6.5830 Lecture 14

Advanced Cardinality Estimation

October 30, 2023

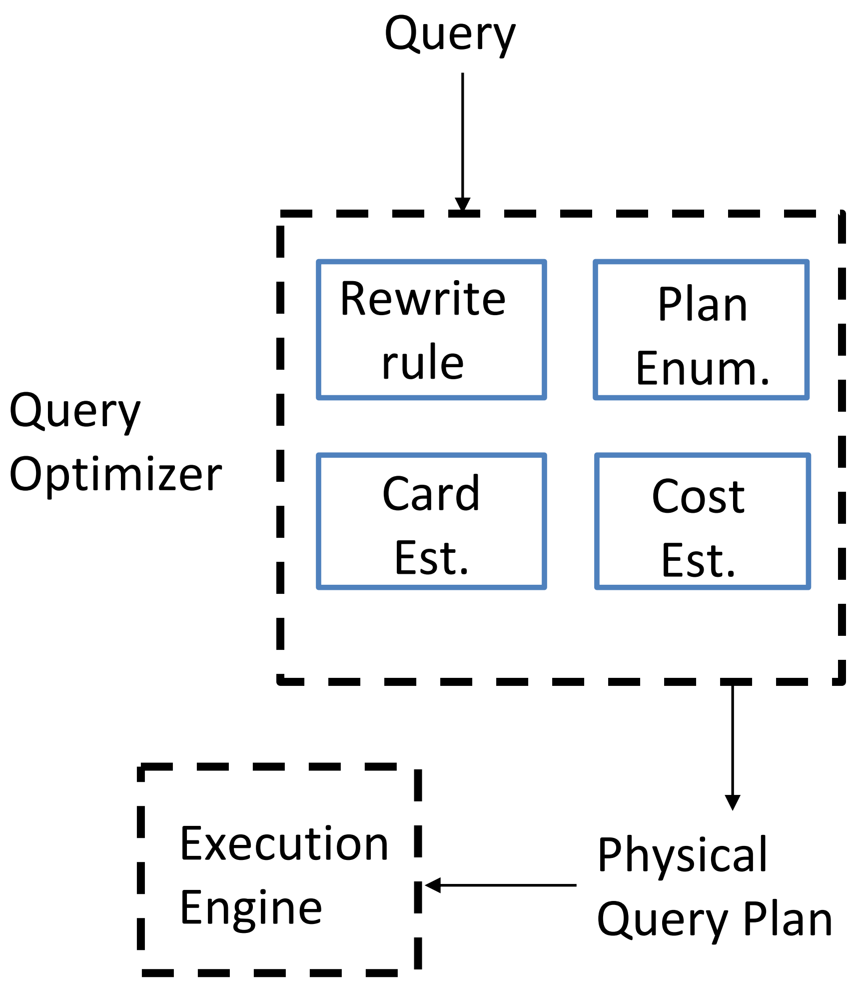
Logistics

- 6.5830 Project meetings
- Lab3 due this Friday
- PS3 release this Wednesday due Nov 27
  - Treat as quiz 2 practice

Recap: Query optimization

Query Optimizer:

- Rewrite rules
  - Expert-designed rules
- Plan enumeration
  - Selinger DP
- Cardinality estimator
  - Crucial for join ordering and operator selection
  - Arguably the most challenging problem
- Cost model
  - CPU/IO cost calculation

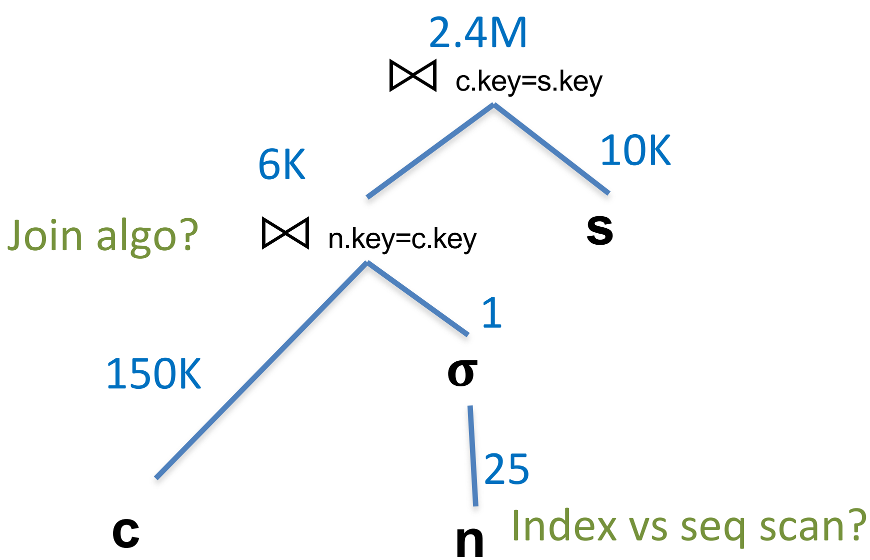


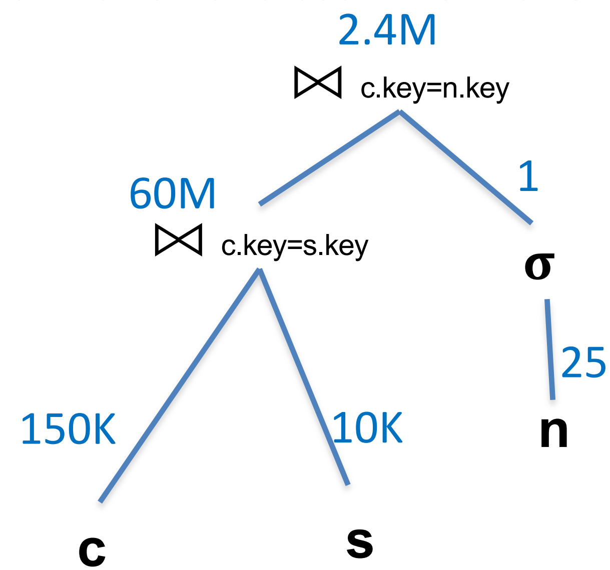
Example (PS2): why crucial

```
SELECT * FROM nation n,
              customer c, supplier s
WHERE n.nationkey = c.nationkey
   AND s.nationkey = c.nationkey
   AND n.name = 'GERMANY'
```

Which join order is better?

