CSE 180, Database Systems I
Shel Finkelstein, UC Santa Cruz

Lecture 7

Relational Algebra, Revisited (Chapter 2); parts of Sections 16.1-16.2.2, on Optimization and Transformations

Database System Concepts, 7th Ed.

Important Notices

- Lab2 scores have been unmuted on Canvas.
  - If you have questions about your scores, ask them by Wed, Feb 19.
- Lab3 has been posted on Piazza, with a lab3_create.sql file that creates tables for Lab3 and a lab3_data_loading.sql file that is **needed** to do Lab3.
  - Lab3 pdf and the create and load file have been revised, as of Feb 10.
    - *Lab3 load file was revised on Saturday, Feb 15.*
  - A transaction (from this lecture, Lecture 6) is also needed for Lab3.
  - Lab3 is due on **Sunday, Feb 23, 11:59pm**.
  - You have 3 weeks to complete Lab3 because the CSE 180 Midterm was on Wednesday, Feb 12.
- The third Gradiance Assignment, ”CSE 180 Winter 2020 #3", is due on Saturday, Feb 22, 11:59pm.
- Attend Lectures, Lab Sections, Office Hours and LSS Tutoring.
  - See Piazza notices about LSS Tutoring with Jeshwanth Bheemanpally.

Important Notices

- Midterm was on **Wednesday, Feb 12.**
  - Midterm and Midterm answers were posted on Piazza that same day.
  - Answers were discussed in class on Friday, Feb 14.
    - Zoom recording of that class is on Piazza.
  - Grades were unmuted on Canvas on Monday, Feb 17.
  - There is a generous curve, 0.9 * X + 12, so 80 becomes 84.
    - Even though exam had 104 points, that curved grade is treated as 84 out of 100, not as 84 out of 104.
    - Canvas shows raw grade, not curved grade.
  - Midterms will be/were returned to students during the last 10 minutes of class on Wednesday, Feb 19.
    - I'll also bring them to class on Friday, Feb 21, but not thereafter.
  - Please speak to your TA if you have questions about grading, then to me if necessary. **Never** ask the Readers about grades!
- No Office Hour for me on Wed, Feb 26, 3:00-4:00pm; instead, my Office Hour that week will be Mon, Feb 24, 3:00-4:00pm in E2-249B.

Chapter 16: Query Optimization

We will only cover the following topics from Chapter 16 of our textbook, as well as some basic topics on Relational Algebra not in the textbook.

- Reminder about Relational Algebra
  - Natural Join and Semi-Join
- Introduction to Query Execution
- Transformation of Relational Expressions

Other Sections of Chapter 16 are covered in CSE 181, not in 180.

- Catalog Information for Cost Estimation
- Statistical Information for Cost Estimation
- Cost-based optimization
- Dynamic Programming for Choosing Evaluation Plans
- Materialized views

Reminder: Relational Algebra

- Relational Algebra is a procedural language consisting of 5 basic operations, each of which takes one or two relations as its input, and produces one new relation as its result.
- 5 basic operations
  - select: σ
  - project: Π
  - union: $\cup$
  - set-difference: $-$
  - Cartesian Product: x

We're only dealing with **set-oriented** Relational Algebra, in which there are no duplicates.

- We'll also use 2 "housekeeping" operations that make queries easier to write.
  - assignment: $\leftarrow$
  - rename: ρ

Relation Algebra Operators

- Ted Codd proved that the relational algebra operators (σ , π , x , U , and -) are independent of each other. That is, you can't define any of these 5 operators using the others.

- However, there are other important operators that can be expressed using (σ , π , x , U , -).
  - Join, Natural Join, Semi-Join, Outer Join
    - Join is discussed in this Lecture.
  - Set Intersection, which is also discussed in this Lecture.
  - Division (/ or $\div$), which we won't discuss in this class.

Natural Join: $r \bowtie s$

- Natural Join is a binary operator:
  - Input: Two relations r(R) and s(S) where $\{ A_1, \ldots, A_k \}$ is the set of common attributes (column names) in both R and S.
  - Output: Relation whose attributes are R U S. In other words, the attributes consists of the attributes in R U S without repeats of the common attributes $\{ A_1, \ldots, A_k \}$

- Meaning:

$$r \bowtie s = \pi_{R \cup S} (\sigma_{r.A1 = s.A1 \text{ AND } r.A2 = s.A2 \text{ AND } \ldots \text{ AND } r.Ak = s.Ak} (r \times s))$$

