Optimization Overview

Today's Lecture

1. Logical Optimization

2. Physical Optimization

3. Course Summary

Logical vs. Physical Optimization

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

- **Physical optimization:**
  - 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)*
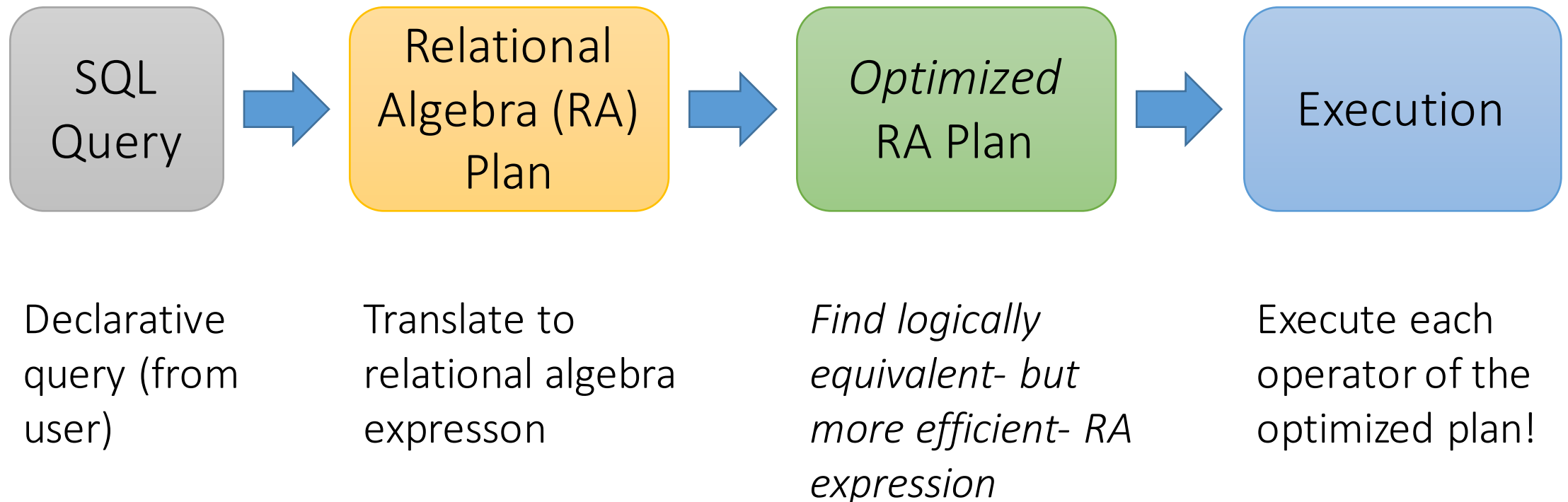
SQL Query

Relational Algebra (RA) Plan

Optimized RA Plan

Execution
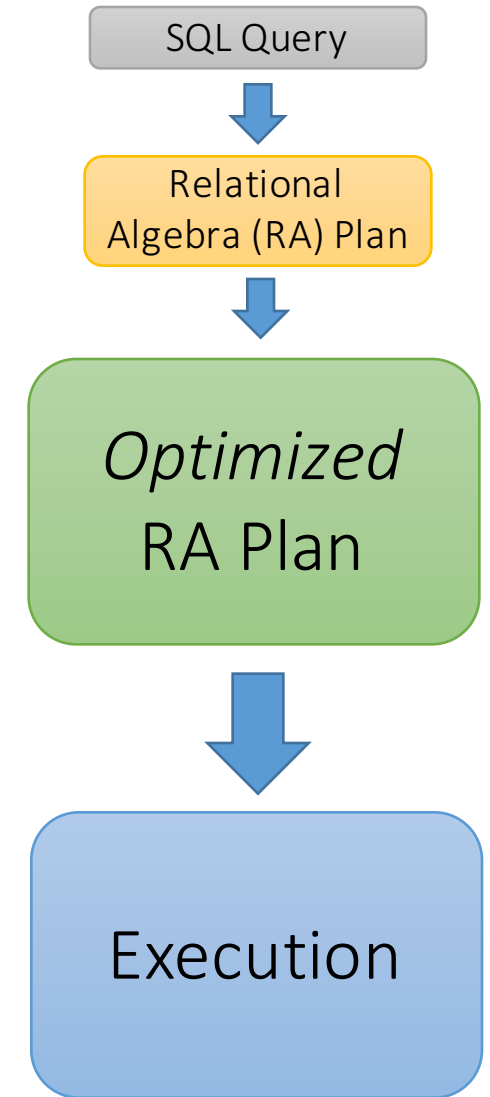
1. Logical Optimization

What you will learn about in this section

1. Optimization of RA Plans

2. ACTIVITY: RA Plan Optimization

RDBMS Architecture

How does a SQL engine work ?

SQL Query	→	Relational Algebra (RA) Plan	→	*Optimized* RA Plan	→	Execution
Declarative query (from user)		Translate to relational algebra expresson		*Find logically equivalent- but more efficient- RA expression*		Execute each operator of the optimized plan!

RDBMS Architecture

How does a SQL engine work ?

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

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

Recall: Relational Algebra (RA)

- <u>Five **basic** operators:</u>

  1. Selection: σ
  2. Projection: Π
  3. Cartesian Product: $\times$

  We'll look at these first!

  4. Union: $\cup$
  5. Difference: -

- <u>Derived or auxiliary operators:</u>

  - 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)

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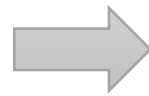
$$\Pi_{gpa,address}(\sigma_{gpa>3.5}(S \bowtie P))$$

How do we represent this query in RA?