What join algo to choose?
What access method to choose?
(Need to consider cardinality estimation and cost information)





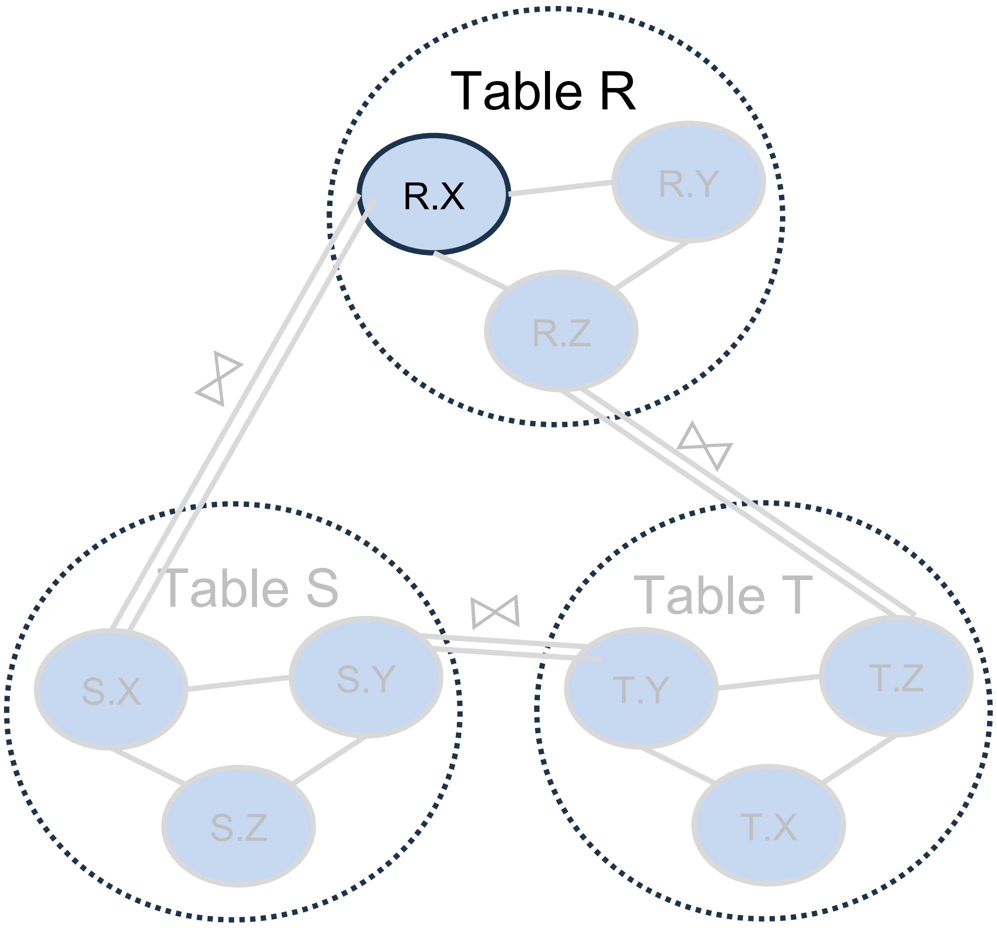
Accurate cardinality estimation is crucial and very challenging.

Overview: Why so challenging

Single column -> very easy.

Multiple columns -> harder because of correlation.

Multiple tables (join)-> Much much harder because distribution and correlation changes after join.

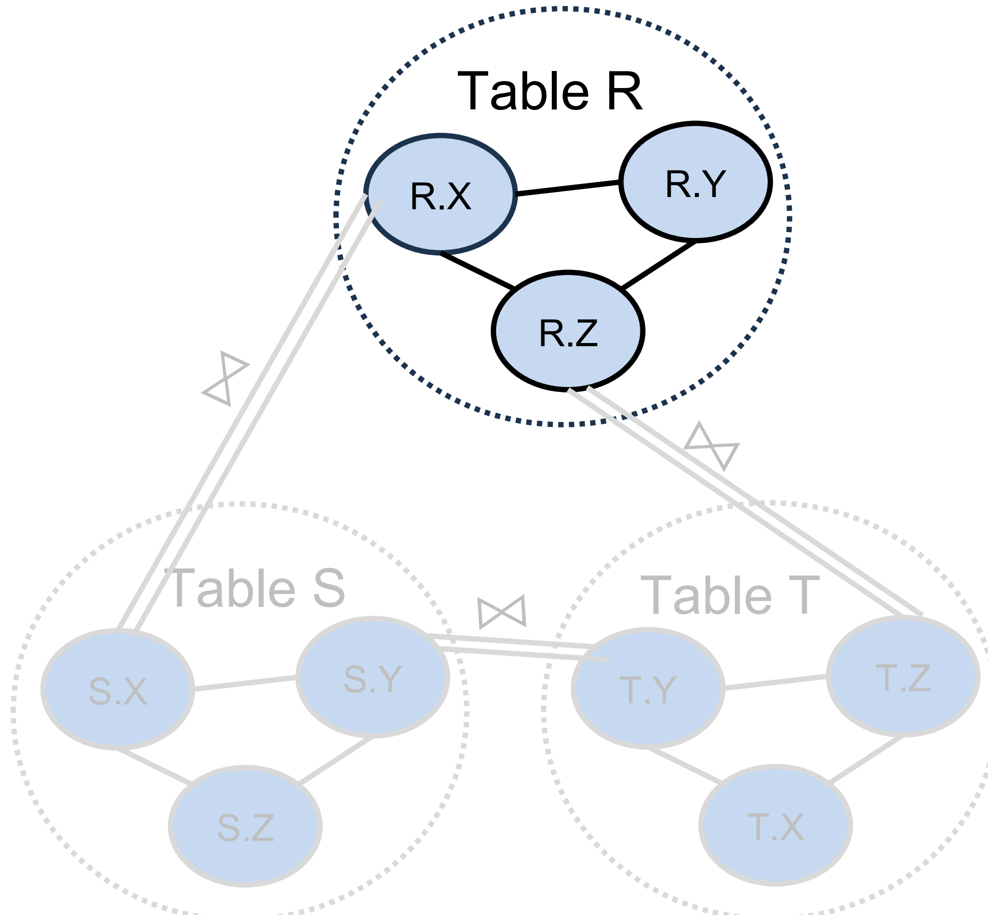


Overview: Why so challenging

Single column -> very easy.