1. Compute R x S
2. Keep only those tuples in R x S satisfying:
   r.A1 = s.A1 AND r.A2 = s.A2 AND … AND r.Ak = s.Ak
3. Output is projection of those tuples on the set of attributes in R U S, without repeats of the attributes that appear in both

Semi-Join: $r \ltimes s$

- We'll define the Semi-Join of relations r(R) and s(S).
- Meaning: $r \ltimes s = \pi_R (r \bowtie s)$

1. Compute Natural Join of r and s
2. Output the projection of that on just the attributes of **r**

- Find all sections that someone is taking.
  section $\ltimes$ takes

Common attributes are course_id, sec_id, semester and year

- Find all departments that have at least one instructor.
  department $\ltimes$ instructor

Common attribute is dept_name

- How does Semi-Join relate to EXISTS in SQL?

Practice Homework for Relational Algebra

Sailors(sid, sname, rating, age) // sailor id, sailor name, rating, age
Boats(bid, bname, color) // boat id, boat name, color of boat
Reserves(sid, bid, day) // sailor id, boat id, date that sid reserved bid.

- Use **Relational Algebra** to write the following **8 queries**.
- Relational Algebra solutions for the problems in **red** (that are on the next slide) will be posted on Piazza.
- How might you optimize execution of queries using ideas in this Lecture?

1. Find the names of sailors who reserved boat 103.

2. Find the colors of boats reserved by Lubber.

3. Find the names of sailors who reserved at least one boat.

Practice Homework for Relational Algebra (Cont.)

4. Find the names of sailors whose age > 20 and who have not reserved any boats.

5. Find the names of sailors who have reserved a red or a green boat.

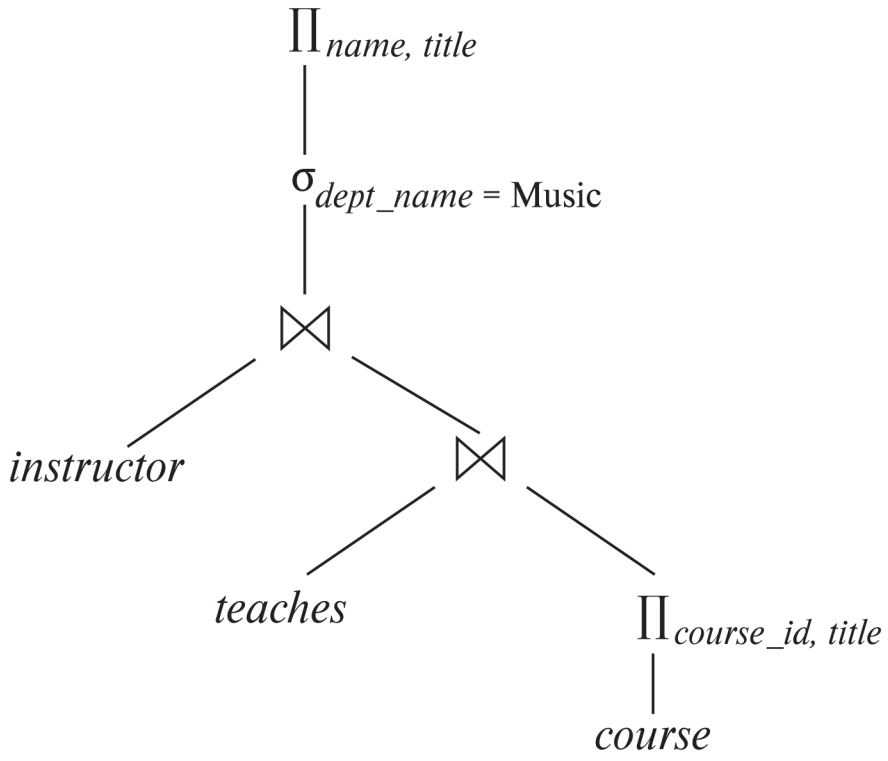
6. Find the names of sailors who have reserved a red and a green boat.

