CS 6400 A

Database Systems Concepts and Design

Lecture 12

10/02/24

Announcements

Midterm stats:

- max: 89
- median: 73.5
- mean: 71.55
- std: 10.94

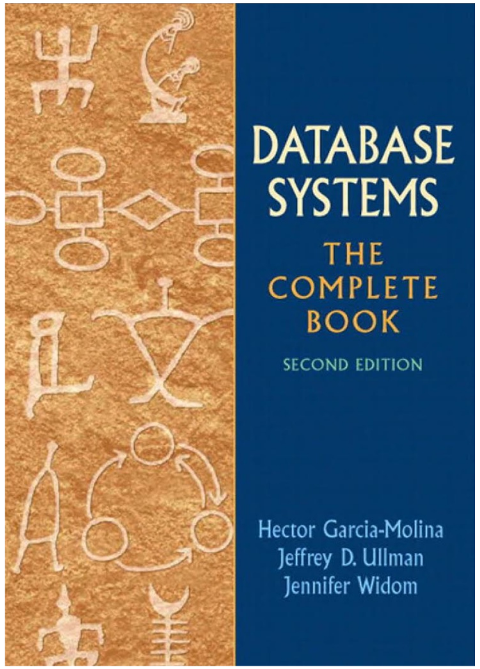
Answer posted on canvas

- Files->Midterm solution

Reading Materials

Database Systems: The Complete Book (2nd edition)

- Chapter 2.4: An Algebraic Query Language



Acknowledgement: The following slides have been adapted from CS145 (Intro to Big Data Systems) taught by Peter Bailis.

Agenda

Recap: The Relational Model

1. Relational Algebra: Basic Operators

2. Relational Algebra Pt. II

3. Logical Optimization

The Relational Model: Schemata

- Relational Schema:

```
Students(sid: string, name: string, gpa: float)
```

Relation name

String, float, int, etc. are the <u>domains</u> of the attributes

<u>Attributes</u>

The Relational Model: Data

An attribute (or column) is a typed data entry present in each tuple in the relation

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

The number of attributes is the arity of the relation

The Relational Model: Data

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

The number of tuples is the cardinality of the relation

A tuple or row (or record) is a single entry in the table having the attributes specified by the schema

The Relational Model: Data

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

Recall: In practice DBMSs relax the set requirement, and use multisets.

A <u>relational instance</u> is a set of tuples all conforming to the same schema

To Reiterate

- A *relational schema* describes the data that is contained in a *relational instance*

Let $R(f_1{:}Dom_1,\ldots,f_m{:}Dom_m)$ be a relational schema then, an instance of R is a subset of $Dom_1 \times Dom_2 \times \ldots \times Dom_n$

In this way, a relational schema R is a total function from attribute names to types

A relational database

- A *relational database schema* is a set of relational schemata, one for each relation

- A *relational database instance* is a set of relational instances, one for each relation

Two conventions:

1. We call relational database instances as simply **databases**
2. We assume all instances are valid, i.e., satisfy the domain constraints

2nd Part of the Model: Querying

```
SELECT S.name
FROM Students S
WHERE S.gpa > 3.5;
```

“Find names of all students with GPA > 3.5”

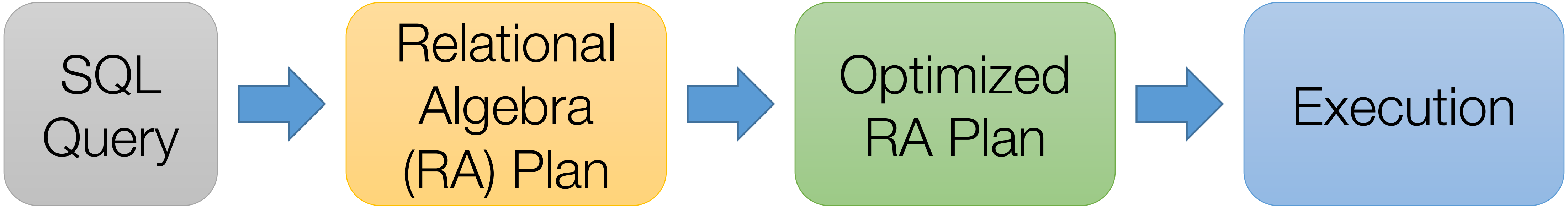
We don’t tell the system how or where to get the data- just what we want, i.e., Querying is declarative

To make this happen, we need to translate the declarative query into a series of operators… we’ll see this next!

1. Relational Algebra

RDBMS Architecture

How does a SQL engine work ?



Declarative query (from user)

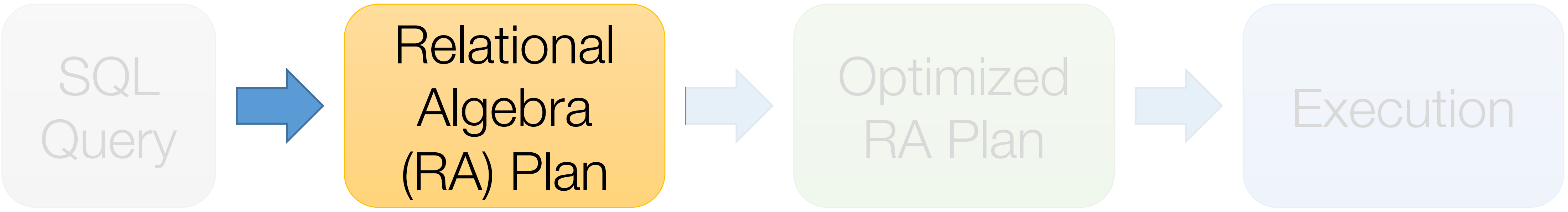
Translate to relational algebra expression

Find logically equivalent- but more efficient- RA expression

Execute each operator of the optimized plan!