Multiple columns -> harder because of correlation.

Multiple tables (join)-> Much much harder because distribution and correlation changes after join.

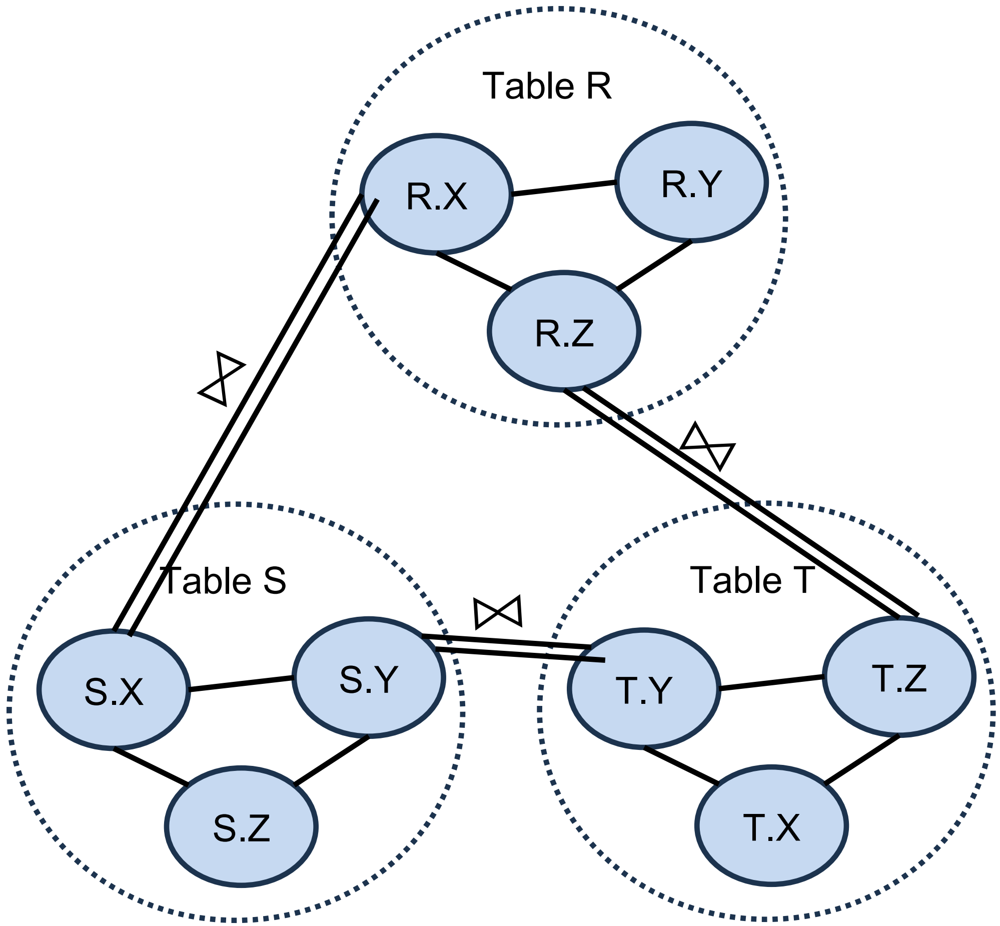


Overview: Why so challenging

Single column -> very easy.

Multiple columns -> harder because of correlation.

Multiple tables (join)-> Much much harder because distribution and correlation changes after join.



Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances

Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances

What are we estimating?

Table R

X	Y
1	10
2	6
2	2
3	31
4	44
8	-5
8	-12
10	-82
15	97

…

…

…

Filter predicate

Selectivity Pr(X<5)

- Card(X<5) = 5/9 * 9 = 5

Table size {R}

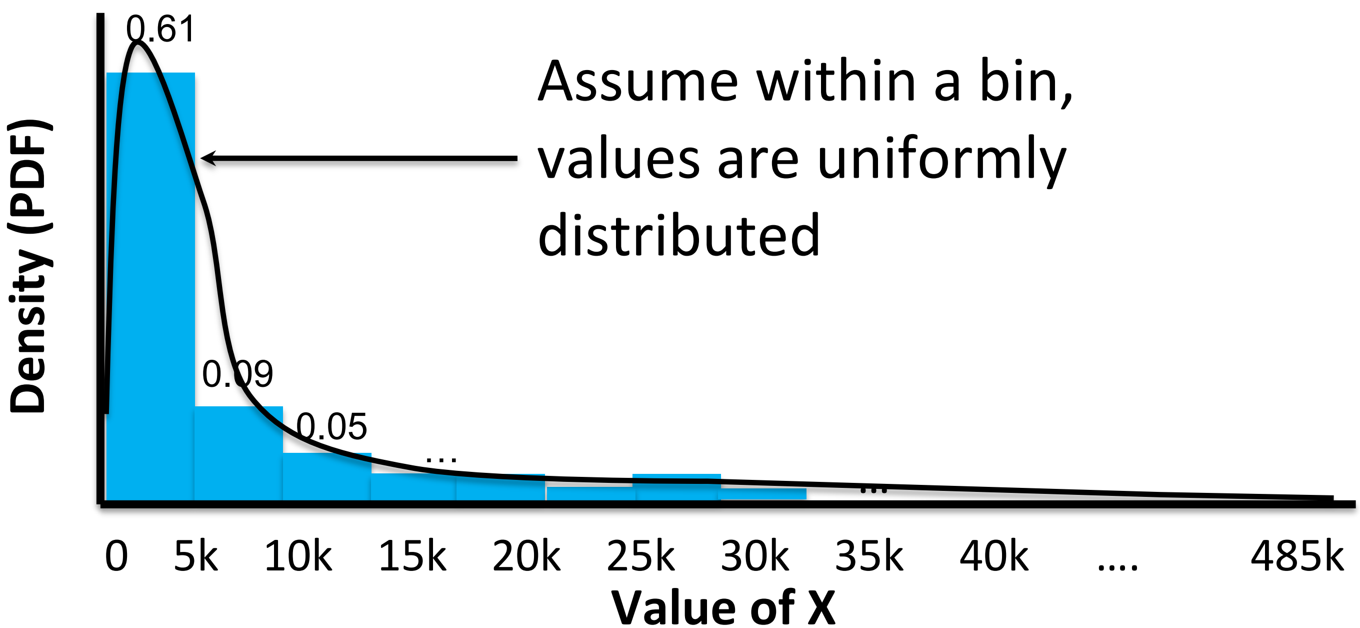
Cardinality

Isn't this just probability distribution P(X) of X?