7. Find the names of sailors who have reserved at least 2 different boats.

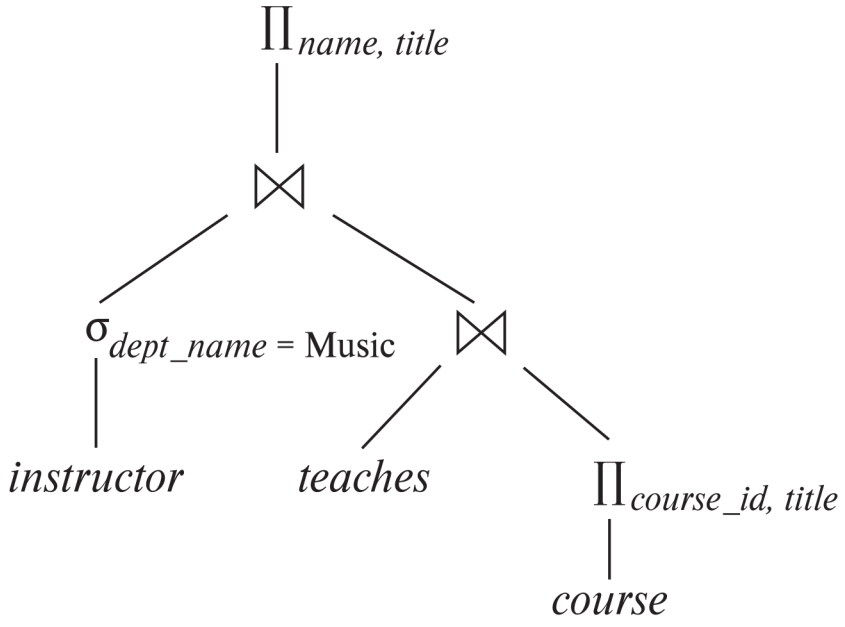
8. Find the names of sailors who have reserved exactly 2 different boats.

Introduction to Query Execution

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation



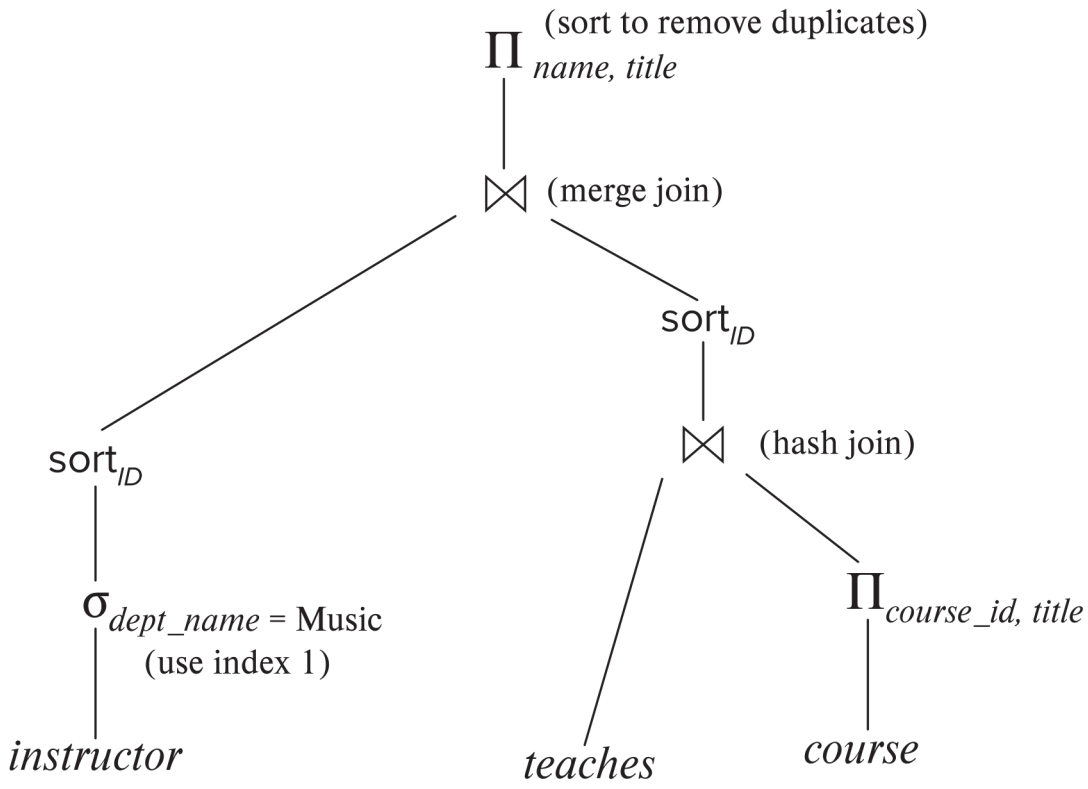
(a) Initial expression tree



(b) Transformed expression tree

Introduction to Query Execution (Cont.)

- An **evaluation plan** defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated.



- For PostgreSQL, you can view query execution plans using the EXPLAIN statement.

Introduction to Query Execution (Cont.)

