatenates every pair of tuples  $t_1 \in R_1, t_2 \in R_2$  into a single tuple  $t \in R_1 \times R_2$
  - Output schema is the concatenation of the two input schemas
    - There might be naming conflicts, use renaming operator to avoid that

student

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021

enrollment

sid	semester	cno	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7



*student × enrollment*

sid	name	login	major	adm_year	sid	semester	cno	grade
100	Alice	alicer34	CS	2021	100	s22	562	2.0
100	Alice	alicer34	CS	2021	102	s22	562	2.3
100	Alice	alicer34	CS	2021	100	f21	560	3.7
101	Bob	bob5	CE	2020	100	s22	562	2.0
More results follows .....								

# Union

- $R \cup R'$ 
  - Union of two relations of the *compatible* schema
  - Output schema remains the same as inputs

Same number of columns. The  $i^{th}$  columns in both relations have the same type for all  $i$ .

student

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020

new\_students

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
102	Charlie	charlie7	CS	2021
104	Carol	carol20	CS	2021

$students \cup new\_students$



<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
104	Carol	carol20	CS	2021

# Set difference

- $R - R'$ 
  - Set difference of two relations of the *compatible* schema
  - Output schema remains the same as inputs

student

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020

new\_students

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
102	Charlie	charlie7	CS	2021
104	Carol	carol20	CS	2021

$students \cup new\_students$



<u>sid</u>	name	login	major	adm_year
101	Bob	bob5	CE	2020

# Assignment notation

---

- To help compose more complex queries with shared subqueries
  - $A \leftarrow Q$ :  $A$  denotes the output of relational algebra expression  $Q$
  - E.g.,

$studentInCS \leftarrow \pi_{major='CS'} student$   
 $students2021 \leftarrow \pi_{admyaer=2021} student$   
 $studentInCS \cup students2021$

# Compound operators

---

- Several useful compound operators
  - Join  $\bowtie$ 
    - Inner join
      - Natural join
    - Outer join
  - Set intersection  $\cap$
  - Division operator  $/$
  - ...
- All of them can be composed from the 6 basic operators
- Does not add expressiveness of the relational algebra

# Inner join

---

- $R \bowtie_P R' = \sigma_P(R \times R')$ 
  - Selecting records that satisfy the predicate  $P$  from  $R \times R'$
- Most common special case is *natural join*
$$R \bowtie R' = \pi_{A(R) \cup A(R')} \sigma_{\forall a \in A(R) \cap A(R') : R.a = R'.a} (R \times R')$$
  - $A(R)$  : attributes of  $R$
  - The predicate  $P$  is implicitly equality between common attributes of  $R$  and  $R'$
  - Projecting to all unique attributes of  $R$  and  $R'$  (only one copy for common attributes)
- Equi-join:  $P$  is conjunction of equality predicates
- Useful for denormalization

# Natural join

student S

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

enrollment E

sid	semester	cno	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7
101	s21	560	3.3



$$S \bowtie E = \pi_{S.sid, S.name, S.login, S.major, S.adm\_year, E.semester, E.cno, E.grade} \sigma_{S.sid=E.sid} S \times E$$


sid	name	login	major	adm_year	semester	cno	grade
100	Alice	alicer34	CS	2021	s22	562	2.0
100	Alice	alicer34	CS	2021	f21	560	3.7
101	Bob	bob5	CE	2020	s21	560	3.3
102	Charlie	charlie7	CS	2020	s22	562	3.7

# Inner join

enrollment E

sid	semester	cno	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7
101	s21	560	3.3
102	f21	560	4.0
103	s22	460	2.7
101	f21	560	3.3
103	f21	250	4.0

sid	cno	grade	sid	cno	grade
100	562	2.0	102	562	2.3
100	560	3.7	102	560	4.0
101	560	3.3	100	560	3.7
101	560	3.3	102	560	4.0
100	560	3.7	102	560	4.0


$$E_1, E_2 \leftarrow \pi_{sid, cno, grade} E$$
$$E_1 \bowtie_{E_1.cno=E_2.cno \wedge E_1.grade < E_2.grade} E_2$$

# Outer join

- Inner join results  $\cup$  tuples without matches (augmented with NULLs)

- Types of outer joins

- Left outer join  $R \bowtie_p R' = R \bowtie_P R' \cup \left( (R - \pi_{A(R)} R \bowtie_P R') \times \{(\overbrace{\phi, \phi, \dots, \phi}^{|A(R')| \text{ NULLs}})\} \right)$
- Right outer join  $R \bowtie_p R' = R \bowtie_P R' \cup \left( \{(\underbrace{\phi, \phi, \dots, \phi}_{|A(R)| \text{ NULLs}})\} \times (R' - \pi_{A(R')} R \bowtie_P R') \right)$
- Full outer join

$$R \bowtie_p R' = R \bowtie_p R' \cup R \bowtie_P R'$$

- Useful for preserving all unique values in one or both relations

# Outer join

student S

sid	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020
102	Charlie	charlie7	CS	2021
103	David	davel	CS	2020

enrollment E

sid	semester	cno	grade
100	s22	562	2.0
102	s22	562	2.3
100	f21	560	3.7
101	s21	560	3.3



$$S \bowtie_{S.sid=E.sid} E$$

S.sid	S.name	S.login	S.major	S.adm_year	E.sid	E.semester	E.cno	E.grade
100	Alice	alicer34	CS	2021	100	s22	562	2.0
100	Alice	alicer34	CS	2021	100	f21	560	3.7
101	Bob	bob5	CE	2020	101	s21	560	3.3
102	Charlie	charlie7	CS	2020	102	s22	562	2.3
103	David	davel	CS	2020	NULL	NULL	NULL	NULL