Recall: Logical Equivalece of RA Plans

- Given relations R(A,B) and S(B,C):

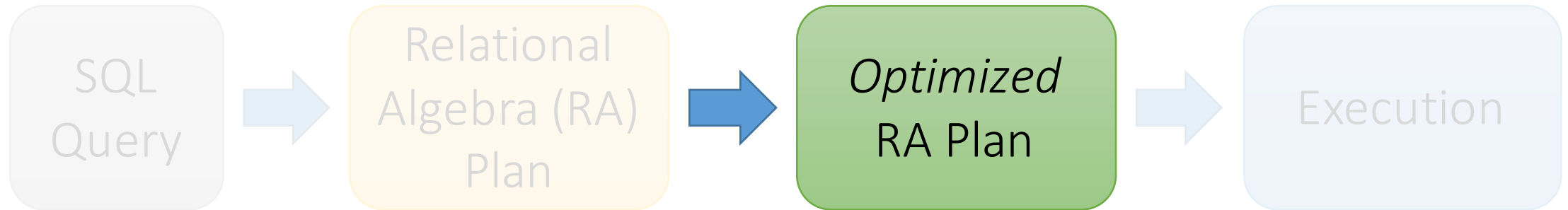 - Here, projection & selection commute:
    - $\sigma_{A=5}(\Pi_A(R)) = \Pi_A(\sigma_{A=5}(R))$

  - What about here?
    - $\sigma_{A=5}(\Pi_B(R)) \; ? = \Pi_B(\sigma_{A=5}(R))$
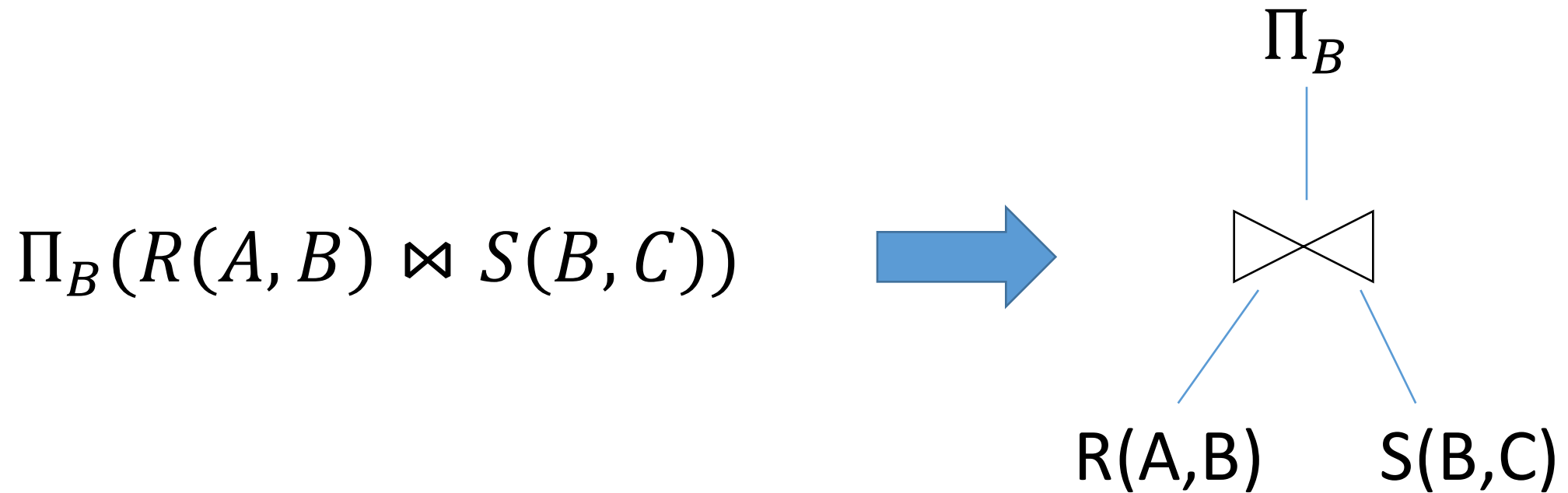
We'll look at this in more depth later in the lecture...

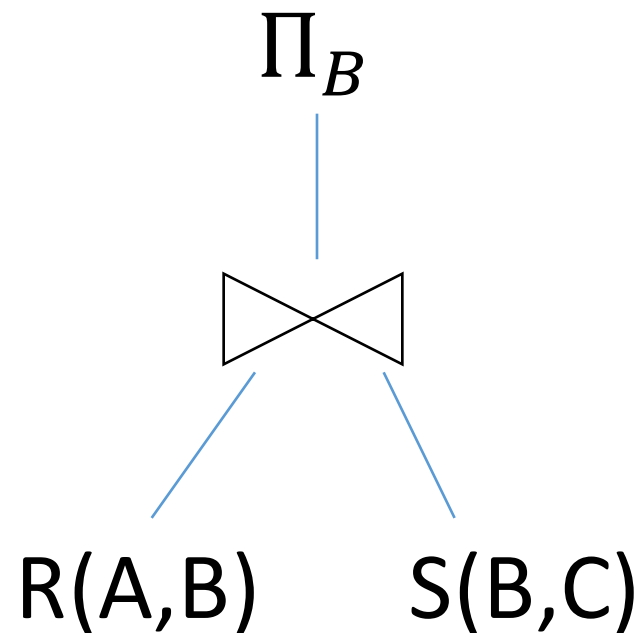# RDBMS Architecture

How does a SQL engine work ?

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

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!

A simple plan

$$\Pi_B$$

$$\bowtie$$

R(A,B)   S(B,C)

What SQL query does this correspond to?

Are there any logically equivalent RA expressions?