- Cost difference between evaluation plans for a query can be enormous
  - E.g., seconds vs. days in some cases
- Steps in **cost-based query optimization**
  1. Generate logically equivalent expressions using **equivalence rules.**
  2. Annotate resultant expressions to get alternative query plans,
  3. Choose the cheapest plan based on **estimated cost.**
- Estimation of plan cost is described in CSE 181, which is a systems course that is much tougher than CSE 180. (Spring 2020)

Generating Equivalent Expressions

Transformation of Relational Expressions

- Two relational algebra expressions are said to be **equivalent** if the two expressions generate the same set of tuples on every **legal** database instance
  - Note: order of tuples in result is irrelevant.
  - We don't care if they generate different results on "illegal" databases that violate integrity constraints.
- In SQL, inputs and outputs are multisets of tuples
  - Two expressions in the multiset version of the relational algebra are said to be equivalent if the two expressions generate the same multiset of tuples on every legal database instance.
- An **equivalence rule** says that two relational algebra expressions are equivalent.
  - Can replace first expression by the second expression, and vice versa.

Some Terminology

- Commutative Operations
  - Addition is Commutative, since a+b = b+a
  - But Subtraction isn't Commutative, since a-b ≠ b-a
- Associative Operations
  - Addition is Associative, since (a + b) + c = a + (b + c)
  - But Subtraction isn't Associative, since sometimes (a - b) - c ≠ a - (b - c)
- Distributive Operations
  - Multiplication distributes over Addition, since a * (b + c) = a * b + a * c
  - But Addition doesn't distribute over Multiplication, since sometimes a + (b * c) ≠ (a + b) * (a + c)

Equivalence Rules

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections.

$$\sigma_{\theta_1 \wedge \theta_2}(E) \equiv \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Selection operations are Commutative.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) \equiv \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Only the last in a sequence of projection operations is needed; the other projection operations can be omitted.

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) \equiv \Pi_{L_1}(E)$$

where $L_1 \subseteq L_2 \ldots \subseteq L_n$

4. Selections can be combined with Cartesian products to form a Theta Join.

$$\sigma_{\theta}(E_1 \times E_2) \equiv E_1 \bowtie_{\theta} E_2$$

Equivalence Rules (Cont.)

5. Theta-Join is Commutative, and Natural Join is also Commutative.

$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$$

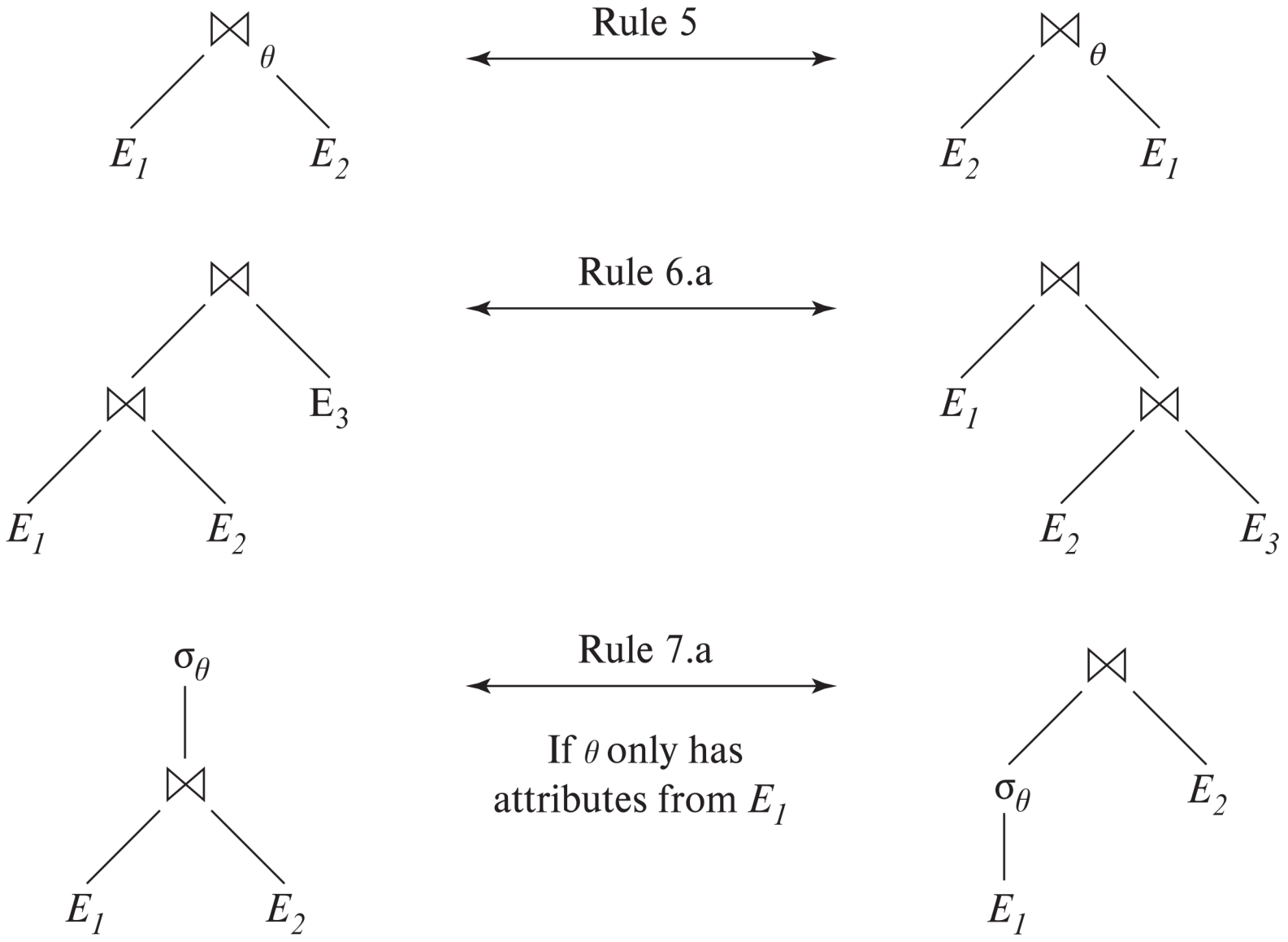
6. (a) Natural Join is Associative:

$$(E_1 \bowtie E_2) \bowtie E_3 \equiv E_1 \bowtie (E_2 \bowtie E_3)$$

(b) Cartesian product is Associative:

$$(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$$

Pictorial Depiction of Equivalence Rules



Equivalence Rules (Cont.)

7. The Selection operation distributes over the Theta Join operation under the following two conditions:
   (a) When all the attributes in θ_0 involve only the attributes of one of the expressions (E_1) being joined.

$$\sigma_{\theta_0}(E_1 \bowtie_{\theta} E_2) \equiv (\sigma_{\theta_0}(E_1)) \bowtie_{\theta} E_2$$

   (b) When θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2.

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) \equiv (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))$$

Equivalence Rules (Cont.)

8. The set operations Union and Intersection are Commutative.

$$E_1 \cup E_2 \equiv E_2 \cup E_1$$
$$E_1 \cap E_2 \equiv E_2 \cap E_1$$

(Set Difference is not Commutative.)

9. Set Union and Intersection are associative.

$$(E_1 \cup E_2) \cup E_3 \equiv E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 \equiv E_1 \cap (E_2 \cap E_3)$$

(Set Difference is not Associative.)

Equivalence Rules (Cont.)

10. The selection operation distributes over $\cup$, $\cap$ and $-$.

    a. $\sigma_\theta (E_1 \cup E_2) \equiv \sigma_\theta (E_1) \cup \sigma_\theta(E_2)$

    b. $\sigma_\theta (E_1 \cap E_2) \equiv \sigma_\theta (E_1) \cap \sigma_\theta(E_2)$

    c. $\sigma_\theta (E_1 - E_2) \equiv \sigma_\theta (E_1) - \sigma_\theta(E_2)$

    d. $\sigma_\theta (E_1 \cap E_2) \equiv \sigma_\theta(E_1) \cap E_2$

    e. $\sigma_\theta (E_1 - E_2) \equiv \sigma_\theta(E_1) - E_2$

Assumes attribute names are same in both relations.

Equivalences 10d and 10e do not hold for $\cup$

11. The projection operation distributes over union

$$\Pi_L(E_1 \cup E_2) \equiv (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

More Complex Queries

- Relational operators can be composed to form more complex queries. We have already seen examples of this in SQL.
- Just for the next few slide, we're going to work with a different, much simpler university database.

Enrollments(esid, ecid, grade)

Courses(cid, cname, instructor-name)

- Query 1: Find the student id, grade and instructor where the student had a grade that was more than 80 points in a course.

$$\sigma_{grade>80} (\pi_{esid,\ grade,\ instructor\text{-}name} (\sigma_{Enrollments.ecid = Courses.cid} (Enrollments \times Courses)))$$

Query 2

Enrollments(esid, ecid, grade)

Courses(cid, cname, instructor-name)

Students(sid, sname)

- Find the student name and course name where the student had a grade that was more than 80 points in a course.