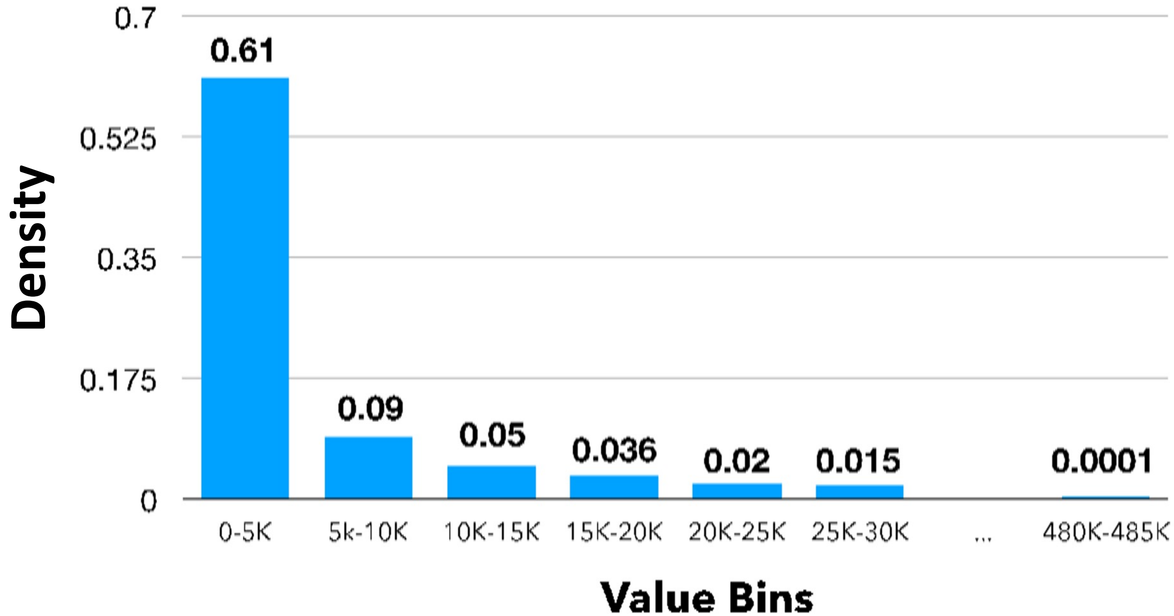
We denote the estimation of Card(X) as ^Card(X)

Equal-width histograms

- Histograms can approximate any distribution (pdf) for a single attribute.
- Easy to build (ANALYZE): scan (sample of) one table.



Equal-width histograms



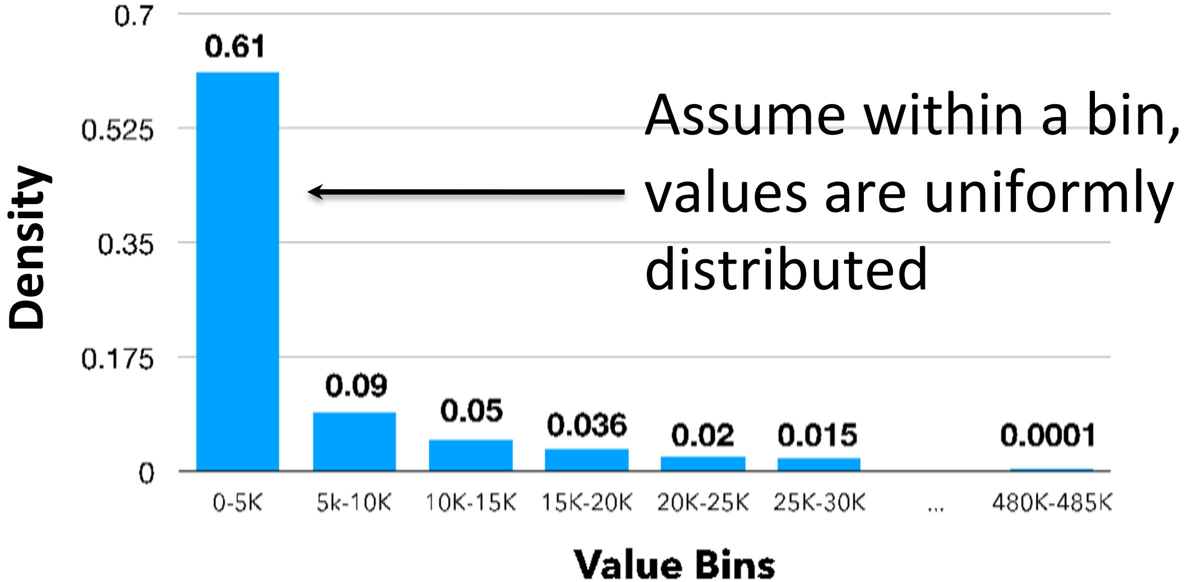
What is the estimation of ^Card(X<10K)? (0.61 + 0.09)*{R}

Is this accurate? ^Card(X<10K) = Card(X<10K)

What about ^Card(X<23K)?

