Relational Model & Algebra

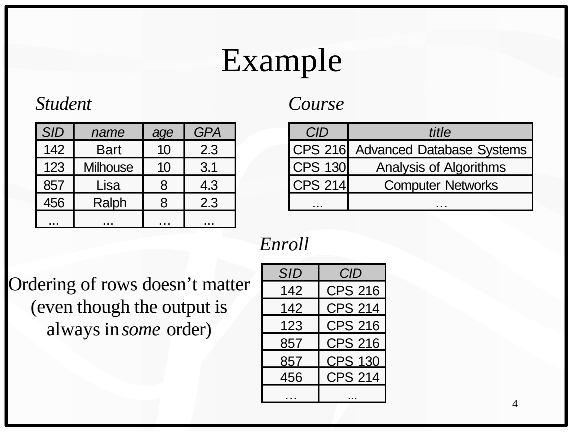
CPS 216
Advanced Database Systems

Announcements

- Lecture notes
  - "Notes" version (incomplete) available in the morning on the day of lecture
  - "Slides" version (complete) available after the lecture
- We are working on installing IBM DB2!
  - Help needed
  - Good learning experience
- Reminder: check CourseInfo for announcements!

Relational data model

- A database is a collection of relations (or tables)
- Each relation has a list of attributes (or columns)
  - Set-valued attributes not allowed
- Each attribute has a domain (or type)
- Each relation contains a set of tuples (or rows)
  - Duplicates not allowed
- Simplicity is a virtue!



Example

Student

SID	*name*	*age*	*GPA*
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3
…	…	…	…

Course

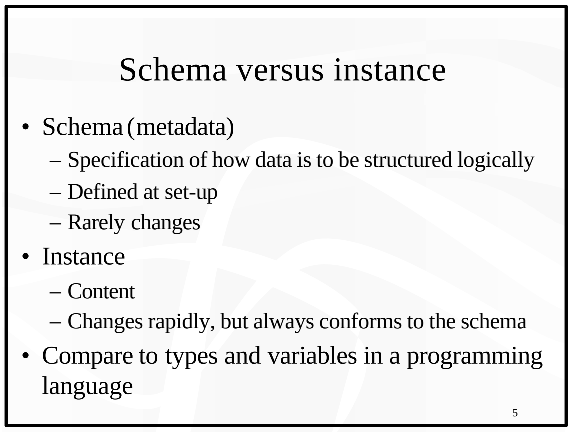
CID	*title*
CPS 216	Advanced Database Systems
CPS 130	Analysis of Algorithms
CPS 214	Computer Networks
…	…

Enroll

SID	*CID*
142	CPS 216
142	CPS 214
123	CPS 216
857	CPS 216
857	CPS 130
456	CPS 214
…	…

Ordering of rows doesn't matter (even though the output is always in *some* order)

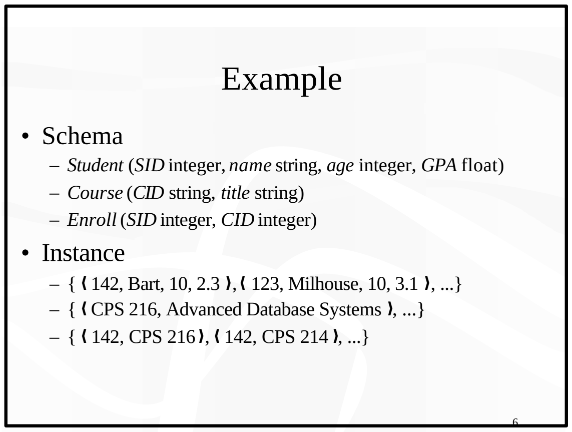
4



Schema versus instance

- Schema (metadata)
  - Specification of how data is to be structured logically
  - Defined at set-up
  - Rarely changes
- Instance
  - Content
  - Changes rapidly, but always conforms to the schema
- Compare to types and variables in a programming language

5

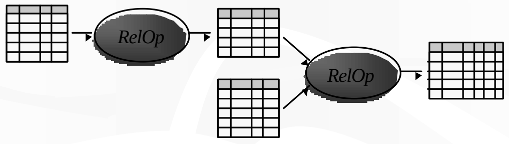


Example

- Schema
  - *Student* (*SID* integer, *name* string, *age* integer, *GPA* float)
  - *Course* (*CID* string, *title* string)
  - *Enroll* (*SID* integer, *CID* integer)
- Instance
  - { ⟨ 142, Bart, 10, 2.3 ⟩, ⟨ 123, Milhouse, 10, 3.1 ⟩, ...}
  - { ⟨ CPS 216, Advanced Database Systems ⟩, ...}
  - { ⟨ 142, CPS 216 ⟩, ⟨ 142, CPS 214 ⟩, ...}

6

Relational algebra operators



- Core set of operators:
  - Selection, projection, cross product, union, difference, and renaming
- Additional, derived operators:
  - Join, natural join, intersection, etc.