"Pushing down" projection
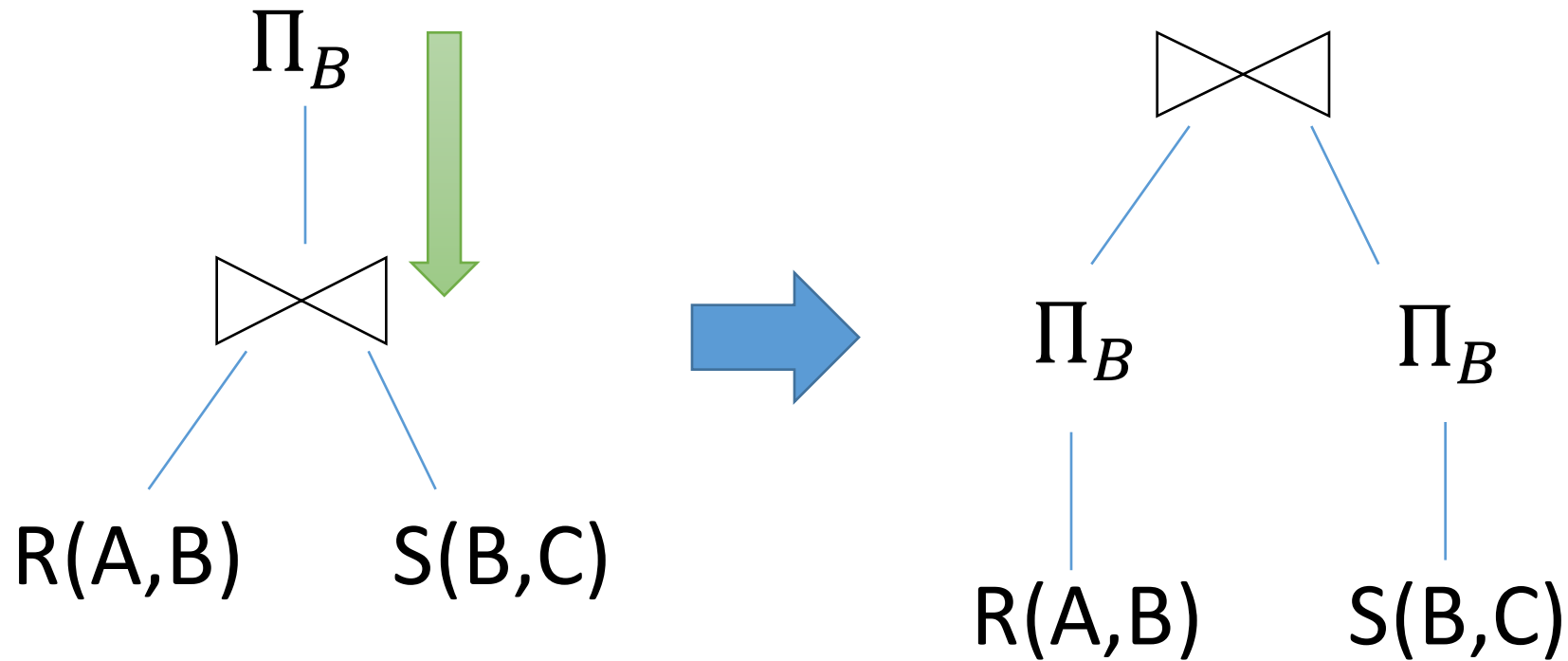
Why might we prefer this plan?

Takeaways

- This process is called logical optimization

- Many equivalent plans used to search for "good plans"

- Relational algebra is an important abstraction.

RA commutators

- The basic commutators:
  - Push **projection** through **(1) selection**, **(2) join**
  - Push **selection** through **(3) selection, (4) projection, (5) join**
  - *Also:* Joins an be re-ordered!

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

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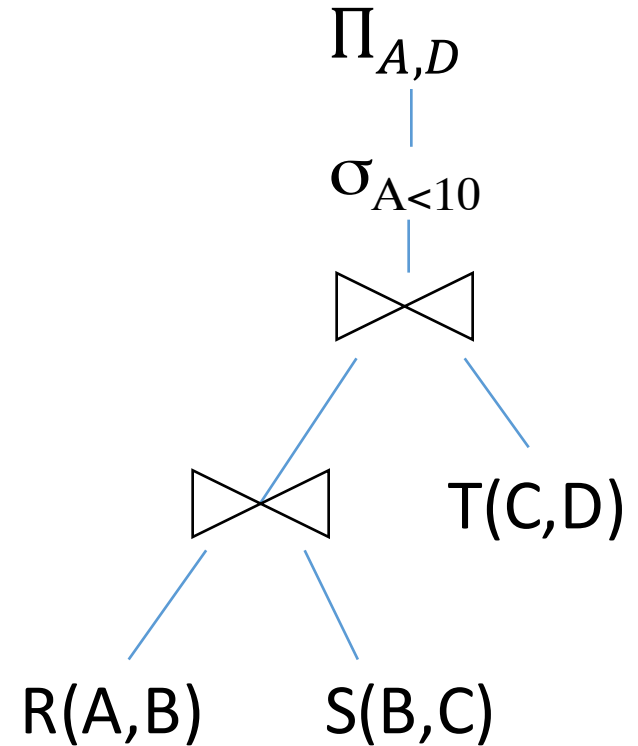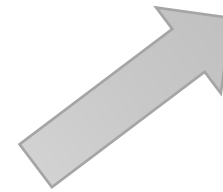
Optimizing the SFW RA Plan

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;
```

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

$\Pi_{A,D}$

$\sigma_{A<10}$

T(C,D)

R(A,B)   S(B,C)

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).
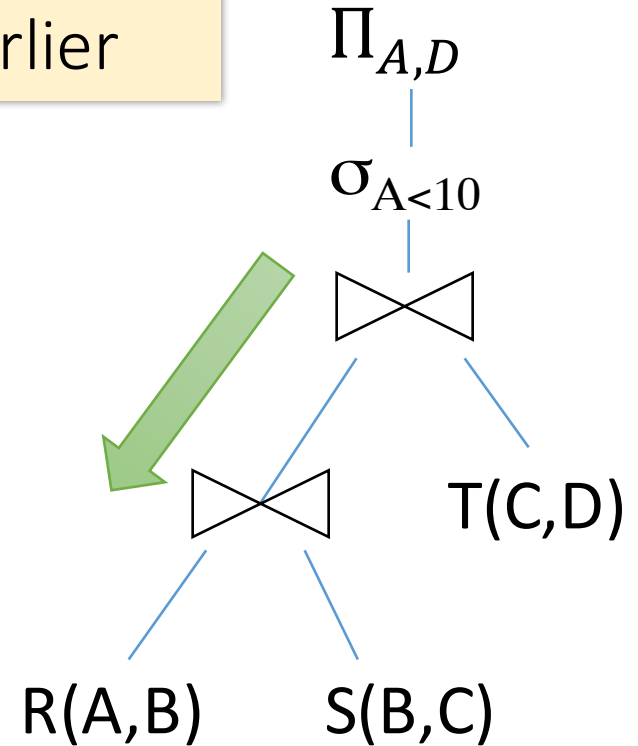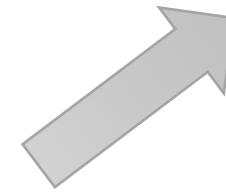 - Projection could be a waste of effort, but more rarely.

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;
```