RDBMS Architecture

How does a SQL engine work ?



Relational Algebra allows us to translate declarative (SQL) queries into precise and optimizable expressions!

Relational Algebra (RA)

- Five **basic** operators:
  1. Selection: σ
  2. Projection: Π
  3. Cartesian Product: $\times$
  4. Union: $\cup$
  5. Difference: -

We'll look at these first!

- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural, equi-join, theta join, semi-join)
  - Renaming: ρ
  - Division

And also at one example of a derived operator (natural join) and a special operator (renaming)

Note: RA operates on sets!

- RDBMSs use *multisets*, however in relational algebra formalism we will consider **sets!**

- Also: we will consider the ***named perspective***, where every attribute must have a unique name
  - →attribute order does not matter…

Now on to the basic RA operators…

1. Selection (σ)

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{\text{Salary} > 40000}$ (Employee)
  - $\sigma_{\text{name} = \text{“Smith”}}$ (Employee)
- The condition c can be =, <, ≤, >, ≥, <>

```
Students(sid,sname,gpa)
```

SQL:

```
SELECT *
FROM Students
WHERE gpa > 3.5;
```

RA:

$$\sigma_{gpa > 3.5}(Students)$$

Another example:

SSN	Name	Salary
1234545	John	200000
5423341	Smith	600000
4352342	Fred	500000

$\sigma_{\text{Salary} > 40000}$ (Employee)

SSN	Name	Salary
5423341	Smith	600000
4352342	Fred	500000

2. Projection (Π)

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A1,\ldots,An}(R)$
- Example: project social-security number and names:
  - $\Pi_{SSN,\ Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

```
Students(sid,sname,gpa)
```

SQL:

```
SELECT DISTINCT
   sname,
   gpa
FROM Students;
```

RA:

$$\Pi_{sname,gpa}(Students)$$

Another example:

SSN	Name	Salary
1234545	John	200000
5423341	John	600000
4352342	John	200000

$\Pi_{\text{Name,Salary}}$ (Employee)

Name	Salary
John	200000
John	600000

Note that RA Operators are Compositional!

```
Students(sid,sname,gpa)
```

```sql
SELECT DISTINCT
   sname,
   gpa
FROM Students
WHERE gpa > 3.5;
```

$$\Pi_{sname,gpa}(\sigma_{gpa>3.5}(Students))$$

$$\sigma_{gpa>3.5}(\Pi_{sname,gpa}(Students))$$

How do we represent this query in RA?

Are these logically equivalent?

3. Cross-Product (×)

- Each tuple in R1 with each tuple in R2
- Notation: R1 × R2
- Example:
  - Employee × Dependents
- Rare in practice; mainly used to express joins

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

```
SELECT *
FROM Students, People;
```

RA:

$$Students \times People$$

Another example:

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

$\times$

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$$Students \times People$$

ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
5423341	Bob	217 Rosse	001	John	3.4
1234545	John	216 Rosse	002	Bob	1.3
5423341	Bob	216 Rosse	002	Bob	1.3

Renaming (ρ)

- Changes the schema, not the instance
- A 'special' operator- neither basic nor derived
- Notation: $\rho_{B1,\ldots,Bn}(R)$

- **Note: this is shorthand for the proper form (since names, not order matters!):**
  - $\rho_{A1 \rightarrow B1,\ldots,An \rightarrow Bn}(R)$

```
Students(sid,sname,gpa)
```

SQL:

```
SELECT
   sid AS studId,
   sname AS name,
   gpa AS gradePtAvg
FROM Students;
```

RA:

$$\rho_{studId,name,gradePtAvg}(Students)$$

We care about this operator because we are working in a named perspective

Another example:

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$$\rho_{studId,name,gradePtAvg}(Students)$$

Students

studId	name	gradePtAvg
001	John	3.4
002	Bob	1.3

Natural Join (⋈)

- $R_1 \bowtie R_2$: Joins R_1 and R_2 on *equality of all shared attributes*
  - If R_1 has attribute set A, and R_2 has attribute set B, and they share attributes A ⋂ B = C, can also be written: $R_1 \bowtie_{C} R_2$

- Our first example of a *derived* RA operator:
  - $R_1 \bowtie R_2 = \Pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R_1) \times R_2))$
  - Where:
    - The rename $\rho_{C \to D}$ renames the shared attributes in one of the relations
    - The selection $\sigma_{C=D}$ checks equality of the shared attributes
    - The projection $\Pi_{A \cup B}$ eliminates the duplicate common attributes

```
Students(sid,name,gpa)
People(ssn,name,address)
```

SQL:

```
SELECT DISTINCT
  ssid, S.name, gpa,
  ssn, address
FROM
  Students S,
  People P
WHERE S.name = P.name;
```

RA:

$$Students \bowtie People$$

Another example:

Students S

sid	S.name	gpa
001	John	3.4
002	Bob	1.3

$\bowtie$

People P

ssn	P.name	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

$$Students \bowtie People$$

sid	S.name	gpa	ssn	address
001	John	3.4	1234545	216 Rosse
002	Bob	1.3	5423341	216 Rosse

Just to check your understanding

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

- Given R(A, B, C), S(D, E), what is R ⋈ S ?