7

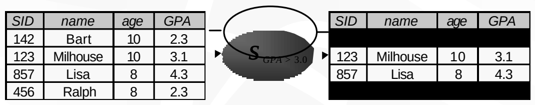
Selection

- Input: a table *R*
- Notation: $S_p(R)$
  - *p* is called a selection condition/predicate
- Purpose: filter rows according to some criteria
- Output: same columns as *R*, but only rows of *R* that satisfy *p*

8

Selection example

- Students with GPA higher than 3.0



SID	*name*	*age*	*GPA*
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3

$S_{GPA > 3.0}$

SID	*name*	*age*	*GPA*
123	Milhouse	10	3.1
857	Lisa	8	4.3

9

More on selection

- Selection predicate in general can include any column of R, constants, comparisons such as =, ≤, etc., and Boolean connectives $\wedge$, $\vee$, and $\neg$
  - Example: straight A students under 18 or over 21
- But you must be able to evaluate the predicate over a single row
  - Example: student with the highest GPA?

10

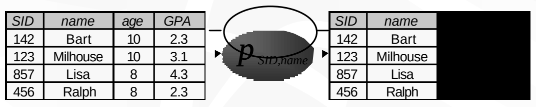
Projection

- Input: a table *R*
- Notation: $p_L(R)$
  - *L* is a list of columns in *R*
- Purpose: select columns to output
- Output: same rows, but only the columns in *L*

11

Projection example

- IDs and names of all students



SID	name	age	GPA
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3

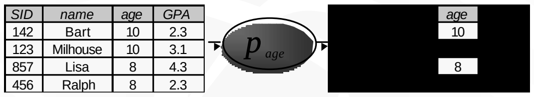
$p_{SID,name}$

SID	name
142	Bart
123	Milhouse
857	Lisa
456	Ralph

12

More on projection

- Duplicate output rows must be removed
  - Example: age distribution of students



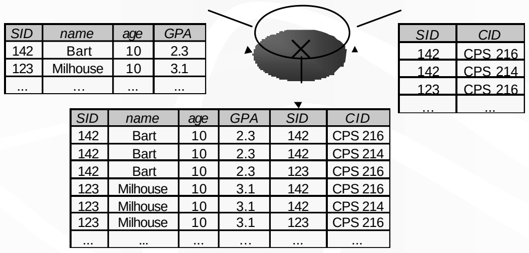
SID	name	age	GPA
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3

age
10
8

Cross product

- Input: two tables *R* and *S*
- Notation: $R \times S$
- Purpose: pairs rows from two tables
- Output: for each row *r* in *R* and each row *s* in *S*, output a row *rs* (concatenation of *r* and *s*)

Cross product example



SID	name	age	GPA
142	Bart	10	2.3
123	Milhouse	10	3.1
…	…	…	…

SID	CID
142	CPS 216
142	CPS 214
123	CPS 216
…	…

SID	name	age	GPA	SID	CID
142	Bart	10	2.3	142	CPS 216
142	Bart	10	2.3	142	CPS 214
142	Bart	10	2.3	123	CPS 216
123	Milhouse	10	3.1	142	CPS 216
123	Milhouse	10	3.1	142	CPS 214
123	Milhouse	10	3.1	123	CPS 216
…	…	…	…	…	…

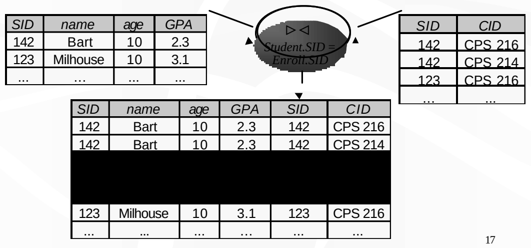
Derived operator: join

- Input: two tables *R* and *S*
- Notation: $R \bowtie_p S$
  - *p* is called a join condition/predicate
- Purpose: related rows from two tables according to some criteria
- Output: for each row *r* in *R* and each row *s* in *S*, output a row *rs* (concatenation of *r* and *s*) if *r* and *s* satisfy *p*
- Shorthand for

16

Join example

- Info about students, plus CIDs of their courses



SID	*name*	*age*	*GPA*
142	Bart	10	2.3
123	Milhouse	10	3.1
...	...	...	...