Push down selection on A so it occurs earlier

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

$\Pi_{A,D}$

$\sigma_{A<10}$

T(C,D)

R(A,B)   S(B,C)

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 selection on A so it occurs earlier

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

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;
```
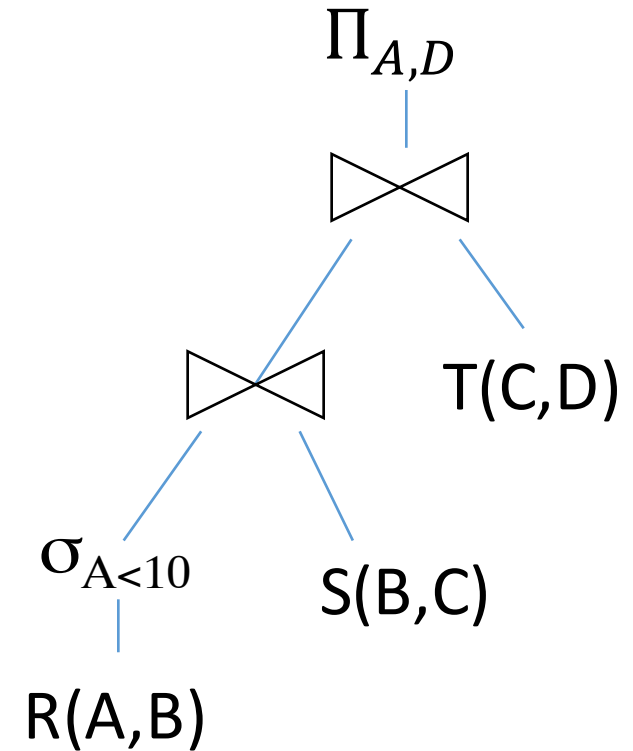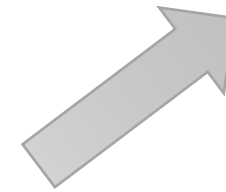
Push down projection so it occurs earlier
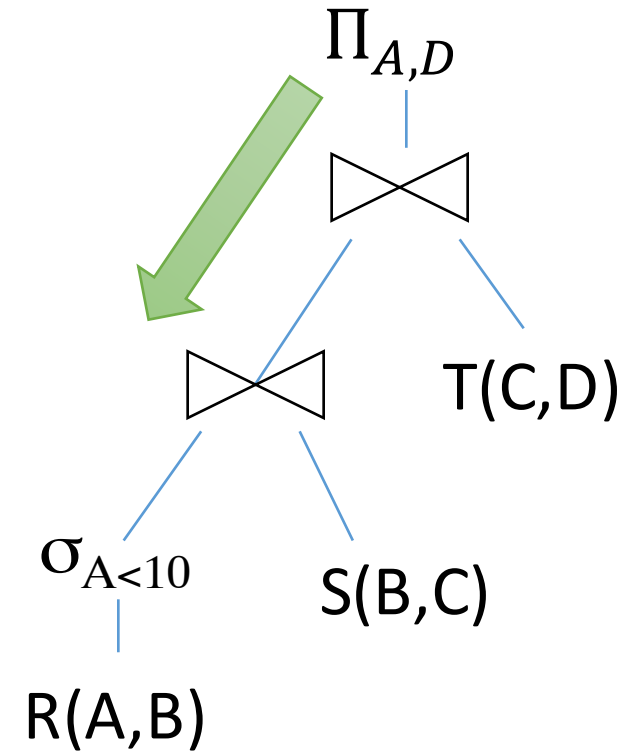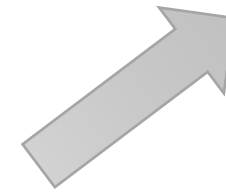
$$\Pi_{A,D}\big(T \bowtie (\sigma_{A<10}(R) \bowtie S)\big)$$

$\Pi_{A,D}$

$\sigma_{A<10}$

T(C,D)

S(B,C)

R(A,B)

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;
```

We eliminate B earlier!

In general, when is an attribute not needed...?

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

$\Pi_{A,D}$

$\bowtie$

$\Pi_{A,C}$ T(C,D)

$\bowtie$

$\sigma_{A<10}$ S(B,C)

