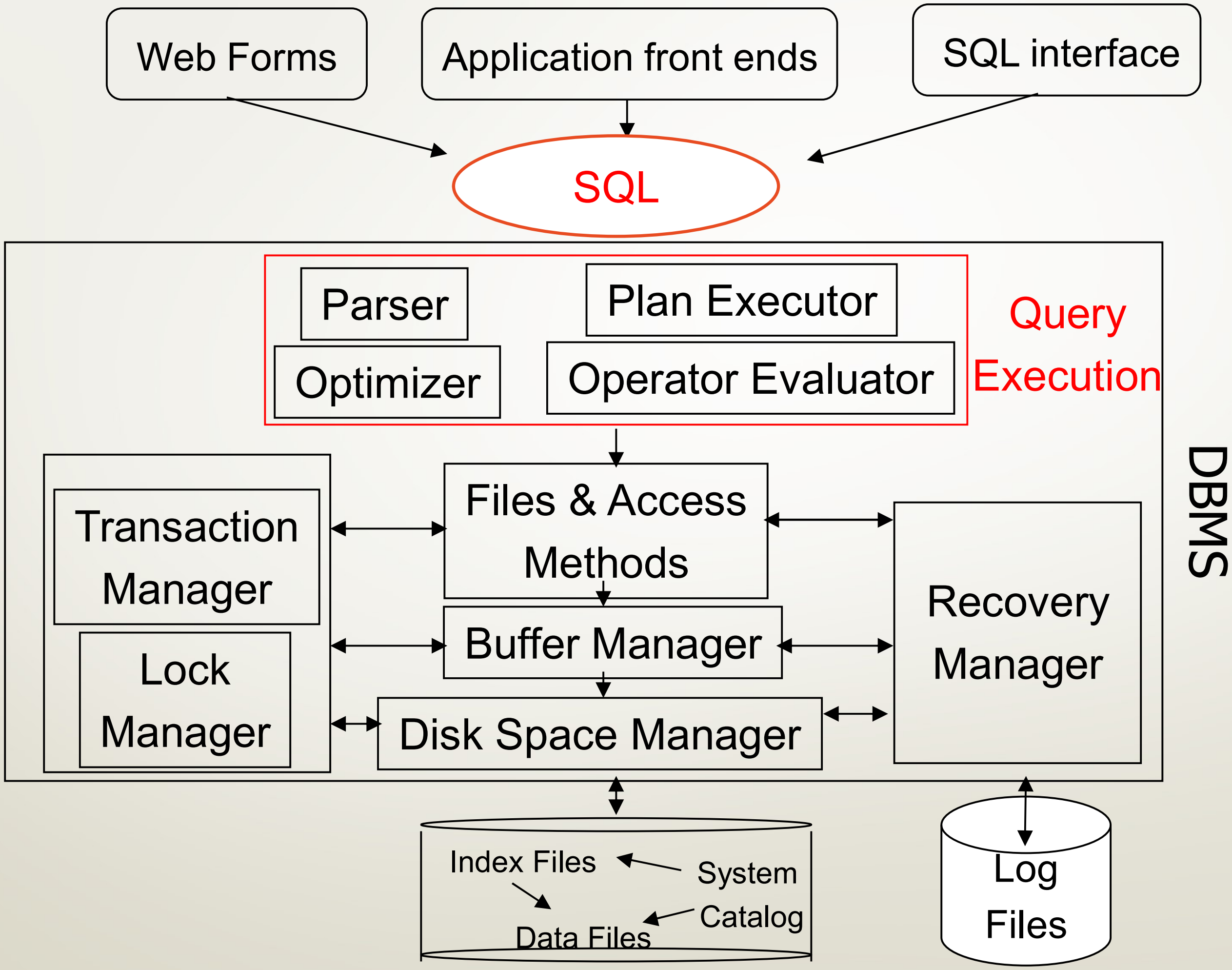
Relational Algebra

Abdu Alawini
University of Illinois at Urbana-Champaign
CS411: Database Systems
July 25, 2020

Relational Database Ecosystem



Why study the relational algebra?

- SQL is complicated!
- We need a simpler language of mathematical operations whose properties (e.g. commutativity and associativity) allow us to explore a space of equivalent expressions and find the one which is the “fastest” to execute
  - different ways of executing operations using different access methods, e.g. indexing
  - using statistics gathered about the size of relations, the distribution of attribute values, etc.