SID	*CID*
142	CPS 216
142	CPS 214
123	CPS 216
...	...

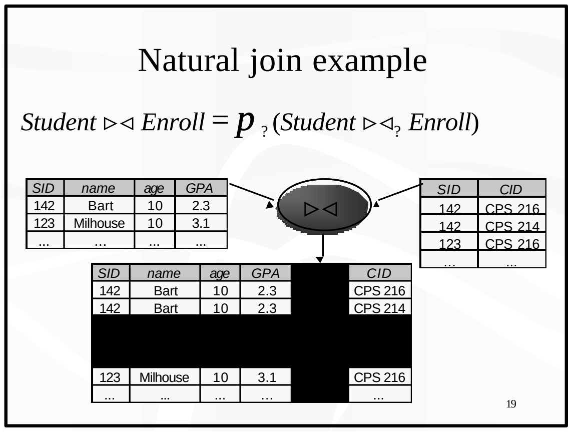
SID	*name*	*age*	*GPA*	*SID*	*CID*
142	Bart	10	2.3	142	CPS 216
142	Bart	10	2.3	142	CPS 214
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
123	Milhouse	10	3.1	123	CPS 216
...	...	...	...	...	...

17

Derived operator: natural join

- Input: two tables *R* and *S*
- Notation: $R \bowtie S$
- Purpose: related rows from two tables, and
  - Enforce equality on all common attributes
  - Eliminate one copy of common attributes
- Shorthand for

18



Natural join example

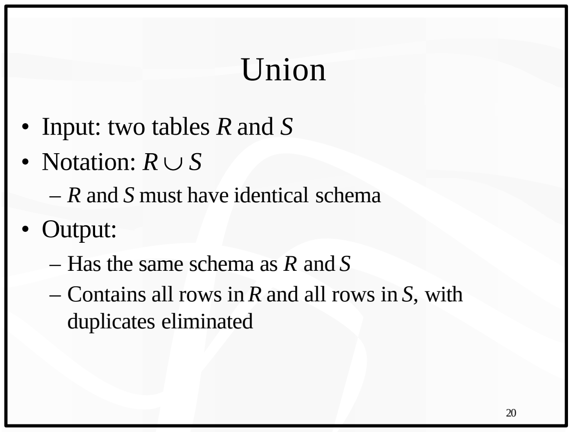
Student ⋈ *Enroll* = $p_{?}$(*Student* $⋈_{?}$ *Enroll*)

SID	name	age	GPA
142	Bart	10	2.3
123	Milhouse	10	3.1
…	…	…	…

SID	CID
142	CPS 216
142	CPS 214
123	CPS 216
…	…

SID	name	age	GPA		CID
142	Bart	10	2.3		CPS 216
142	Bart	10	2.3		CPS 214
123	Milhouse	10	3.1		CPS 216
…	…	…	…		…

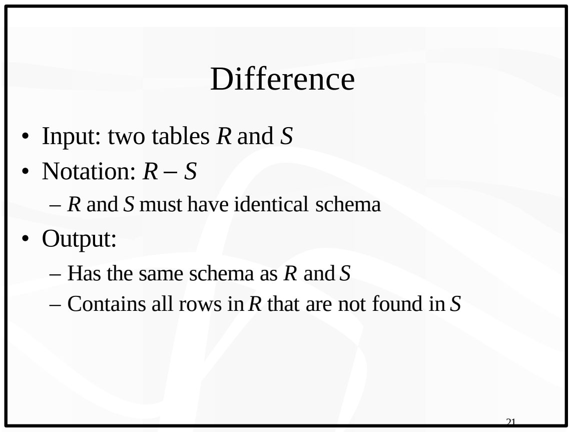
19



Union

- Input: two tables *R* and *S*
- Notation: *R* ∪ *S*
  - *R* and *S* must have identical schema
- Output:
  - Has the same schema as *R* and *S*
  - Contains all rows in *R* and all rows in *S*, with duplicates eliminated