R(A,B)
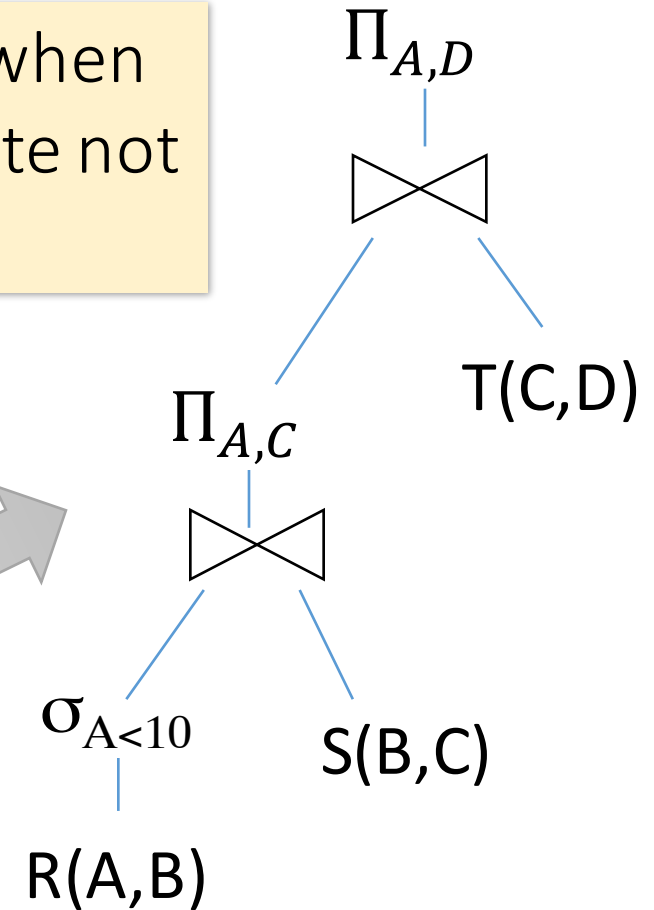
Activity-17-1.ipynb

2. Physical Optimization

What you will learn about in this section

1. Index Selection

2. Histograms

3. ACTIVITY

Index Selection

Input:
- Schema of the database
- **Workload description:** set of (query template, frequency) pairs

Goal: Select a set of indexes that minimize execution time of the workload.
- Cost / benefit balance: Each additional index may help with some queries, but requires updating

This is an optimization problem!

Example

Workload description:

```
SELECT pname
FROM Product
WHERE year = ? AND category = ?
```

Frequency 10,000,000

```
SELECT pname,
FROM Product
WHERE year = ? AND Category = ?
AND manufacturer = ?
```

Frequency 10,000,000

Which indexes might we choose?

Example

Workload description:

```
SELECT pname
FROM Product
WHERE year = ? AND category =?
```

Frequency
10,000,000

```
SELECT pname,
FROM Product
WHERE year = ? AND Category =?
AND manufacturer = ?
```

Frequency
100

Now which indexes might we choose? Worth keeping an index with manufacturer in its search key around?

Simple Heuristic

- Can be framed as standard optimization problem: Estimate how cost changes when we add index.
  - We can ask the optimizer!

- Search over all possible space is too expensive, optimization surface is really nasty.
  - Real DBs may have 1000s of tables!

- Techniques to exploit *structure* of the space.
  - In SQLServer Autoadmin.

NP-hard problem, but can be solved!

How do we get index cost?

- Note that to phrase as optimization problem, we need an estimate of the **cost** of an index lookup

- In other words: we need estimates of the <u>result set size</u>
  - The cost of e.g. an unclustered index vs. a scan is $\propto$ result set size

- So how do we estimate the result set size from an index lookup?

We can use histograms...

Histograms & IO Cost Estimation

IO Cost Estimation via Histograms

- For **index selection**:
  - What is the cost of an index lookup?

- Also for **deciding which algorithm to use**:
  - Ex: To execute $R \bowtie S$, which join algorithm should DBMS use?

  - **What if we want to compute $\sigma_{A>10}(\mathbf{R}) \bowtie \sigma_{B=1}(S)$?**

 - In general, we will need some way to *estimate intermediate result set sizes*

Histograms provide a way to efficiently store estimates of these quantities

Histograms

- A histogram is a set of value ranges ("buckets") and the frequencies of values in those buckets occurring

- How to choose the buckets?
  - Equiwidth & Equidepth

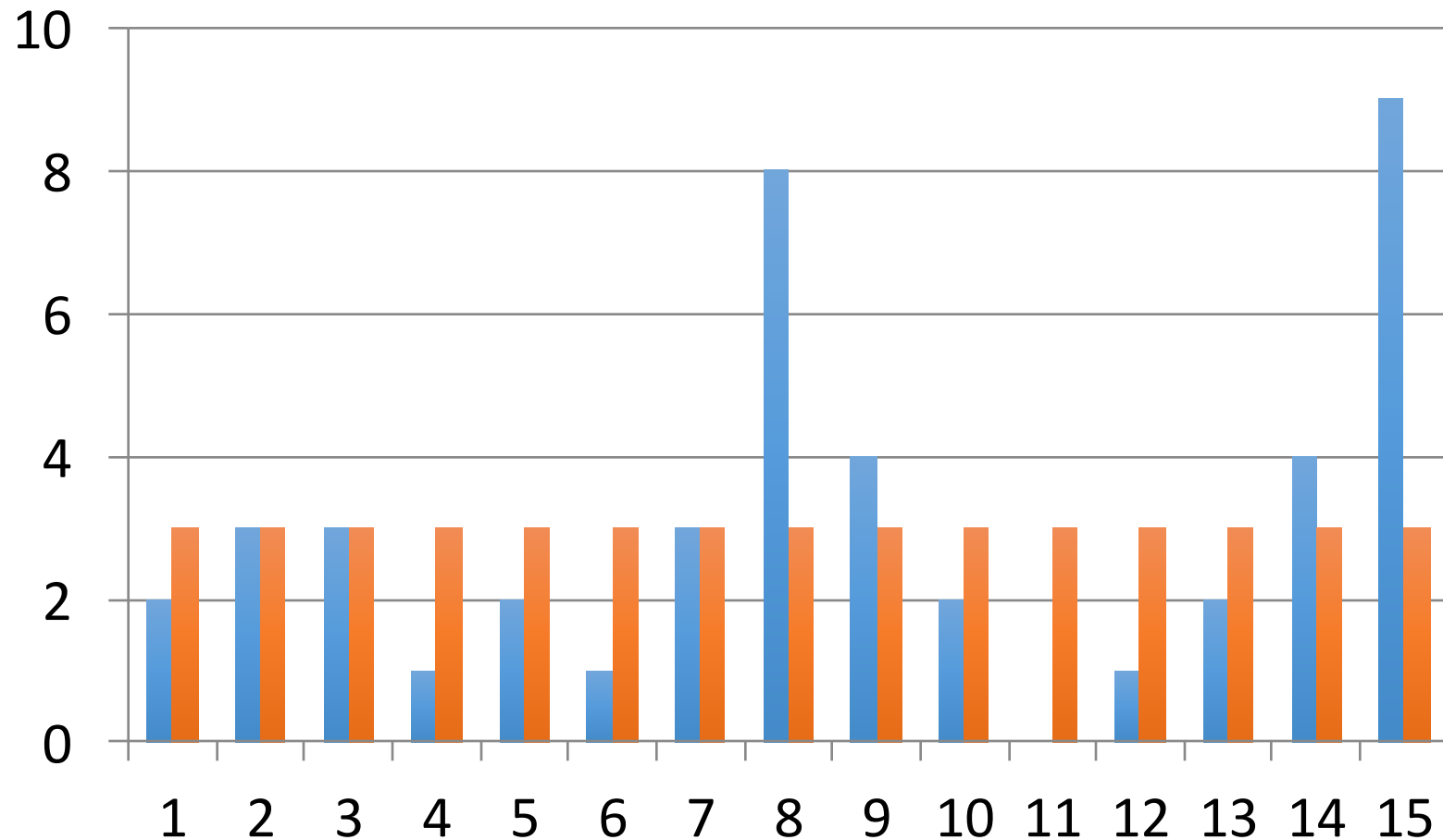- Turns out high-frequency values are **very** important