Recall the Relational Data Model

It all began with a breakthrough paper, by E.F. Codd in 1970:
"A relational model of data for large shared data banks".
Communications of the ACM **13** (6): 377

Codd's insights:

- Separate physical implementation from logical
- Model the data independently from how it will be used (accessed, printed, etc.)
  - Describe the data minimally and mathematically
    - A relation describes an association between data items – tuples with attributes
  - Use standard mathematical (logical) operations over the data – these are the relational algebra or relational calculus

Codd's Relational Algebra

- An algebra whose:
  - Operands are relations or variables that represent relations.
  - Operators are designed to do common things that we need to do with relations in a database.
- Relational algebra operations operate on relations and produce relations
  - Unary: f: Relation → Relation
  - Binary: g: Relation x Relation → Relation
- **The result is an algebra that can be used as a *query language* for relations.**

Codd's Logical Operations: The Relational Algebra

- Six basic operations:
  - Projection π_{α} (R)
  - Selection σ_{θ} (R)
  - (Rename) ρ_{α} (R)
  - Union $R_1 \cup R_2$
  - Difference $R_1 - R_2$
  - Product $R_1 \times R_2$
- And some other useful ones:
  - Join $R_1 \bowtie_{\theta} R_2$
  - Intersection $R_1 \cap R_2$

Example Relational Instance (again)

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

Takes

sid	exp-grade	cid	sem
1	A	550-001	F14
1	C	502-001	F14
3	A	555-001	S15
3	B	550-001	F14

COURSE

cid	subj	sem
550-001	DB	F14
502-001	Algo	F14
555-001	I&W	S15
666-001	Ethics	S66

PROFESSOR

fid	name
13	Ives
09	Guha
34	Tannen
66	Faust

Teaches

fid	cid	sem
34	550-001	F14
09	502-001	F14
13	555-001	S15

Projection, Π_α

- Given a list of column names α and a relation R, π_α (R) extracts the columns in α from the relation. Example:

Takes

sid	exp-grade	cid	sem
1	A	550-001	F14
1	C	502-001	F14
3	A	555-001	S15
3	B	550-001	F14

$\Pi_{\text{sid,exp-grade}}$ Takes

sid	exp-grade
1	A
1	C
3	A
3	B

Note: duplicate elimination.

What does this operator correspond to in SQL? SELECT.

SELECT DISTINCT sid, exp-grade
FROM Takes

Selection, σ_θ

Always applied to a single relation – a unary operator

$\sigma_{\text{exp-grade="A"}}$ Takes

the select operator

a relation name
(or a relational expression)

the predicate:

- a comparator ($\geq$, $>$, $\neq$, $=$, $\leq$, $<$)
- an attribute or a constant
- AND, OR, NOT

Example

- Selection σ_θ R takes a relation R and extracts those rows from it that satisfy the condition θ.

Example:

Takes

sid	exp-grade	cid	sem
1	A	550-001	F14
1	C	502-001	F14
3	A	555-001	S15
3	B	550-001	F14

$\sigma_{\text{exp-grade=“A”} \wedge \text{sid=1}}$ Takes

sid	exp-grade	cid	sem
1	A	550-001	F14

Can the result have duplicates? No.

What does this operator correspond to in SQL? WHERE

*SELECT DISTINCT **

FROM Takes

WHERE exp-grade='A' AND sid=1

Renaming, ρ_α (R)

- Does not change the relational instance
- Changes the relational schema only
- Notation: $\rho_{S(B1,\ldots,Bn)}(R)$
- Input schema: $R(A1, \ldots, An)$
- Output schema: $S(B1, \ldots, Bn)$
- Example: rename Student(sid, name)

$$\rho_{RenamedStudent(UIN,\ lastname)}(Student)$$

Cartesian Product X

- "Join" is a generic term for a variety of operations that connect two relations. The basic operation is the product, RxS, which concatenates every tuple in R with every tuple in S. Example:

STUDENT

sid	name
1	Jill
2	Bo

SCHOOL

school
UPenn
Cornell

STUDENT x SCHOOL

sid	name	school
1	Jill	UPenn
2	Bo	UPenn
1	Jill	Cornell
2	Bo	Cornell

What does this operator correspond to in SQL?