$\pi_{\text{Students.sname, Courses.cname}}$

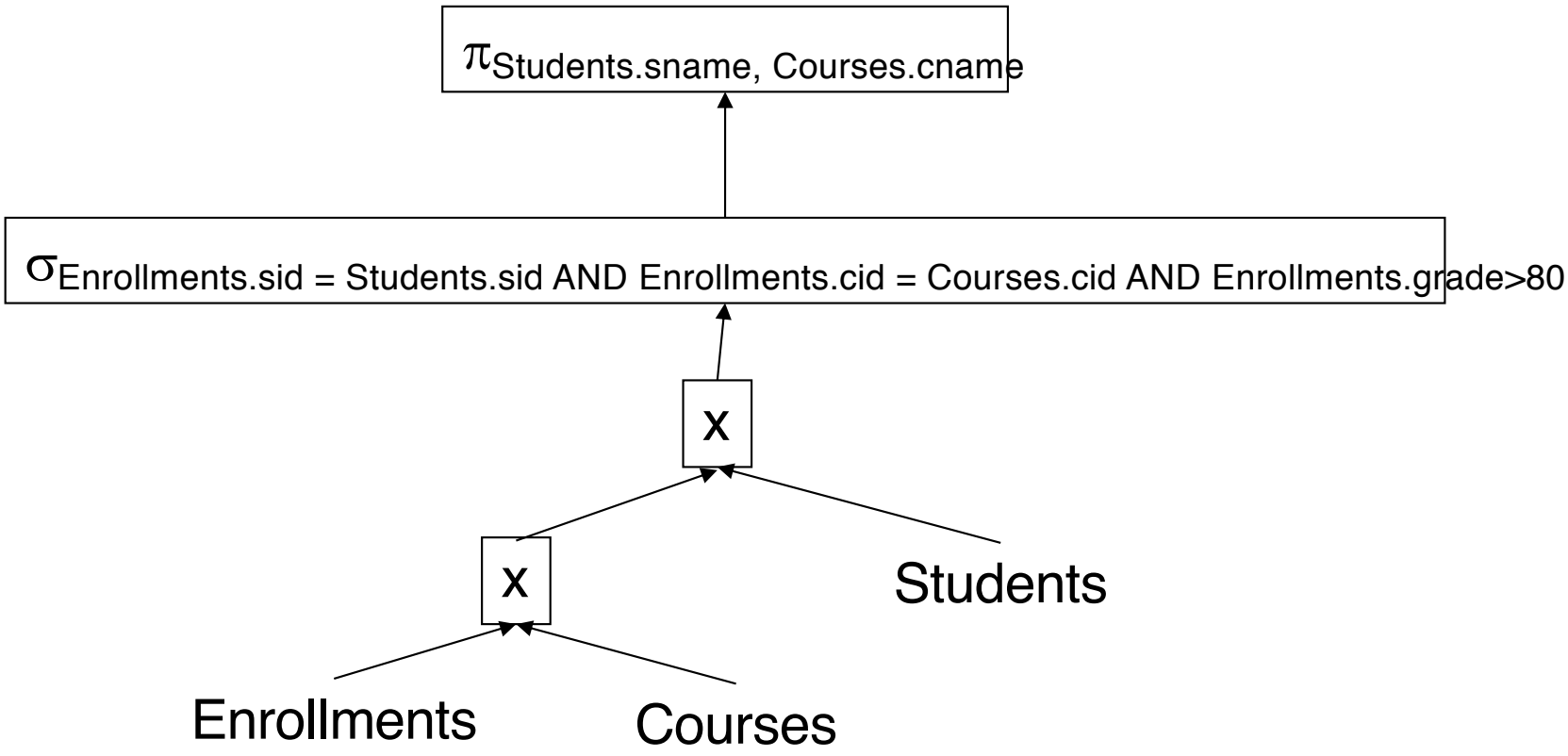
($\sigma_{\text{Enrollments.ecid = Courses.cid}}$ (Enrollments x Courses x Students))