- Given R(A, B), S(A, B), what is R ⋈ S ?

Example: Converting SFW Query -> RA

```
Students(sid,sname,gpa)
People(ssn,sname,address)
```

```
SELECT DISTINCT
   gpa,
   address
FROM Students S,
     People P
WHERE gpa > 3.5 AND
    sname = pname;
```

$$\Pi_{gpa,address}(\sigma_{gpa>3.5}(S \bowtie P))$$

How do we represent this query in RA?

2. Relational Algebra Pt. II

Relational Algebra (RA)

- Five **basic** operators:
    1. Selection: σ
    2. Projection: Π
    3. Cartesian Product: $\times$
    4. Union: $\cup$
    5. Difference: -

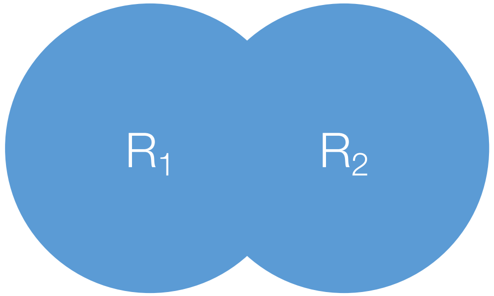
We'll look at these

- Derived or auxiliary operators:
    - Intersection, complement
    - Joins (natural, equi-join, theta join, semi-join)
    - Renaming: ρ
    - Division

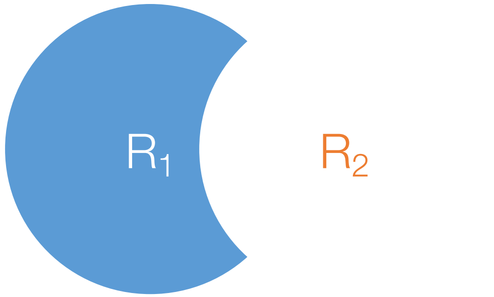
And also at some of these derived operators

1. Union (∪) and 2. Difference (–)

- R1 ∪ R2
- Example:
    - ActiveEmployees ∪ RetiredEmployees

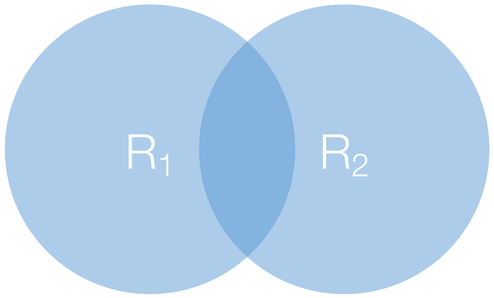


- R1 – R2
- Example:
    - AllEmployees -- RetiredEmployees



What about Intersection (∩) ?

- It is a derived operator
- $R1 \cap R2 = R1 - (R1 - R2)$
- Also expressed as a join!
- Example
  - UnionizedEmployees ∩ RetiredEmployees



Theta Join ($\bowtie_\theta$)

- A join that involves a predicate
- R1 $\bowtie_\theta$ R2 $= \sigma_\theta$ (R1 $\times$ R2)
- Here θ can be any condition

Note that natural join is a theta join + a projection.

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

```
SELECT *
FROM
  Students,People
WHERE θ;
```

RA:

$$Students \bowtie_\theta People$$

Equi-join ($\bowtie_{A=B}$)

- A theta join where θ is an equality
- $R1 \bowtie_{A=B} R2 \;=\; \sigma_{A=B}(R1 \times R2)$
- Example:
  - $\text{Employee} \bowtie_{SSN=SSN} \text{Dependents}$

Most common join in practice!

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

```
SELECT *
FROM
  Students S,
  People P
WHERE sname = pname;
```

RA:

$$S \bowtie_{sname=pname} P$$

Semijoin ($\ltimes$)

- $R \ltimes S = \Pi_{A1,\ldots,An} (R \bowtie S)$
- Where $A_1, \ldots, A_n$ are the attributes in R
- Example:
  - Employee $\ltimes$ Dependents