# Other useful operators

- Set intersection:  $R \cap R' = R - (R - R')$

student

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
101	Bob	bob5	CE	2020

new\_students

<u>sid</u>	name	login	major	adm_year
100	Alice	alicer34	CS	2021
102	Charlie	charlie7	CS	2021
104	Carol	carol20	CS	2021

$students \cap new\_students$



<u>sid</u>	name	login	major	adm_year
100	Alic	alicer34	CS	2021

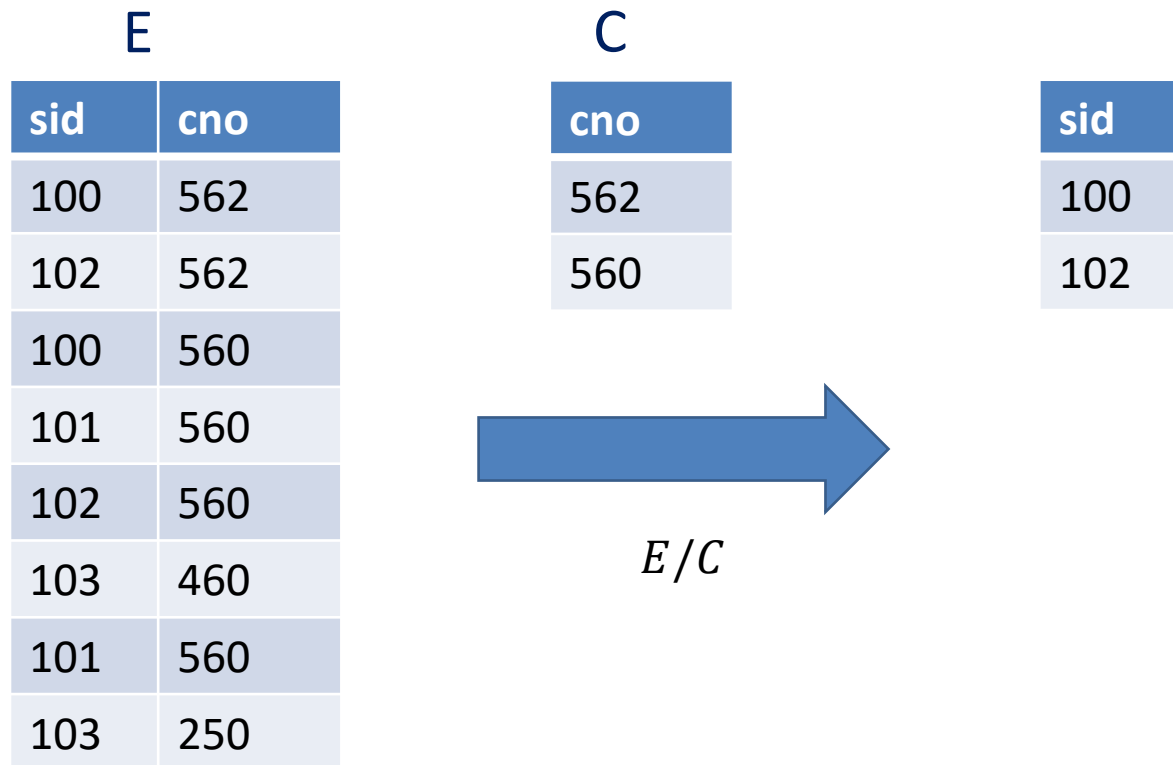
# Other useful operators

---

- Division:  $R/R'$ 
  - Attributes of  $R'$  must be a subset of the attributes of  $R$
  - The output schema of division is the extra attributes  $A_o = A(R) - A(R')$  of  $R$
  - $R/R'$  contains all tuples  $t_o \in \pi_{A_o} R$  such that for every  $t' \in R'$ , the concatenation  $t_o \circ t' \in R$
- Useful for expressing “for all” queries like
  - Find all students who have enrolled in both CSE560 and CSE562

# Division

Find all students who have enrolled in both CSE560 and CSE562



# Division

---

- Exercise: how to express  $R/R'$  using the basic operators
- Idea: find all  $t_o \in \pi_{A_o}R$  such that some combination  $t_o \circ t'$  is missing from  $R$
- $$R/R' = \pi_{A_o}R - \pi_{A_o} \left( (\pi_{A_o}R \times R') - R \right)$$

# Summary

---

- Relational Algebra: a functional language
  - A few operators that map relations to another relation
  - A closed set of operators that may be composed.
- Basic ops:  $\sigma, \pi, \times, \cup, -$
- Other important compound ops:  $\bowtie, \bowtie_{\text{left}}, \bowtie_{\text{right}}, \bowtie_{\text{full}}, \cap, /$
- Next time: SQL