AND Enrollments.esid = Students.sid

AND Enrollments.grade > 80

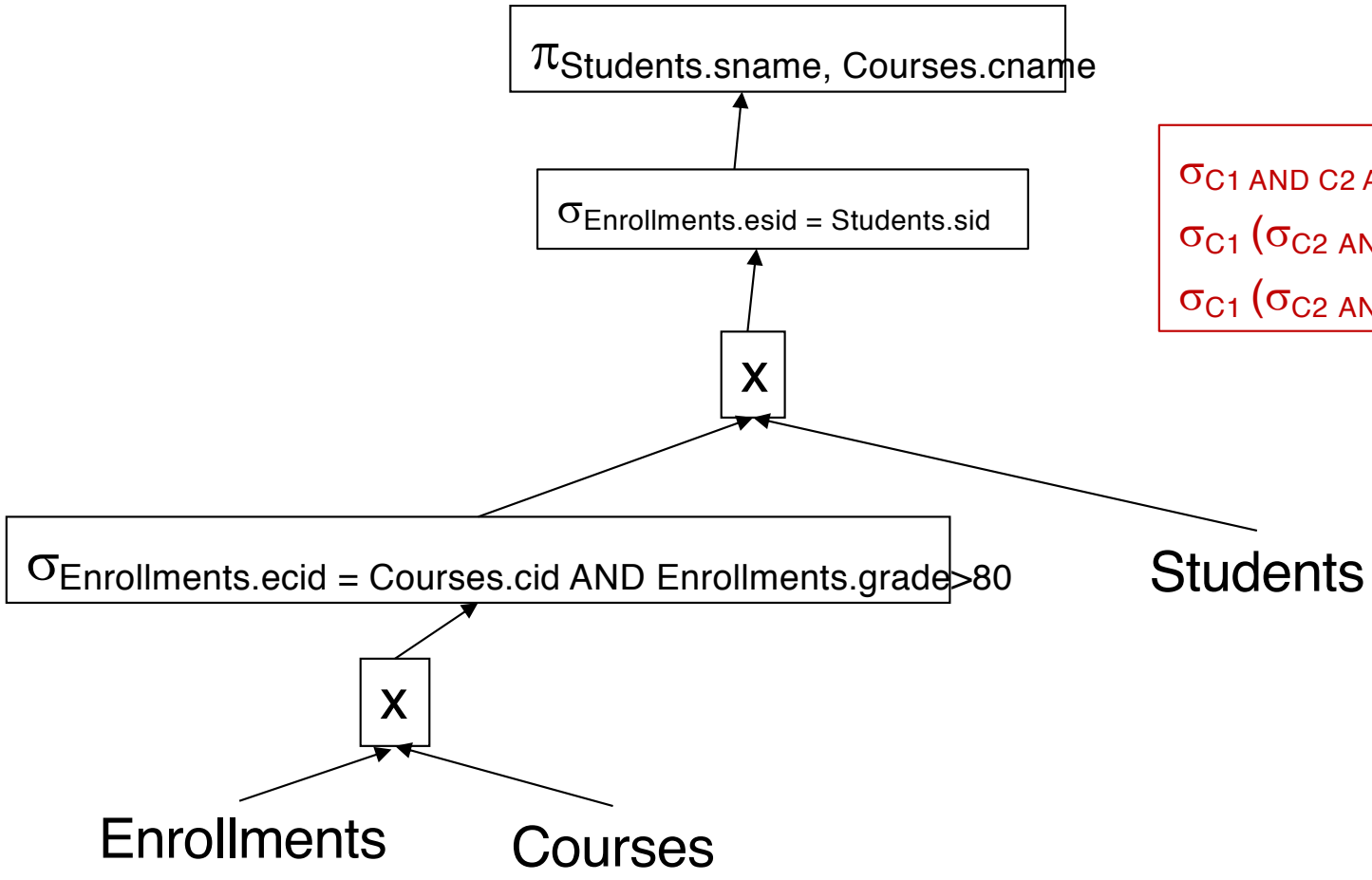
An Execution Plan for Query 2

- Find the student name and course name where the student had a grade more than 80 points in a course.