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

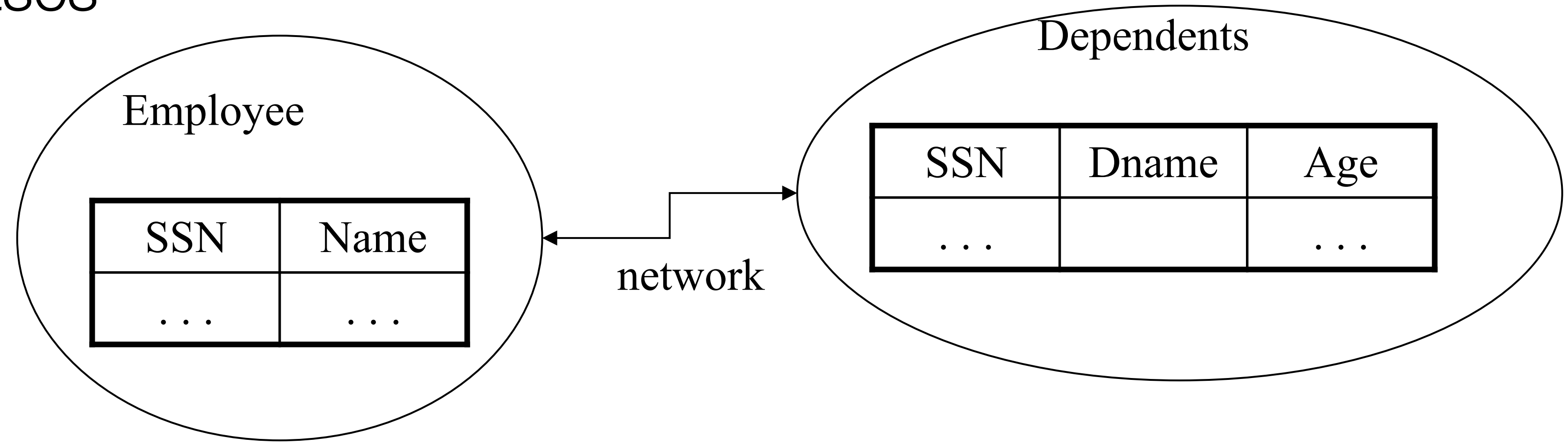
```
SELECT DISTINCT
  sid,sname,gpa
FROM
  Students,People
WHERE
  sname = pname;
```

RA:

$$Students \ltimes People$$

Semijoins in Distributed Databases

- Semijoins are often used to compute natural joins in distributed databases



$$\text{Employee} \bowtie_{ssn=ssn} (\sigma_{age>71}(\text{Dependents}))$$

Send less data to reduce network bandwidth!

$T = \Pi_{SSN}(\sigma_{age>71}(\text{Dependents}))$

$R = \text{Employee} \ltimes T$

$\text{Answer} = R \bowtie \text{Dependents}$

3. Logical Optimization

RDBMS Architecture

How does a SQL engine work ?

SQL Query → Relational Algebra (RA) Plan → Optimized RA Plan → Execution

We saw how we can transform declarative SQL queries into precise, compositional RA plans

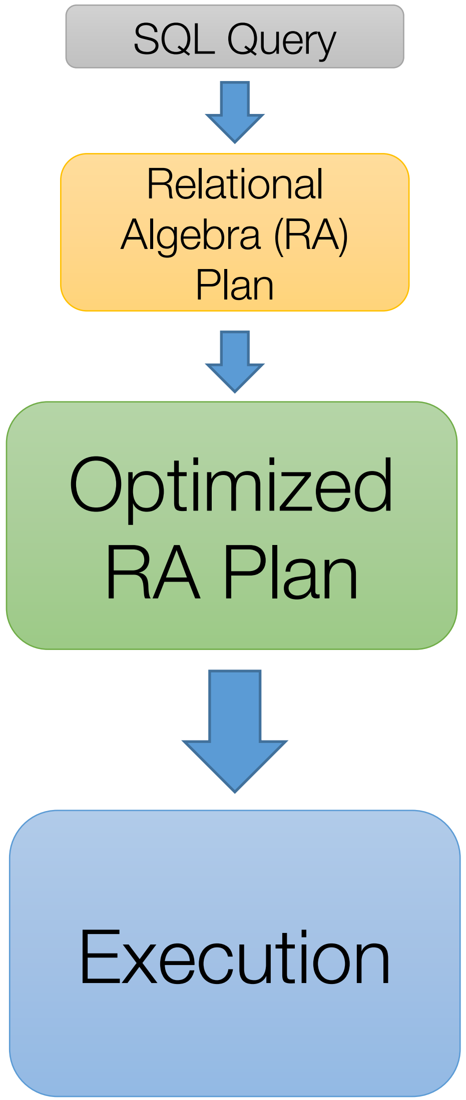
Logical vs. Physical Optimization

<u>Logical optimization:</u>

- Find equivalent plans that are more efficient
- *Intuition: Minimize # of tuples at each step by changing the order of RA operators*

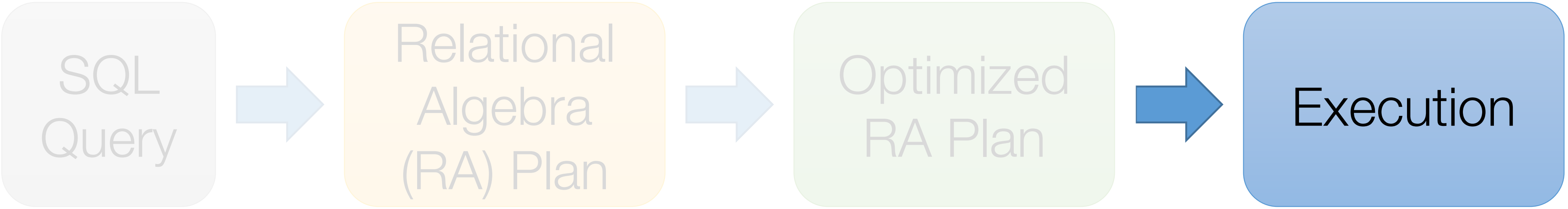
<u>Physical optimization:</u>

- Find algorithm with lowest IO cost to execute our plan
- *Intuition: Calculate based on physical parameters (buffer size, etc.) and estimates of data size (histograms)*



RDBMS Architecture

How is the RA “plan” executed?