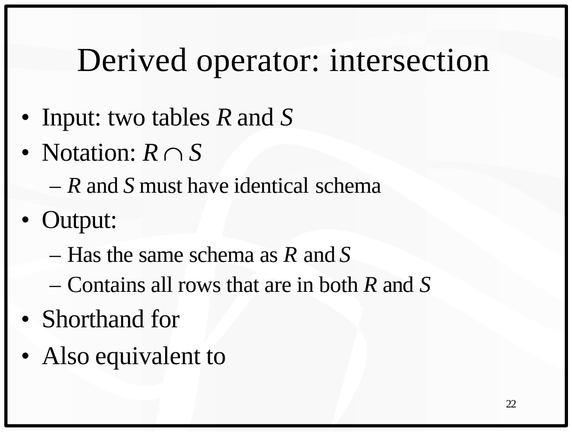
20



Difference

- Input: two tables *R* and *S*
- Notation: *R* – *S*
  - *R* and *S* must have identical schema
- Output:
  - Has the same schema as *R* and *S*
  - Contains all rows in *R* that are not found in *S*

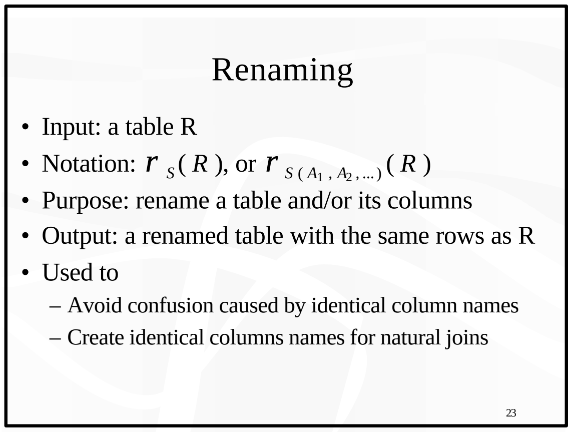
21



Derived operator: intersection

- Input: two tables *R* and *S*
- Notation: $R \cap S$
  - *R* and *S* must have identical schema
- Output:
  - Has the same schema as *R* and *S*
  - Contains all rows that are in both *R* and *S*
- Shorthand for
- Also equivalent to

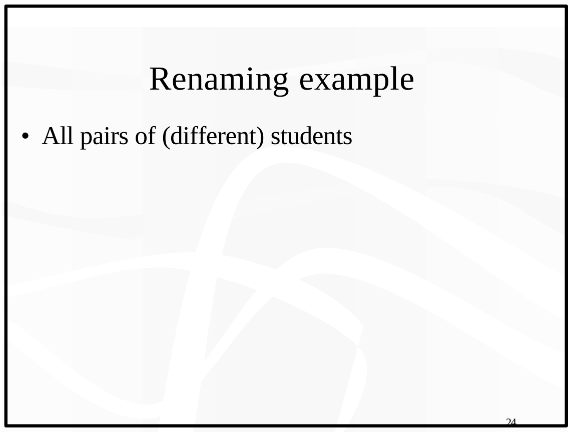
22



Renaming

- Input: a table R
- Notation: $r_{S}(R)$, or $r_{S(A_1, A_2, ...)}(R)$
- Purpose: rename a table and/or its columns
- Output: a renamed table with the same rows as R
- Used to
  - Avoid confusion caused by identical column names
  - Create identical columns names for natural joins

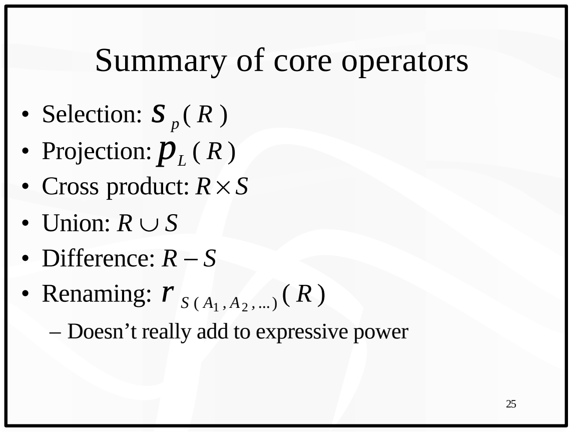
23



Renaming example

- All pairs of (different) students

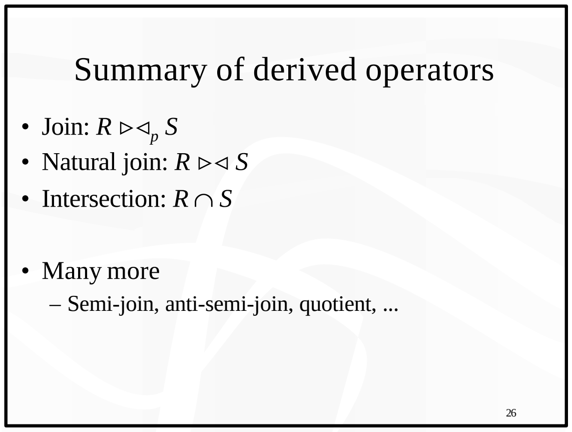
24



Summary of core operators

- Selection: $s_p(R)$
- Projection: $p_L(R)$
- Cross product: $R \times S$
- Union: $R \cup S$
- Difference: $R - S$
- Renaming: $r_{S(A_1, A_2, \ldots)}(R)$
  - Doesn't really add to expressive power