*SELECT DISTINCT **

FROM STUDENT, SCHOOL

Join, $\bowtie_\theta$: A Combination of Product and Selection

- Products are hardly ever used alone; they are typically used in conjunction with a selection. Example:

$$\sigma_{STUDENT.sid=Takes.sid}(STUDENT \times Takes) = STUDENT \bowtie_{STUDENT.sid=Takes.sid} Takes$$

```
SELECT DISTINCT *
FROM Student s, Takes t
WHERE s.sid=t.sid
```

```
SELECT DISTINCT *
FROM Student s JOIN Takes t
On s.sid=t.sid
```

sid:1	name	sid:2	exp-grade	cid	sem
1	Jill	1	A	550-001	F14
1	Jill	1	C	502-001	F14
3	Maya	3	A	555-001	S15
3	Maya	3	B	550-001	F14

“Natural” Join, $\bowtie$

- The most common join to do is an equality join of two relations on commonly named fields, and to leave one copy of those fields in the resulting relation. Example:

STUDENT $\bowtie$ Takes =

$$\rho_{sid:1 \rightarrow sid}(\Pi_{\text{sid:1,name, exp-grade, cid}}(\text{STUDENT} \bowtie_{\text{STUDENT.sid=Takes.sid}} \text{Takes}))$$

sid	name	exp-grade	cid
1	Jill	A	550-0103
1	Jill	A	700-1003
3	Nitin	A	700-1003
3	Nitin	C	500-0103
4	Marty	C	500-0103

Union ∪

- If two relations have the same structure (DB terminology: are union-compatible. Programming language terminology: have the same type) we can perform set operations.

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

POSTDOC

sid	name
1	Jill
12	Susan
18	Roger

STUDENT ∪ POSTDOC

sid	name
1	Jill
2	Bo
3	Maya
12	Susan
18	Roger

```
(SELECT sid, name
 FROM STUDENT)
UNION
(SELECT sid, name
 FROM POSTDOC)
```

Difference –

- Another set operator. Example:

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

POSTDOC

sid	name
1	Jill
12	Susan
18	Roger

STUDENT – POSTDOC

sid	name
2	Bo
3	Maya

```
(SELECT sid, name
 FROM STUDENT)
EXCEPT/MINUS
(SELECT sid, name
 FROM POSTDOC)
```

Example Relational Instance (again)

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

Takes

sid	exp-grade	cid	sem
1	A	550-001	F14
1	C	502-001	F14
3	A	555-001	S15
3	B	550-001	F14

COURSE

cid	subj	sem
550-001	DB	F14
502-001	Algo	F14
555-001	I&W	S15
666-001	Ethics	S66

PROFESSOR

fid	name
13	Ives
09	Guha
34	Tannen
66	Faust

Teaches

fid	cid	sem
34	550-001	F14
09	502-001	F14
13	555-001	S15

Exercise

Try writing queries for these:

- The ids of students named “Bo”
- The names of students expecting an “A”
- The names of students in Ives’s classes
- The sids and names of students not enrolled

Building Complex Expressions

- Algebras allow us to express sequences of operations in a natural way.
- Example
  - in arithmetic algebra: $(x + 4)*(y - 3)$
- Relational algebra allows the same.
- Three notations:
  1. Sequences of assignment statements.
  2. Expressions with several operators.
  3. Expression trees.

1. Sequences of Assignments

- Create temporary relation names.
- Renaming can be implied by giving relations a list of attributes.
  - R3(X, Y) := R1
- Example: R3 := R1 $\bowtie_c$ R2 can be written:

  R4 := R1 x R2

  R3 := σ_c (R4)

Q: How would we do this?

- Using Sells(cafe, drink, price), find the cafes that sell two different drinks at the same price.

- $R1 := \rho_{Sells1(cafe,\ drink1,\ price)}\ Sells$
- $R2 := \sigma_{NOT\ (drink1\ =\ drink)}\ (R1 \bowtie Sells)$
- $R3 := \Pi_{cafe}\ R2$

2. Expressions with Several Operators

Precedence of relational operators:

1. Unary operators --- select, project, rename --- have highest precedence, bind first.
2. Then come products and joins.
3. Then intersection.
4. Finally, union and set difference bind last.