Example
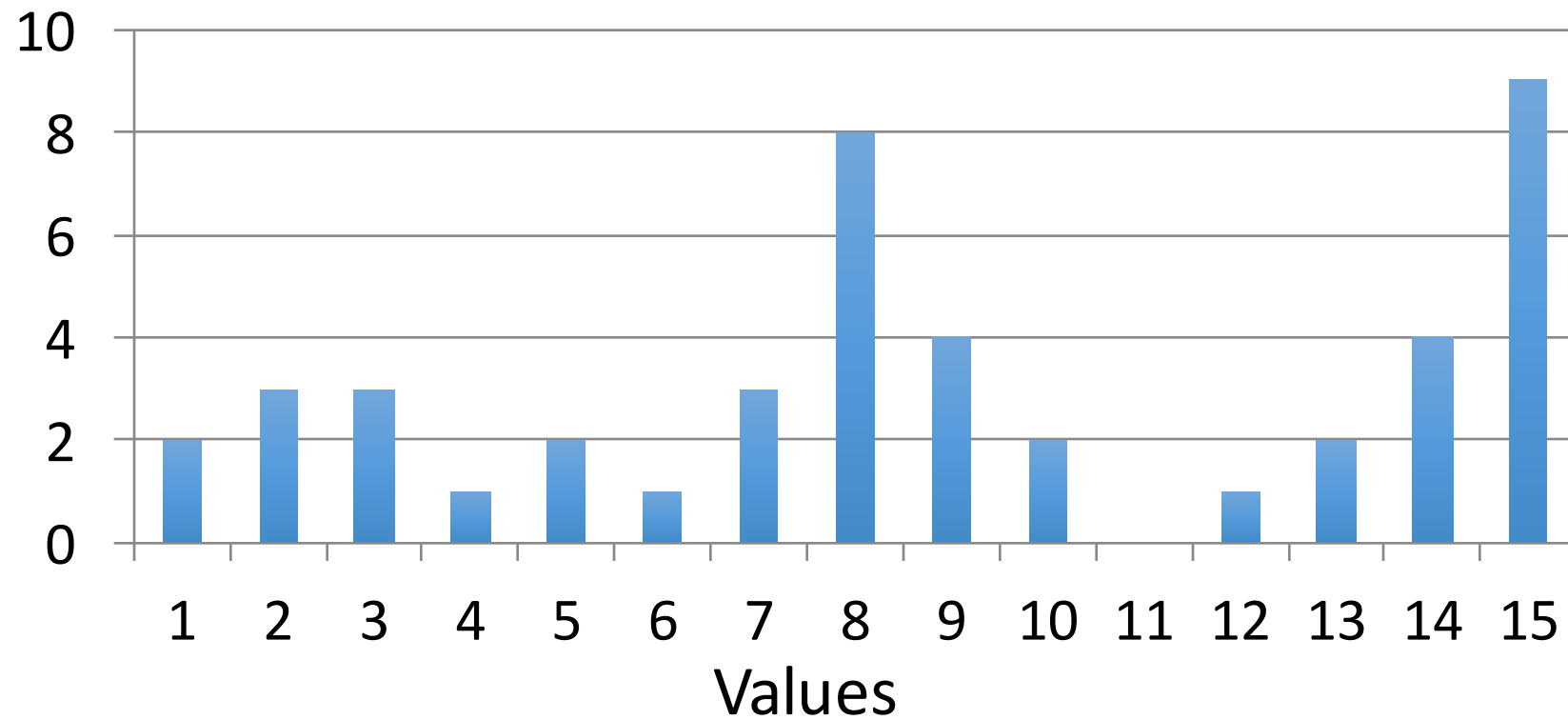
How do we compute how many values between 8 and 10?
(Yes, it's obvious)

Problem: counts take up too much space!