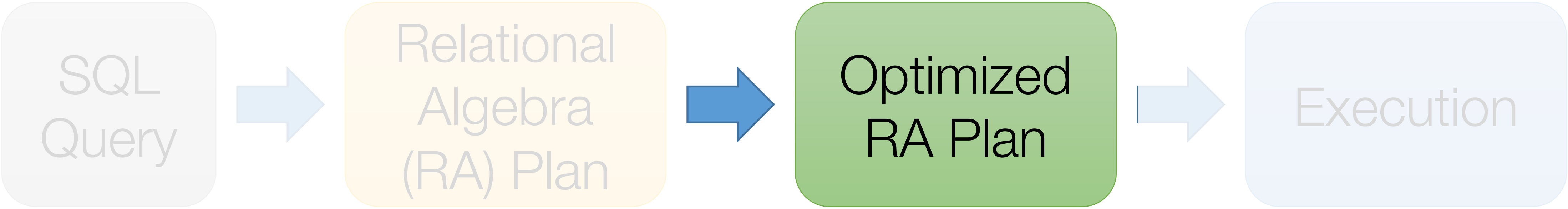
RA Plan Execution

- Natural Join / Join:
  - Next lecture: how to use **memory & IO cost** considerations to pick the correct algorithm to execute a join with (BNLJ, SMJ, HJ…)!
- Selection:
  - We saw how to use **indexes to aid selection**
  - Can always fall back on scan / binary search as well
- Projection:
  - The main operation here is finding *distinct* values of the project tuples; we briefly discussed how to do this with e.g. **hashing** or **sorting**

We already know how to execute all the basic operators!

RDBMS Architecture

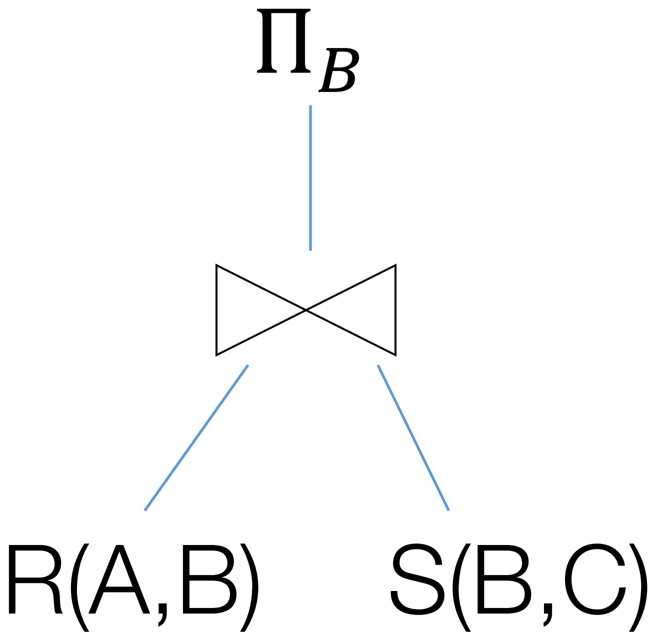
How does a SQL engine work ?



We'll look at how to then optimize these plans now

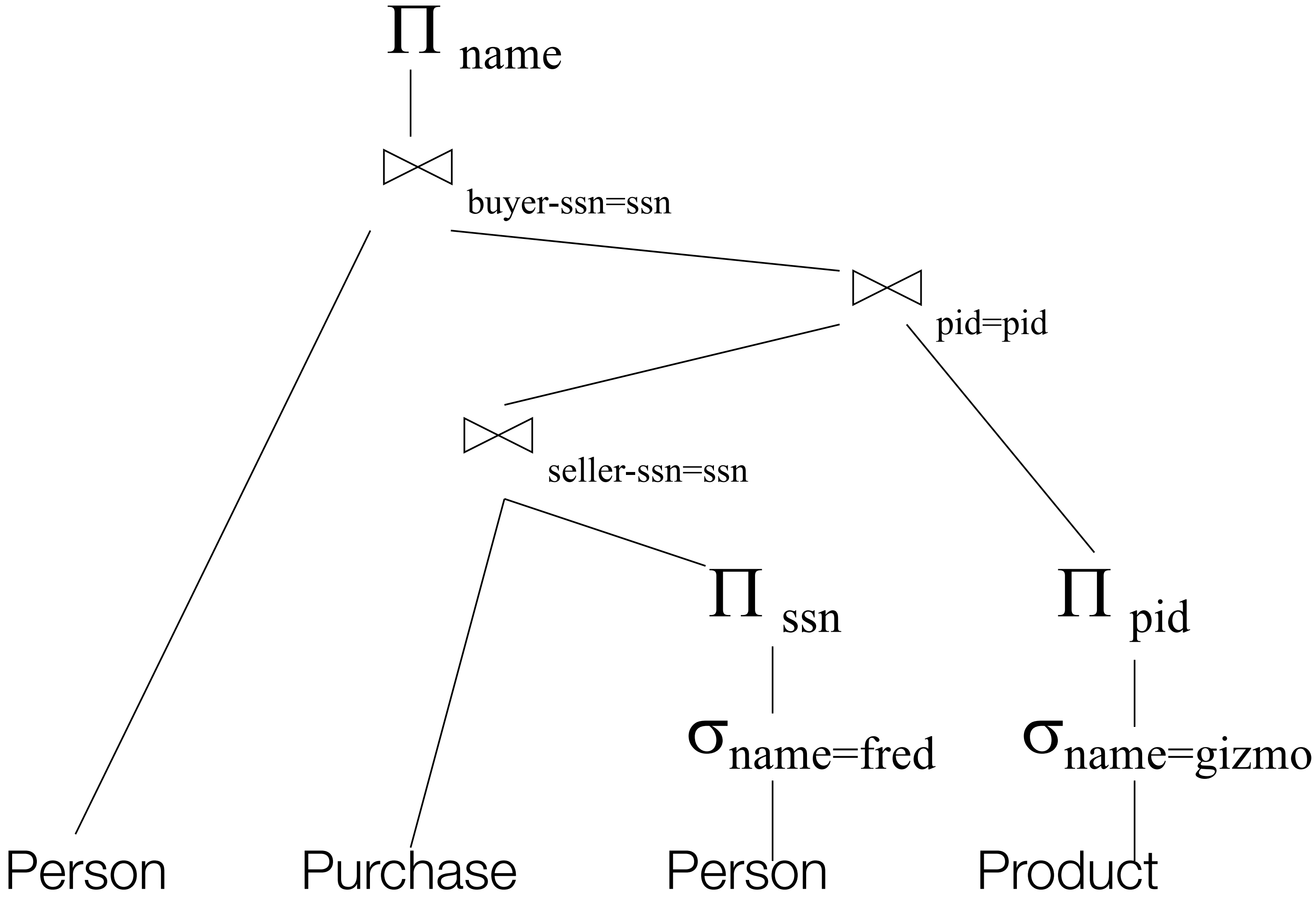
Note: We can visualize the plan as a tree