But you can always insert parentheses to force the order you desire.

3. Expression Trees

- Leaves are operands (relations).
- Interior nodes are operators, applied to their child or children.

Q: How would we do this?

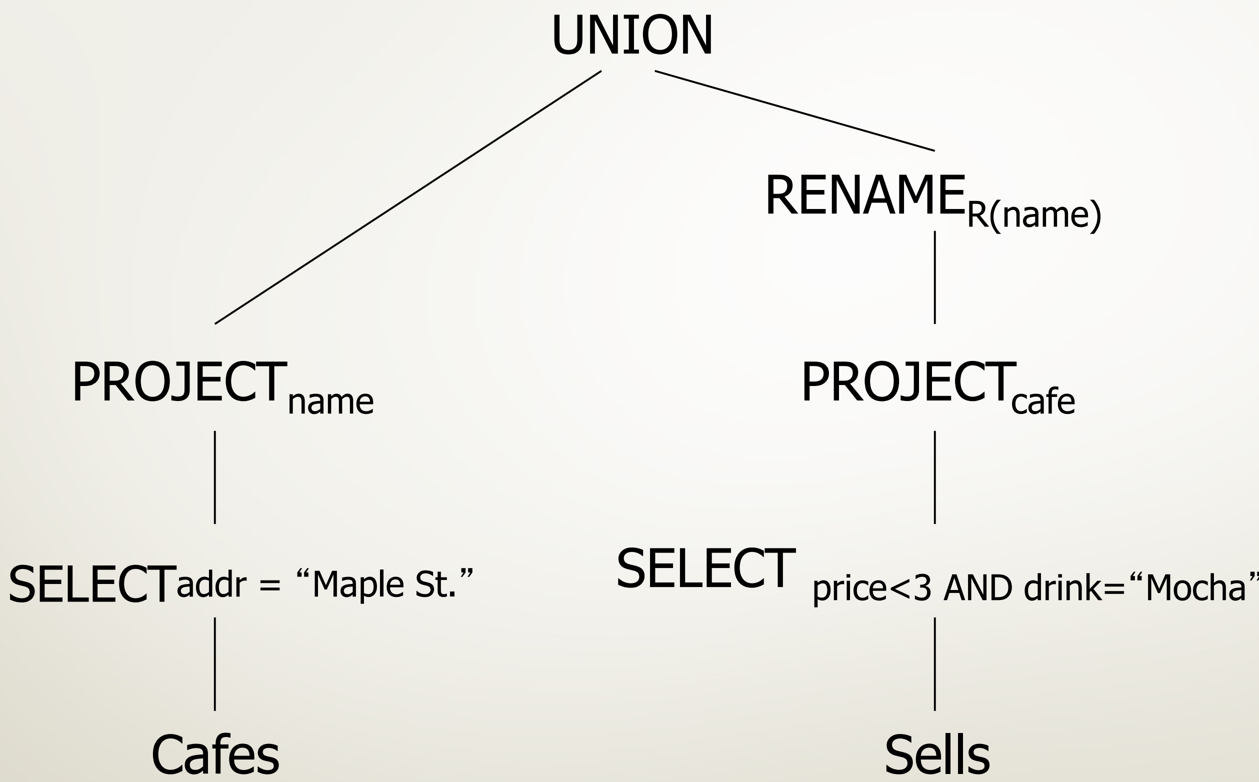
- Using Sells(cafe, drink, price), find the cafes that sell two different drinks at the same price.

- R1:= $\rho_{\text{Sells1(cafe, drink1, price)}}$ Sells
- R2:= $\sigma_{\text{NOT (drink1 = drink)}}$ R1 $\bowtie$ Sells
- R3:= Π_{cafe} R2

Example

- Given Cafes(name, addr), Sells(cafe, drink, price), find the names of all the cafes that are either on Maple St. or sell Mocha for less than $3.