Full vs. Uniform Counts

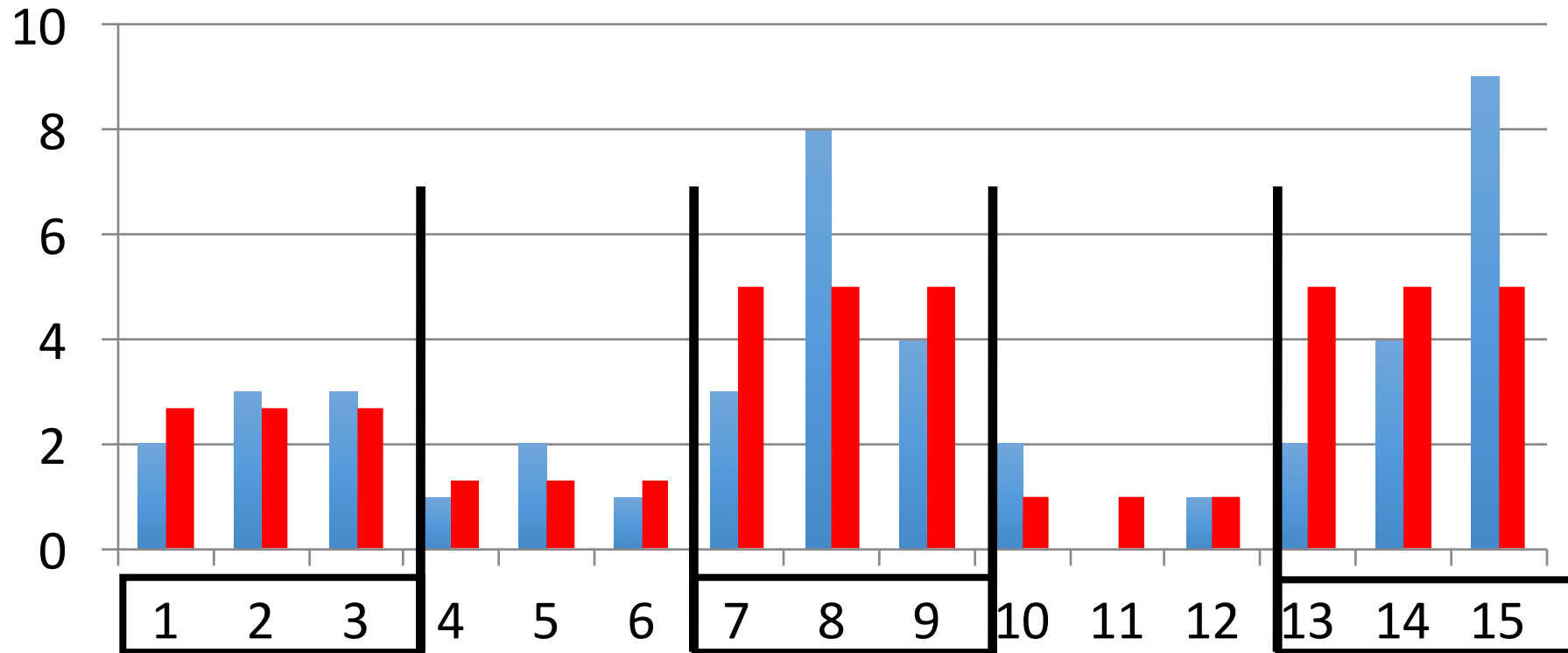How much space do the full counts (bucket_size=1) take?

How much space do the uniform counts (bucket_size=ALL) take?

Fundamental Tradeoffs

- Want high resolution (like the full counts)

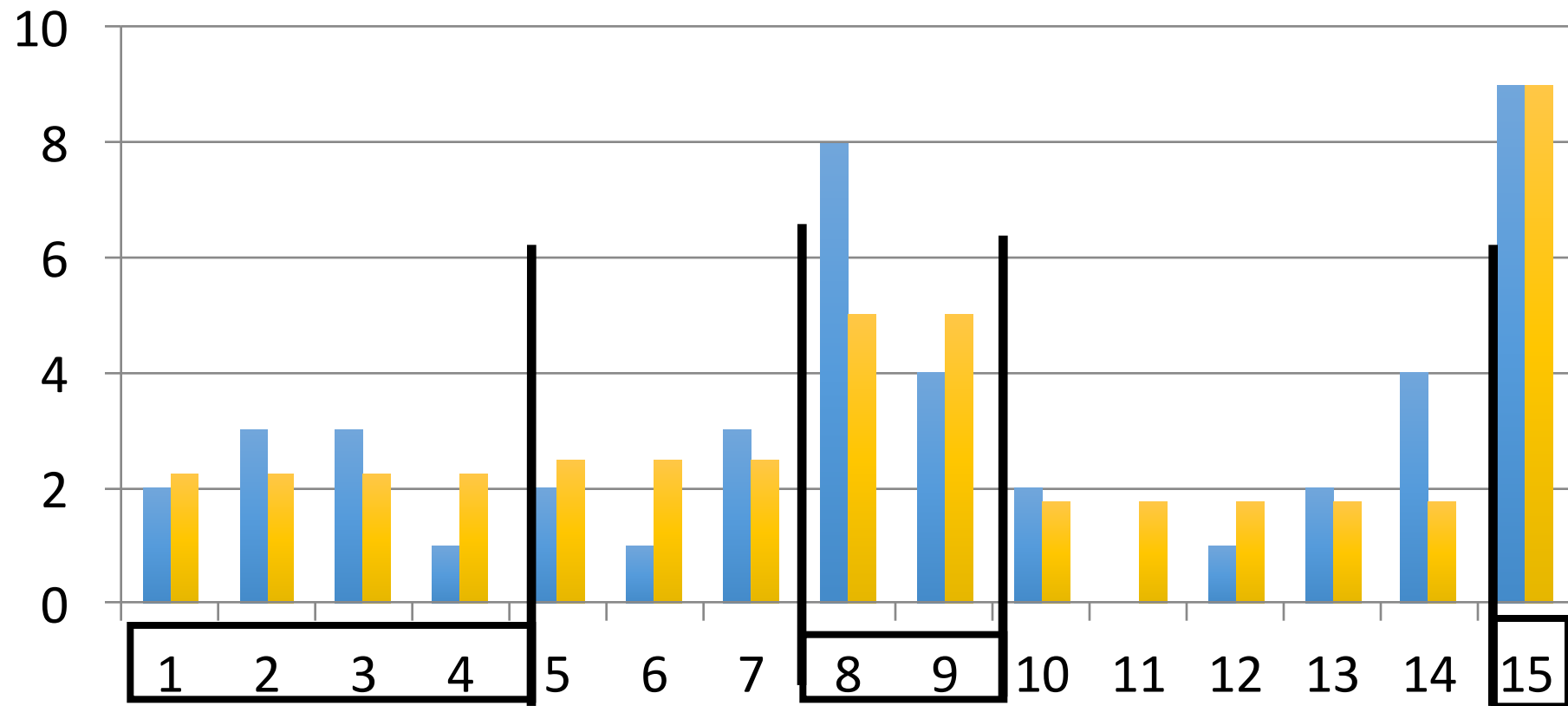- Want low space (like uniform)

- Histograms in are a compromise!