$$\Pi_B(R(A,B) \bowtie S(B,C))$$

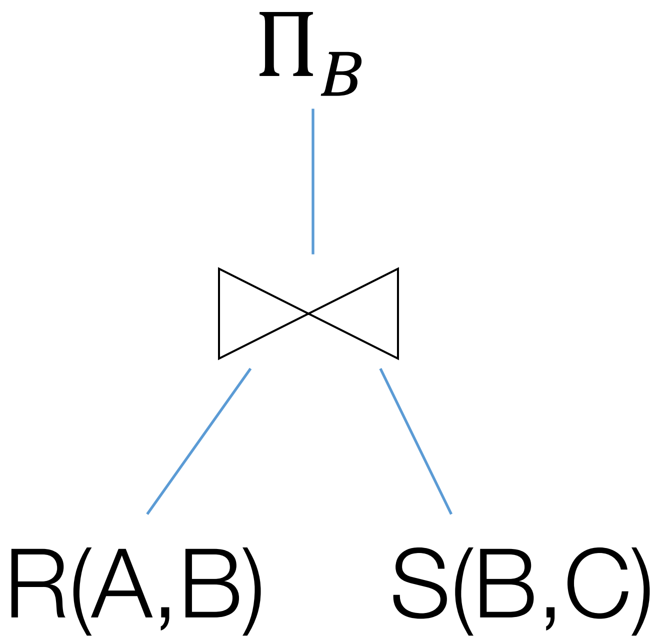


Bottom-up tree traversal = order of operation execution!

RA Expressions Can Get Complex!