We denote the estimation of Card(X) as ^Card(X)

Equal-width histograms https://clicker.mit.edu/6.5830/



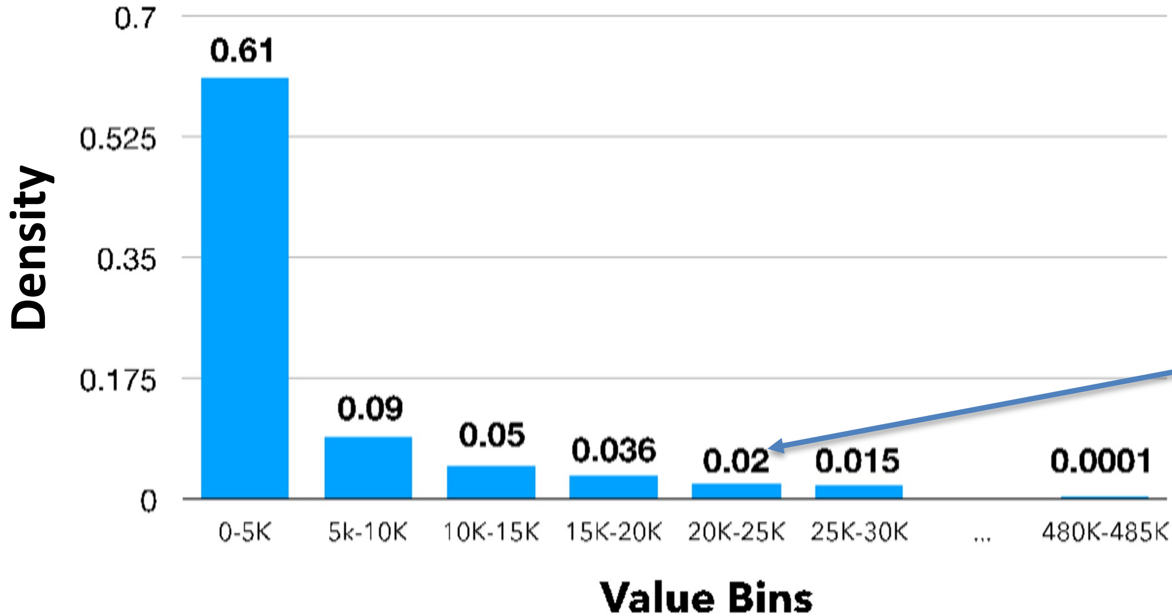
What is the estimation ^Card(X<23K)?

(a) 0.786 * {R} (b) 0.806 * {R}

(c) 0.798 * {R} (d) 0.794 * {R}

We denote the estimation of Card(X) as ^Card(X)

Equal-width histograms



Assume: values are uniformly distributed within a bin

^Card(X<23K) = (0.61+0.09+0.05+0.036+0.02*3/5)*{R}

Is this accurate? At most off by +- 0.02 *{R}

What about ^Card(X=0)? Is this accurate?

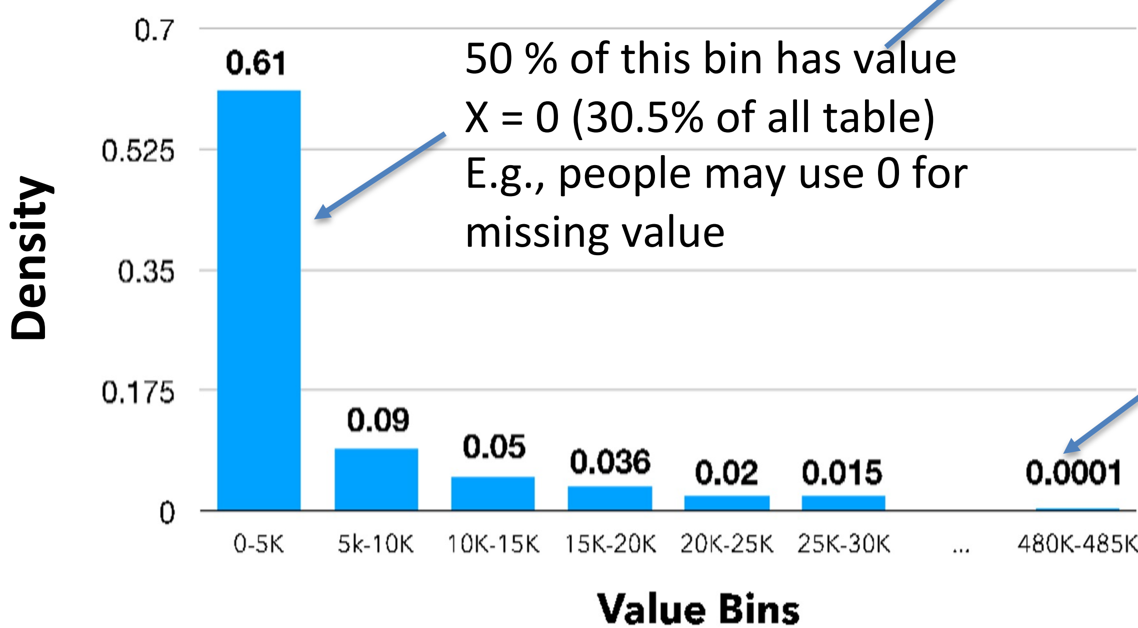
We denote the estimation of Card(X) as ^Card(X)

Equal-width histograms

Reality: non-uniform distribution within a bin.

^Card(X=0) = 0.61*1/5000*{R}   Card(X=0) = 0.61*0.5*{R}

2500x under-estimate!!!!



50 % of this bin has value X = 0 (30.5% of all table)
E.g., people may use 0 for missing value