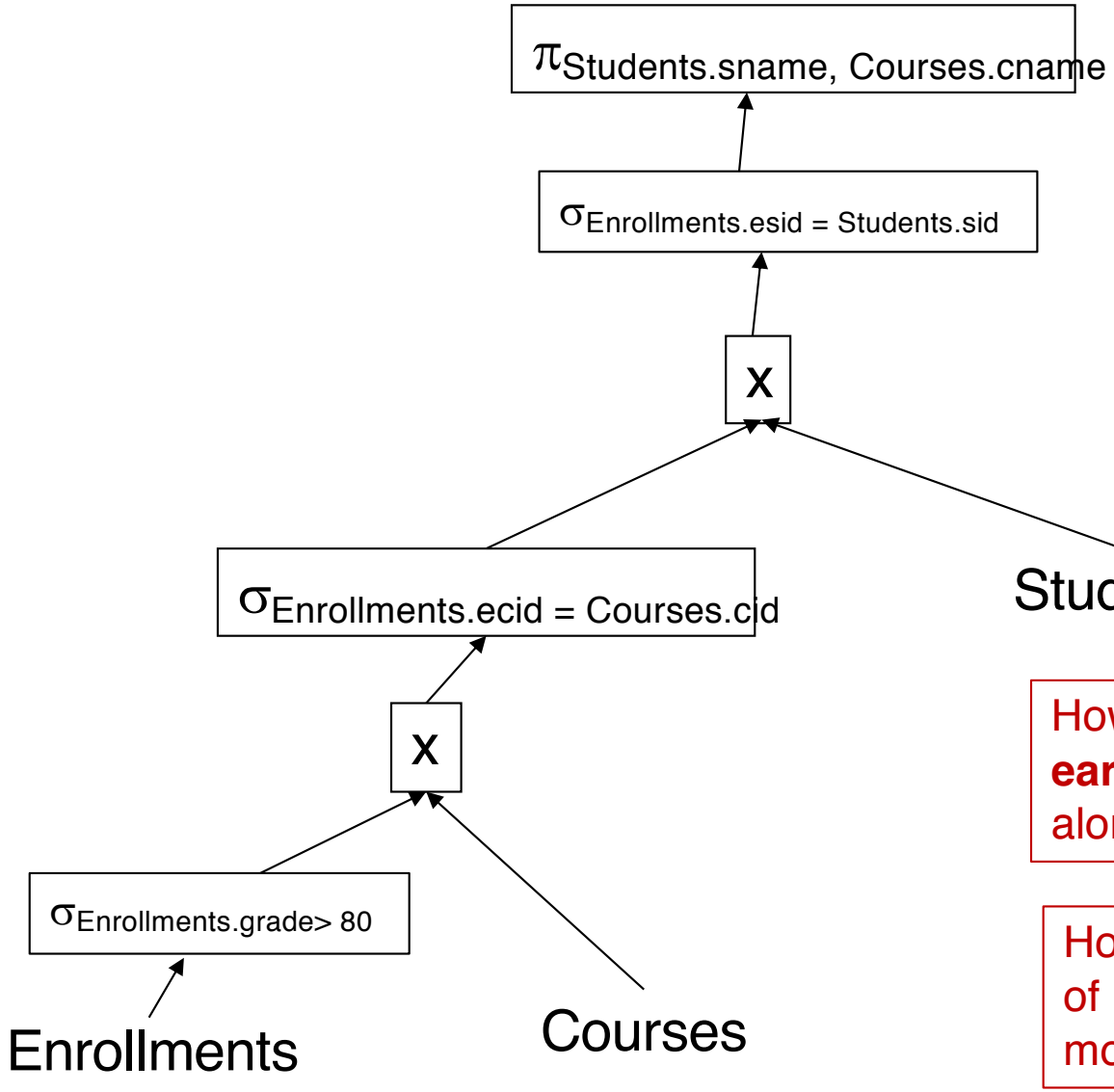
Another Execution Plan for Query 2

- Find the student name and course name where the student had a grade more than 80 points in a course.



$\sigma_{C1 \text{ AND } C2 \text{ AND } C3}$ (E x C x S) =
σ_{C1} ($\sigma_{C2 \text{ AND } C3}$ (E x C x S)) =
σ_{C1} ($\sigma_{C2 \text{ AND } C3}$ (E x C) x S)

A Third Execution Plan for Query 2



How could we do projections **earlier** in plan to avoid carrying along unnecessary attributes?

How could we do **Joins**, instead of Products to make plan more efficient?

Viewing Query Evaluation Plans

- Most database support **EXPLAIN** <query>
  - Displays plan chosen by query optimizer, along with cost estimates
  - Some syntax variations between databases
    - Oracle: **EXPLAIN PLAN FOR** <query>, which stores the plan in a table that can be queried.
    - SQL Server: **SET SHOWPLAN_TEXT ON** before executing query
- Some databases (e.g. PostgreSQL) support both **EXPLAIN** <query> (for plan) and **EXPLAIN ANALYSE** <query>
  - **EXPLAIN ANALYSE** shows the actual runtime statistics found when running the query, in addition to showing the plan .
- Some databases (e.g. PostgreSQL) show both the estimated cost to provide the first tuple in the result and the estimated cost to provide the entire result.