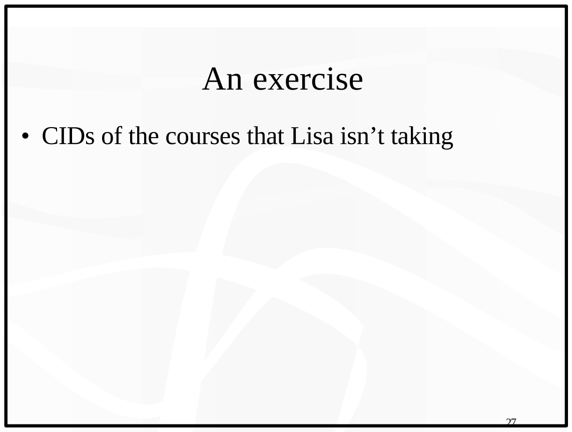
25



Summary of derived operators

- Join: $R \bowtie_p S$
- Natural join: $R \bowtie S$
- Intersection: $R \cap S$
- Many more
  - Semi-join, anti-semi-join, quotient, ...

26



An exercise

- CIDs of the courses that Lisa isn't taking

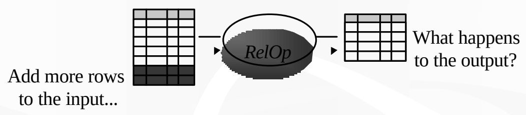
27

A trickier exercise

- Who has the highest GPA?

28

Monotone operators



- If some old output rows must be removed
  - Then the operator is non-monotone
- Otherwise the operator is monotone
  - That is, old output rows remain "correct" when more rows are added to the input
  - Formally, $R \subseteq R' \Rightarrow RelOp(R) \subseteq RelOp(R')$

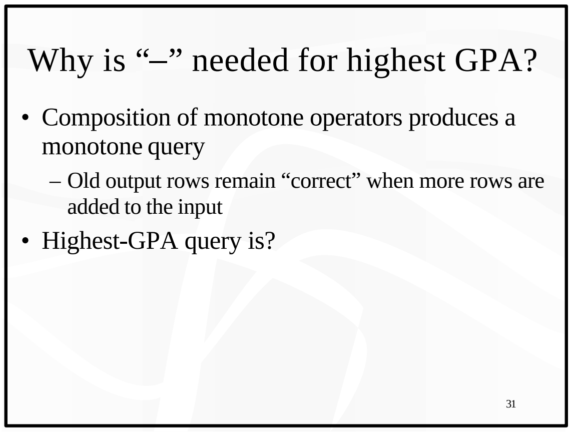
29

Classification of relational operators

Monotone ✓ Non-monotone ✗

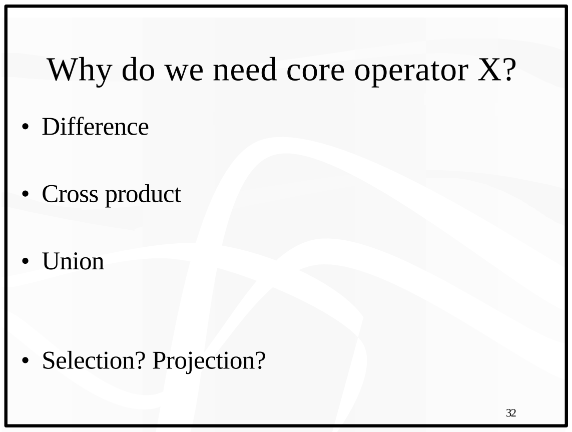
- Selection: $S_p (R)$ ✓
- Projection: $p_L (R)$ ✓
- Cross product: $R \times S$ ✓
- Union: $R \cup S$ ✓
- Difference: $R - S$ ✗ (Not with respect to S)