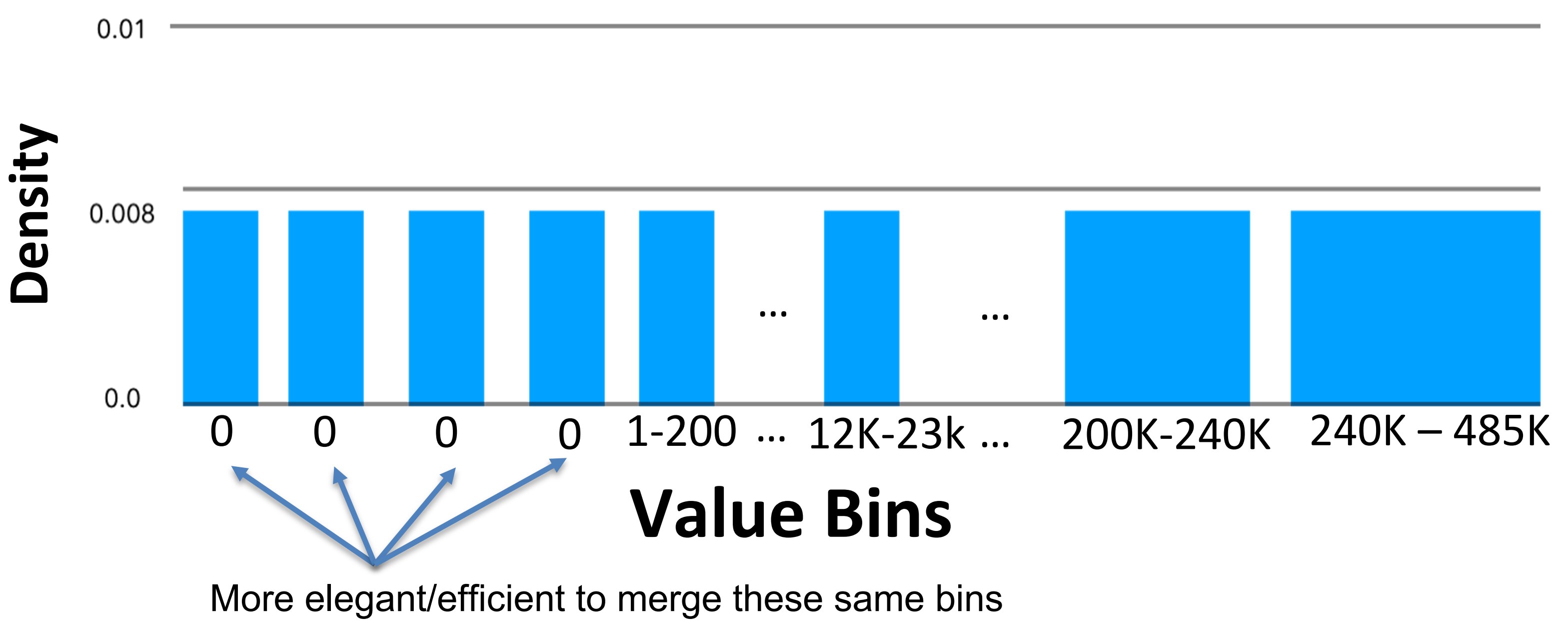
What can we do?

- Equal-depth histograms
- Most common values (MCV)

Not balanced
This bin really doesn't make a difference in terms of accuracy

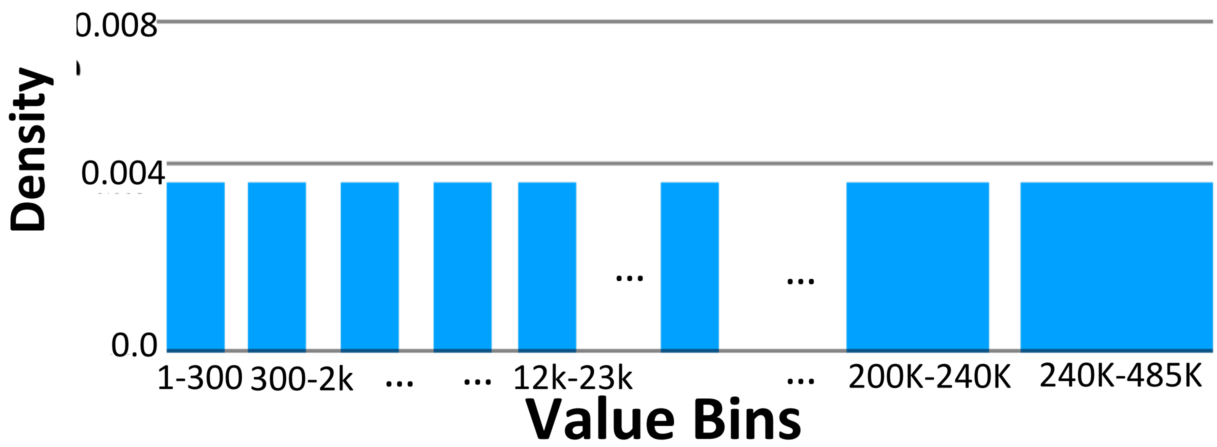
We denote the estimation of Card(X) as ^Card(X)

Equal-depth histograms



Bin width is different but every bin has the same density.
More efficient and more accurate estimation.
Slightly more expensive to build and maintain (e.g. keep balance during data update) than equal-width histogram

Histograms + Most Common Values (MCV)



MCV Table

Value	Density
0	0.305
2	0.05
8	0.008
…	…
340	0.0002

^Card(X<10) = (0.305 + 0.05 + 0.008 + 0.004*10/300)*{R}

First check the MCV table

Then check the histograms

Increases accuracy, especially for point filter estimation.

Relaxes the assumption that values are uniformly distributed within each bin.

Cardinality estimation of a filter on a single-column is very accurate and efficient.

Stats in Postgres

```
mbta=> SELECT histogram_bounds FROM pg_stats
[WHERE tablename='station_orders' AND attname='distance_from_last_station_miles';
                   histogram_bounds
---------------------------------------------------------
 {0.15,0.3,0.35,0.42,0.43,0.47,0.54,0.58,0.59,0.7}
(1 row)
```

11 bins

Default: equal-depth histograms + MCVs

```
mbta=> SELECT most_common_vals, most_common_freqs FROM pg_stats
[WHERE tablename='station_orders' AND attname='distance_from_last_station_miles';

most_common_vals
                  |

                                                                    most_common_freqs
-------------------------------------------------------
 {0,0.27,0.57,0.71,0.25,0.51,0.55,0.56,0.66,0.69,0.82,0.83,0.95,0.96,0.99,1.08,0.22,0.26,0.17,0.18,0.23,0.
28,0.29,0.31,0.38,0.4,0.41,0.44,0.52,0.62,0.63,0.73,0.74,0.78,0.8,0.81,0.84,0.85,0.9,1,1.07,1.19,1.28,1.33
,1.75,1.86,3.53} | {0.036809817,0.018404908,0.015337423,0.015337423,0.012269938,0.012269938,0.012269938,0.
012269938,0.012269938,0.012269938,0.012269938,0.012269938,0.012269938,0.012269938,0.012269938,0.012269938,
0.009202454,0.009202454,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.00613496
9,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134
969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.006134969,0.0061
34969,0.006134969,0.006134969,0.006134969,0.006134969}
(1 row)
```

47 MCVs

Number of bins and MCVs are tunable parameters of Postgres

Stats in Postgres

```
mbta=> SELECT histogram_bounds FROM pg_stats
WHERE tablename='station_orders' AND attname='route_id';
 histogram_bounds
------------------

(1 row)
```