As a Tree:


UNION
RENAME R(name)
PROJECT name
PROJECT cafe
SELECT addr = “Maple St.”
SELECT price<3 AND drink=“Mocha”
Cafes
Sells

More Queries

Product (pid, name, price, category, maker-cid)

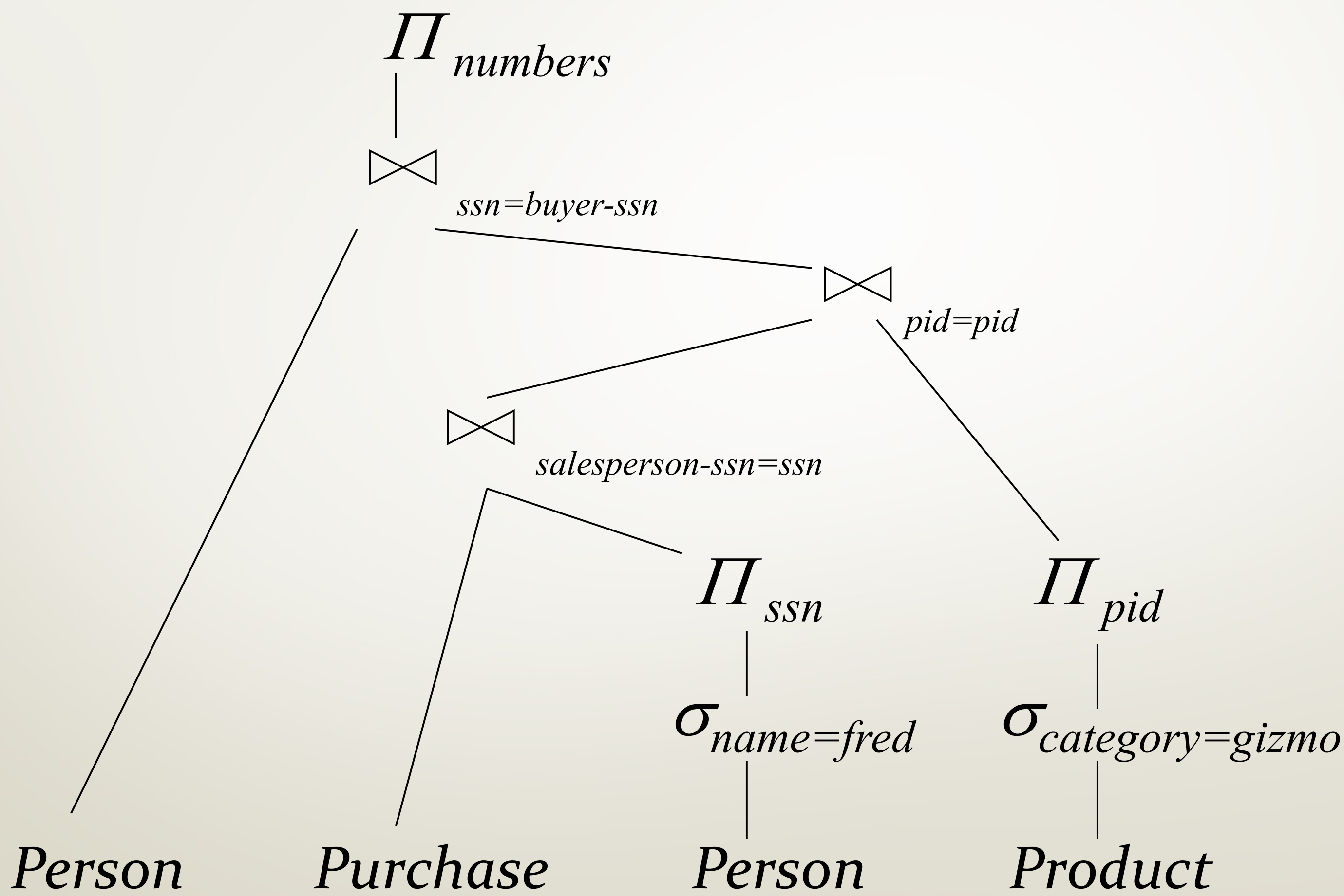
Purchase (buyer-ssn, salesperson-ssn, store, pid)

Company (cid, name, stock price, country)