So how do we compute the "bucket" sizes?

Equi-width

All buckets roughly the same width

Equidepth

All buckets contain roughly the same number of items (total frequency)

Histograms

- Simple, intuitive and popular

- Parameters: # of buckets and type

- Can extend to many attributes (multidimensional)

Maintaining Histograms

- Histograms require that we update them!
  - Typically, you must run/schedule a command to update statistics on the database
  - Out of date histograms can be terrible!

- There is research work on self-tuning histograms and the use of query feedback
  - Oracle 11g

Nasty example
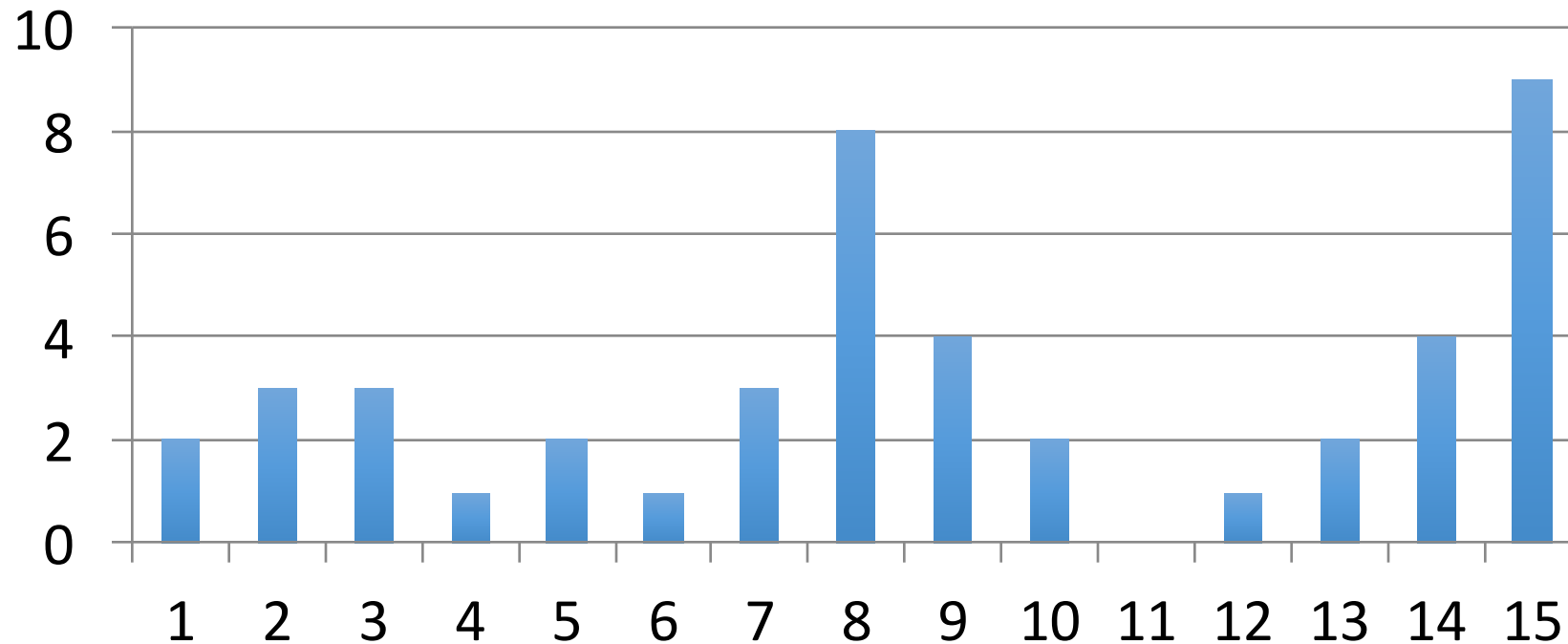
1. we insert many tuples with value > 16
2. we do **not** update the histogram
3. we ask for values > 20?

Compressed Histograms

- One popular approach:
    1. Store the most frequent values and their counts explicitly
    2. Keep an equiwidth or equidepth one for the rest of the values

People continue to try all manner of fanciness here that people try
wavelets, graphical models, entropy models,…

Activity-17-2.ipynb

3. Course Summary

Course Summary

- We learned...

  1. **How to design a database**

Course Summary

• We learned…

1. How to design a database

2. **How to query a database, even with concurrent users and crashes / aborts**

Course Summary

- We learned…

  1. How to design a database

  2. How to query a database, even with concurrent users and crashes / aborts

  3. **How to optimize the performance of a database**

1. Intro

2-3. SQL

4. ER Diagrams

5-7. DB Design

8-9. TXNs

12. IO Cost

13. Indexes

14-15. Joins

16. Rel. Algebra

Course Summary

- We learned…

  1. How to design a database

  2. How to query a database, even with concurrent users and crashes / aborts

  3. How to optimize the performance of a database

- We got a sense (as the old joke goes) of the three most important topics in DB research:
  - Performance, performance, and performance

1. Intro

2-3. SQL

4. ER Diagrams

5-7. DB Design

8-9. TXNs

12. IO Cost

13. Indexes

14-15. Joins

16. Rel. Algebra