0 bin (no histogram needed)

Recap PS2: why is the no estimation error for filter ‘route_id > 10’?

```
mbta=> SELECT most_common_vals, most_common_freqs FROM pg_stats
WHERE tablename='station_orders' AND attname='route_id';
               most_common_vals                |
                                       most_common_freqs

-----------------------------------------------+-------------------------------------------------------
------------------------------------------------------------------------------------------------------------
---------------------------------------------------
 {2,6,4,8,3,7,5,9,12,13,15,17,14,16,0,1,10,11} | {0.07668712,0.07668712,0.06748466,0.06748466,0.064417176,
0.064417176,0.061349694,0.061349694,0.061349694,0.061349694,0.055214725,0.055214725,0.05214724,0.05214724,
0.036809817,0.036809817,0.024539877,0.024539877}
```

18 MCVs

No need for histogram. Perfect stats using MCVs unless data changes.

Postgres automatically “ANALYZE” the table when it is first loaded and whenever changed, unless manually turned off.

What about multi-column filters?

Table R

X	Y
1	10
2	6
2	2
3	31
4	44
8	-5
8	-12
10	-82
15	97

...

...

...

Filter on R: X < 5 AND Y < 0

Attribute independence assumption

Estimated → ^Card(X<5 AND Y<0)

= P(X<5) * P(Y<0) * {R}

= 5/9 * 3/9 * {R}

True → Card(X<5 AND Y<0) = 0

Large estimation error because X and Y are not independent. This error grows exponentially w.r.t. number of columns.

We denote the estimation of Card(X) as ^Card(X)

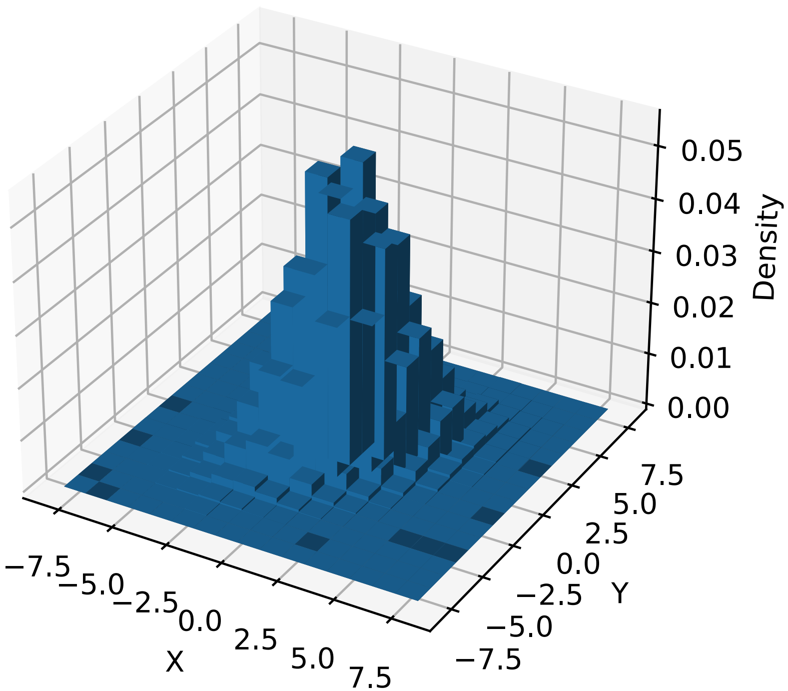
1-D Histogram summary

- Histograms are the most widely used cardinality estimation method, used in Postgres and various commercial DBMSes.
- Pros
  - Fast to build
  - Negligible memory and inference overhead
    - O(nbins)   linear memory and inference time w.r.t. num of bins
  - Easy to update with new data
- Cons
  - Inaccurate for filters involving multiple columns
  - Inaccurate for join size estimation (discuss later)

Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances

Multi-dimensional histograms



Bin the value domain of two attributes.

Multi-dimensional MCV

Value of X	Value of Y	Density
0	0	0.05
0	1	0.04
0	-1	0.03
…		…
2	-3.5	0.0001

Multi-dimensional histograms https://clicker.mit.edu/6.5830/

Table R has attributes X, Z, Y, we want to estimate the cardinality of filter $X = 0$ and $Y < 0$ and $Z > 5$.

You build a 2d histogram on X and Z to estimate ^Card(X = 0 and Z > 5) = 0.2 * {R} and a 1d histogram on Y to estimation ^Card(Y < 0) = 0.2 * {R}. Assume that the true card is 0.16 * {R}. How many times did you under/over estimate by?

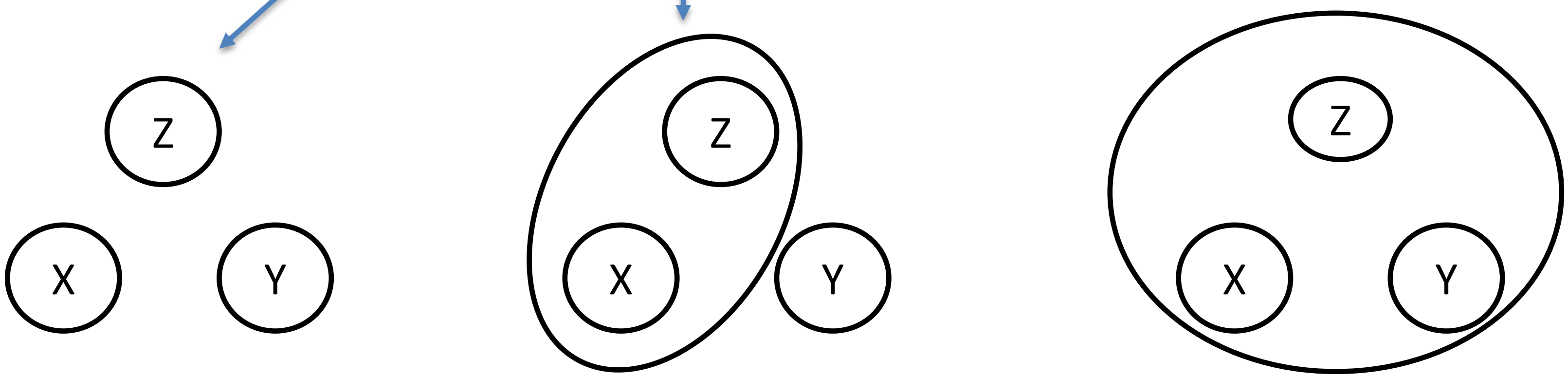
(a) Underestimate by 2x
(b) Overestimate by 2x
(c) Underestimate by 4x
(d) Overestimate by 4x