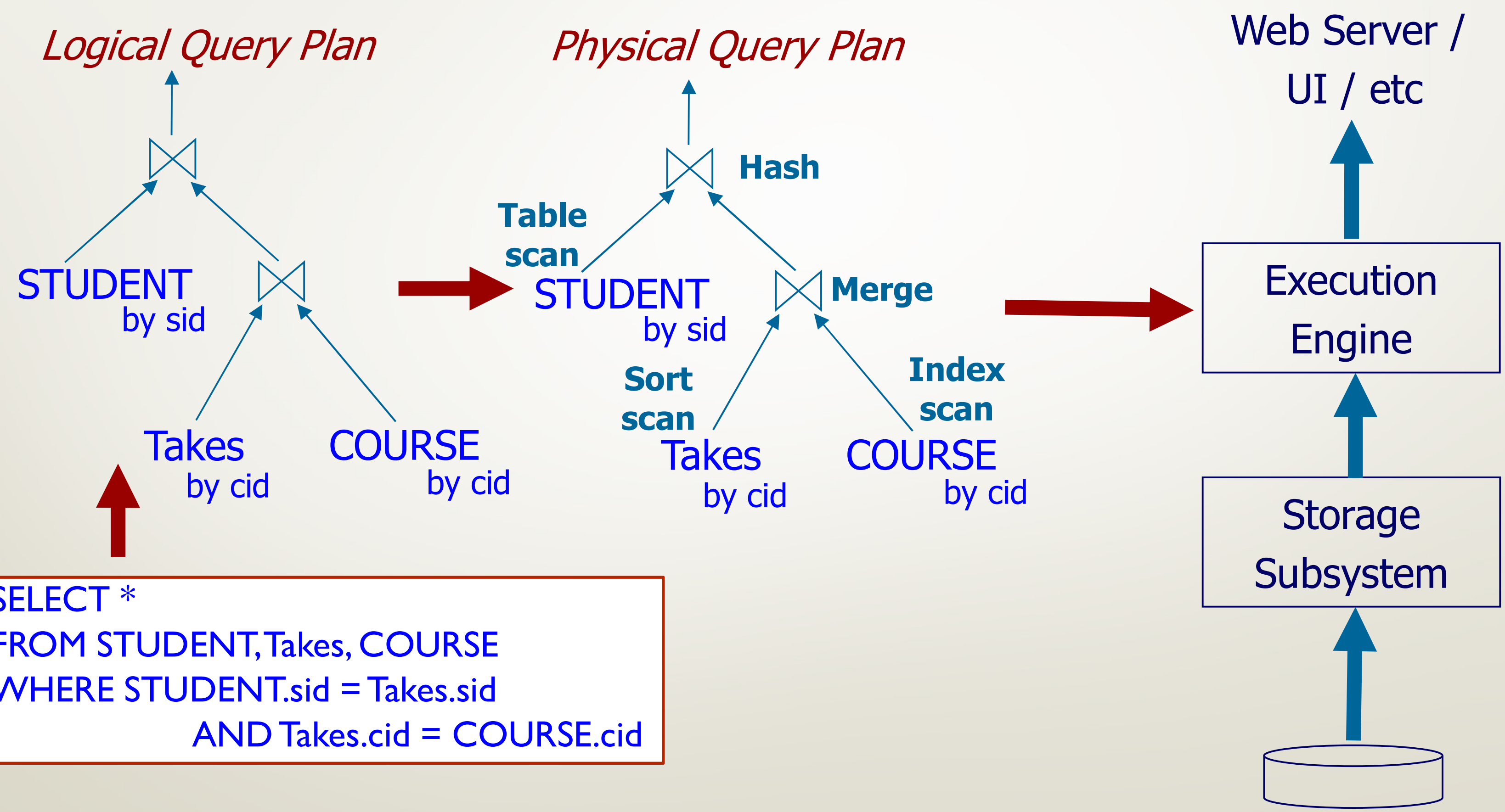
Person (ssn, name, phone number, city)

Find phone numbers of people who bought gizmos from Fred.

Expression Tree



QP: The Big Picture: SQL → Logical Query Plan → Physical Query Plan



SELECT *
FROM STUDENT, Takes, COURSE
WHERE STUDENT.sid = Takes.sid
AND Takes.cid = COURSE.cid

Summary

- The relational algebra is a set of mathematical operators that compose, modify, and combine tuples within different relations
  - Basic SQL expressions can be expressed in this form
- Relational algebra has laws of commutativity, associativity, etc. that imply certain expressions are equivalent in semantics
- This is used in query optimization to find the most efficient representation to evaluate (or one that's not bad)