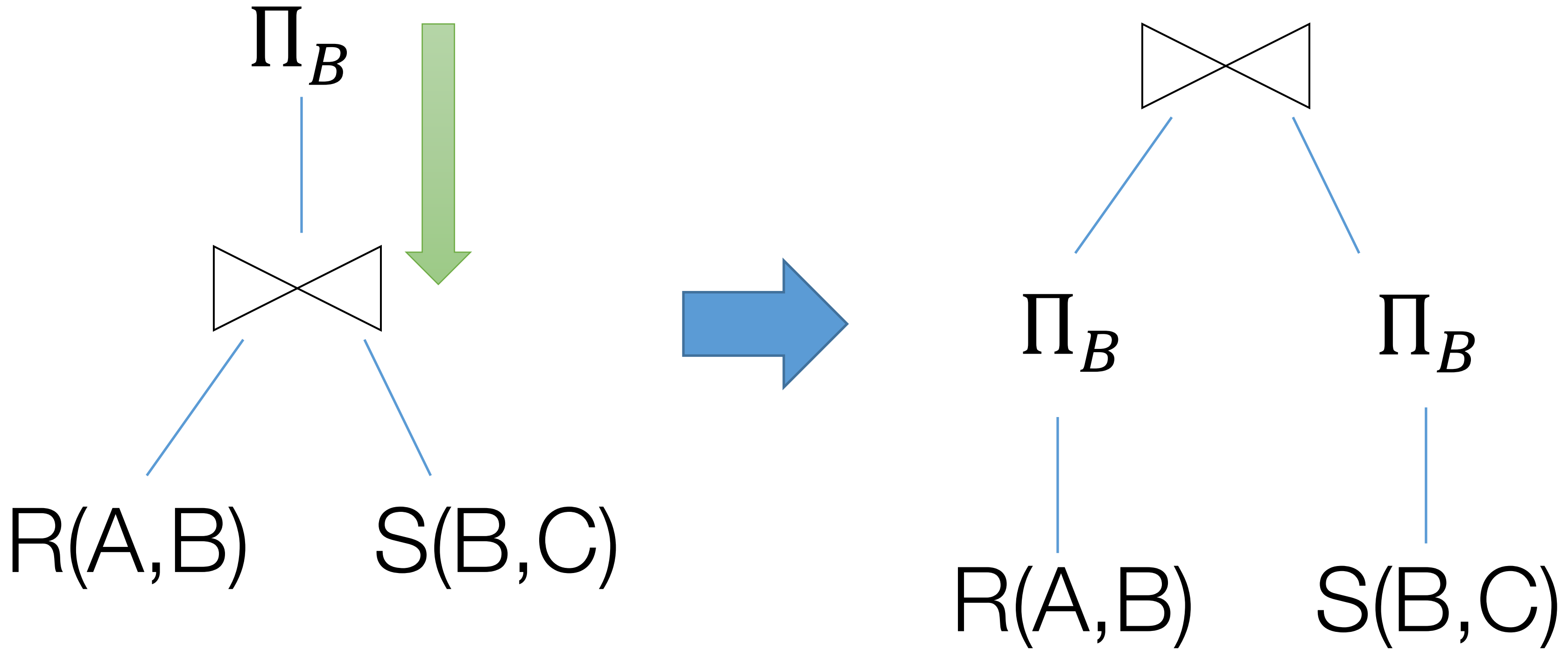
A simple plan



What SQL query does this correspond to?

Are there any logically equivalent RA expressions?

"Pushing down" projection



Why might we prefer this plan?

Takeaways

- This process is called logical optimization
- Many equivalent plans used to search for “good plans”
- Relational algebra is an important abstraction.

Commutative and associative laws

Example:

$$R \times S = S \times R$$
$$(R \times S) \times T = R \times (S \times T)$$

- Same holds for ⋈, ∪, ∩
- Holds for both set and bag semantics

Laws involving projection and selection

- The basic commutators:
  - Push projection through (1) selection, (2) join
  - Push selection through (3) selection, (4) projection, (5) join

- Note that this is not an exhaustive set of operations
  - This covers *local re-writes; global re-writes possible but much harder*
  - Additional reading: Chapter 16.2

This simple set of tools allows us to greatly improve the execution time of queries by optimizing RA plans!

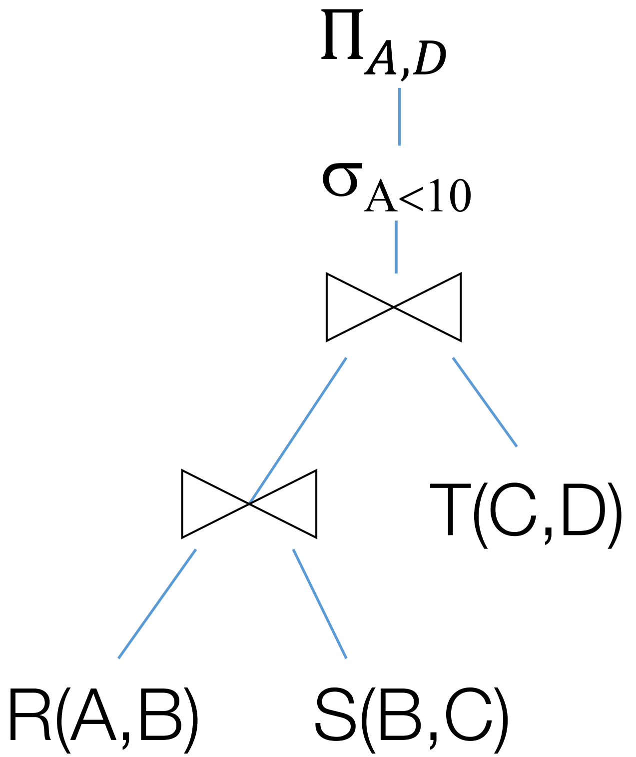
Optimizing the SFW RA Plan

Translating to RA

```
R(A,B)  S(B,C)  T(C,D)
```

```
SELECT R.A,S.D
FROM R,S,T
WHERE R.B = S.B
  AND S.C = T.C
  AND R.A < 10;
```

$$\Pi_{A,D}(\sigma_{A<10}(T \bowtie (R \bowtie S)))$$



Logical Optimization

- Heuristically, we want selections and projections to occur as early as possible in the plan
  - Terminology: “push down **selections**” and “pushing down **projections.”**

- **Intuition:** We will have fewer tuples in a plan.
  - Could fail if the selection condition is very expensive (say runs some image processing algorithm).
  - Projection could be a waste of effort, but more rarely.

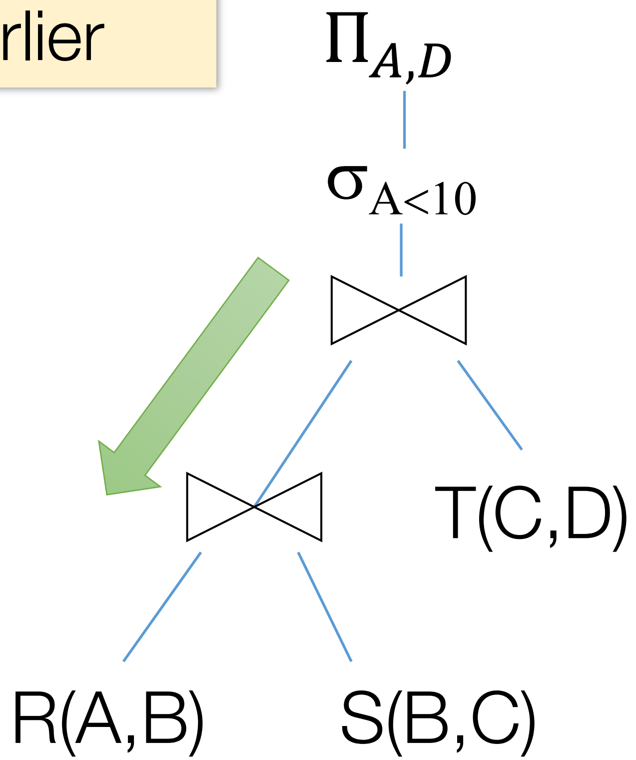
Optimizing RA Plan

```
R(A,B)  S(B,C)  T(C,D)
```

```
SELECT R.A,S.D
FROM R,S,T
WHERE R.B = S.B
  AND S.C = T.C
  AND R.A < 10;
```