30



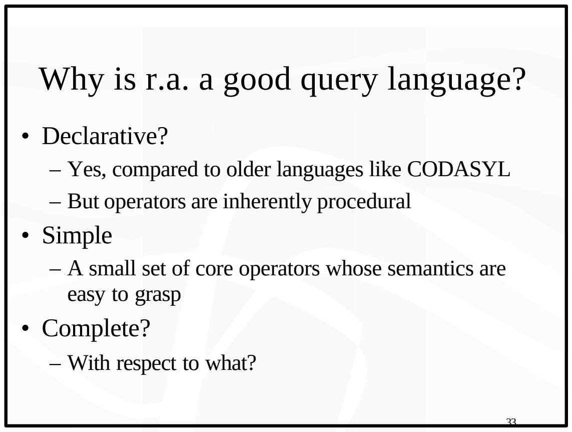
Why is "–" needed for highest GPA?

- Composition of monotone operators produces a monotone query
  - Old output rows remain "correct" when more rows are added to the input
- Highest-GPA query is?



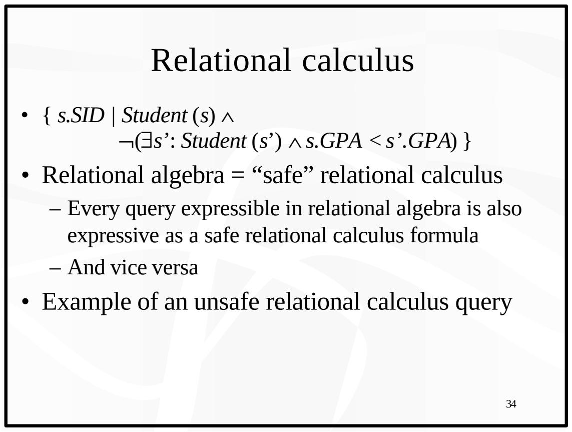
Why do we need core operator X?

- Difference
- Cross product
- Union
- Selection? Projection?



Why is r.a. a good query language?

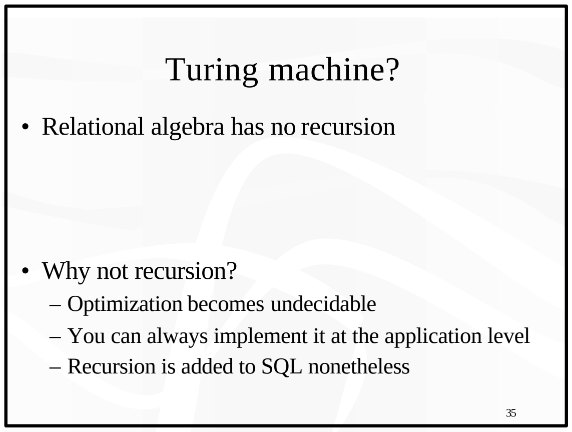
- Declarative?
  - Yes, compared to older languages like CODASYL
  - But operators are inherently procedural
- Simple
  - A small set of core operators whose semantics are easy to grasp
- Complete?
  - With respect to what?



Relational calculus

- $\{ s.SID \mid Student(s) \wedge \neg(\exists s': Student(s') \wedge s.GPA < s'.GPA) \}$
- Relational algebra = “safe” relational calculus
  - Every query expressible in relational algebra is also expressive as a safe relational calculus formula
  - And vice versa
- Example of an unsafe relational calculus query

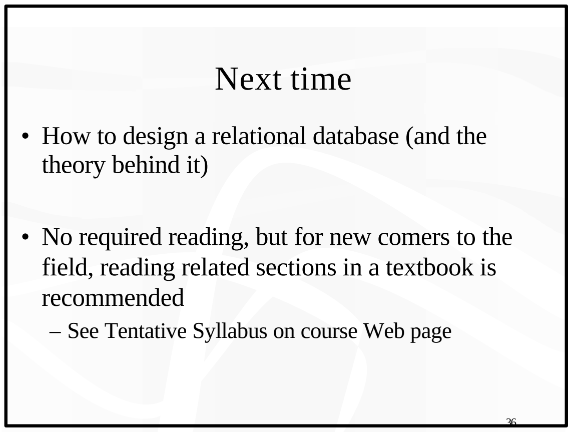
34



Turing machine?

- Relational algebra has no recursion
- Why not recursion?
  - Optimization becomes undecidable
  - You can always implement it at the application level
  - Recursion is added to SQL nonetheless

35



Next time

- How to design a relational database (and the theory behind it)
- No required reading, but for new comers to the field, reading related sections in a textbook is recommended
  - See Tentative Syllabus on course Web page

36