$$\Pi_{A,D}(\sigma_{A<10}(T \bowtie (R \bowtie S)))$$

Push down selection on A so it occurs earlier



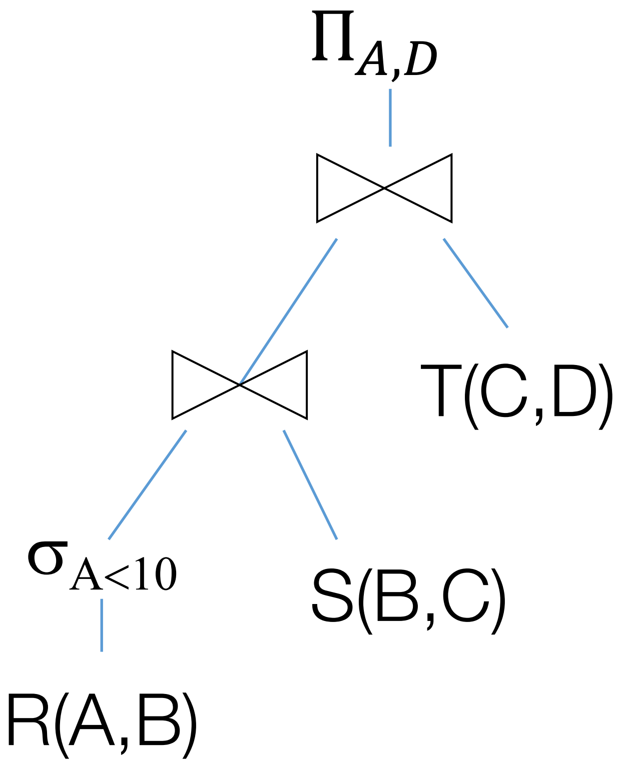
Optimizing RA Plan

```
R(A,B)  S(B,C)  T(C,D)
```

```
SELECT R.A,S.D
FROM R,S,T
WHERE R.B = S.B
  AND S.C = T.C
  AND R.A < 10;
```

$$\Pi_{A,D}\left(T \bowtie \left(\sigma_{A<10}(R) \bowtie S\right)\right)$$

Push down selection on A so it occurs earlier



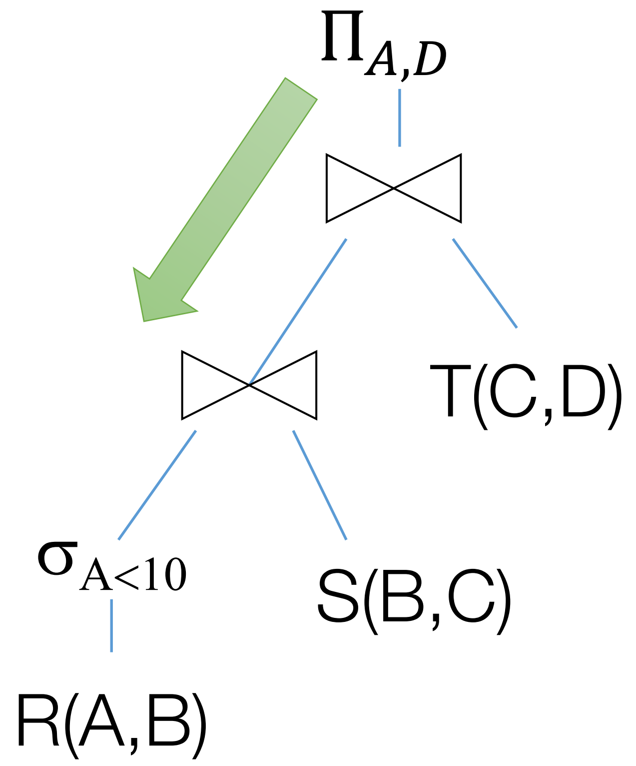
Optimizing RA Plan

```
R(A,B)  S(B,C)  T(C,D)
```

```
SELECT R.A,S.D
FROM R,S,T
WHERE R.B = S.B
   AND S.C = T.C
   AND R.A < 10;
```

Push down projection so it occurs earlier

$$\Pi_{A,D}\left(T \bowtie \left(\sigma_{A<10}(R) \bowtie S\right)\right)$$



Optimizing RA Plan

We eliminate B earlier!

```
R(A,B)  S(B,C)  T(C,D)
```

```
SELECT R.A,S.D
FROM R,S,T
WHERE R.B = S.B
   AND S.C = T.C
   AND R.A < 10;
```

$$\Pi_{A,D}\left(T \bowtie \Pi_{A,c}(\sigma_{A<10}(R) \bowtie S)\right)$$