We denote the estimation of Card(X) as ^Card(X)

Multi-dimensional histograms

Table R with three attribute X, Y, Z

Filter on R with X = 0 and Y < 0 and Z > 5



P(X)*P(Y)*P(Z)

Independence Assumption

O(nbins)

P(X, Z)*P(Y)

Some Independence Assumption

O(nbins^2)

P(X, Z, Y)

No assumption O(nbins^3)

May not be affordable

Strong assumption

No assumption

Multi-dimensional histograms

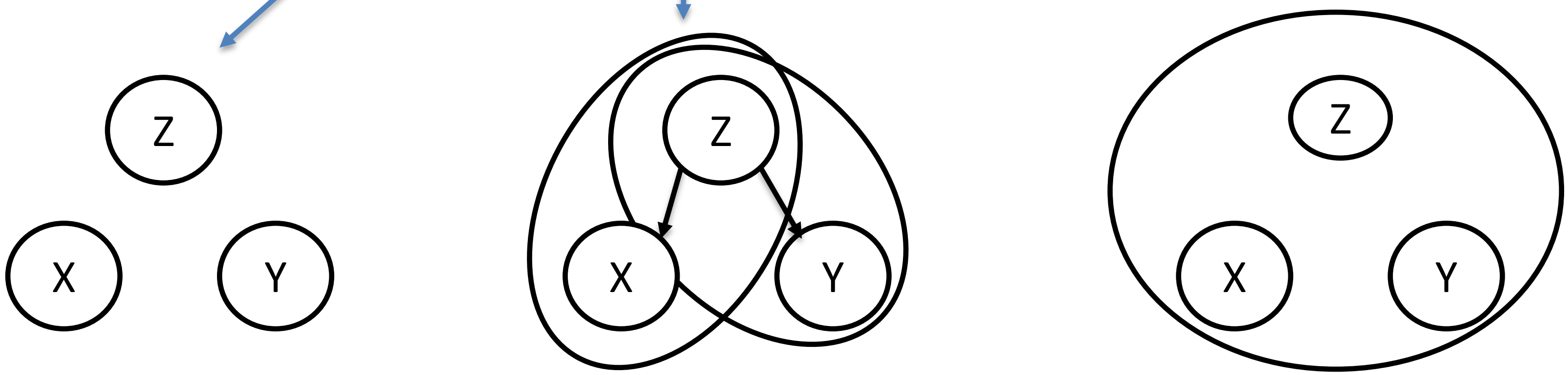
- Many DBMS supports Multi-dimensional histograms (e.g., PostgreSQL) but not by default
- Memory and inference overhead is $O(nbins^d)$
  - d is the number of dimensions (columns)
- Generally unaffordable when d is large (e.g. d > 2) even with modern histogram compression techniques
- What about filters on more (>2) attributes that are correlated? → still very inaccurate.

Probabilistic Graphical Models

Won't be on quiz

Table R with three attribute X, Y, Z

Filter on R with X = 0 and Y < 0 and Z > 5



P(X)*P(Y)*P(Z)

Independence Assumption

O(nbins)

P(X|Z)*P(Y|Z)*P(Z)

Conditional Independence Assumption

$O(nbins^2)$

2D histograms

P(X, Z, Y)

No assumption

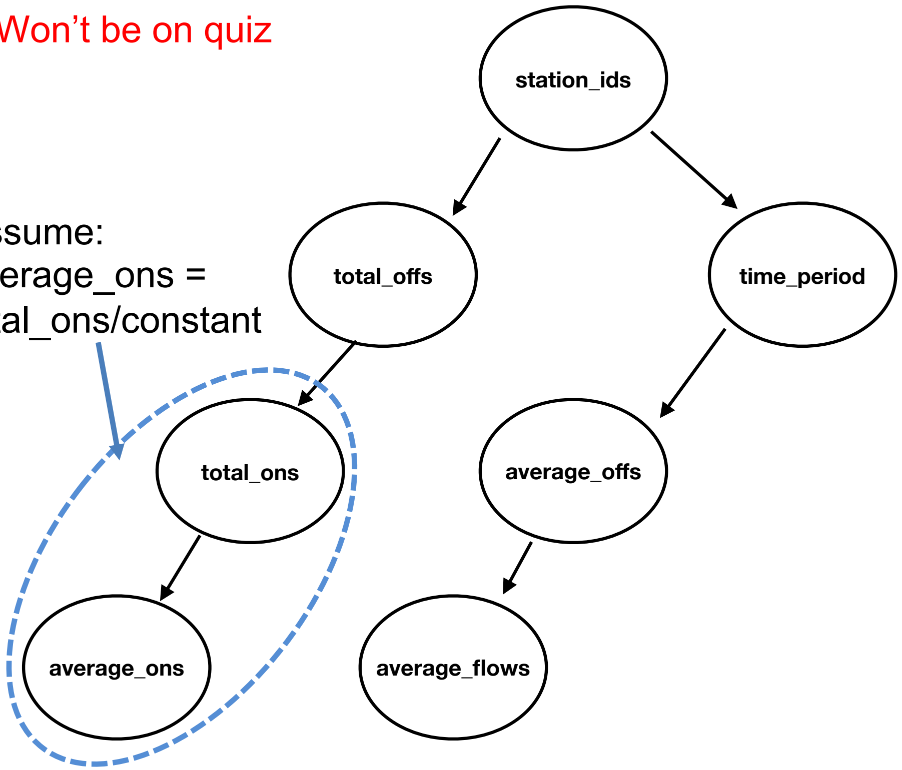
$O(nbins^3)$

May not be affordable

Probabilistic Graphical Models

Won't be on quiz

Assume:
average_ons =
total_ons/constant



Dependency graph (tree) of rail_ridership

Bayesian networks

Conditional independence assumption:

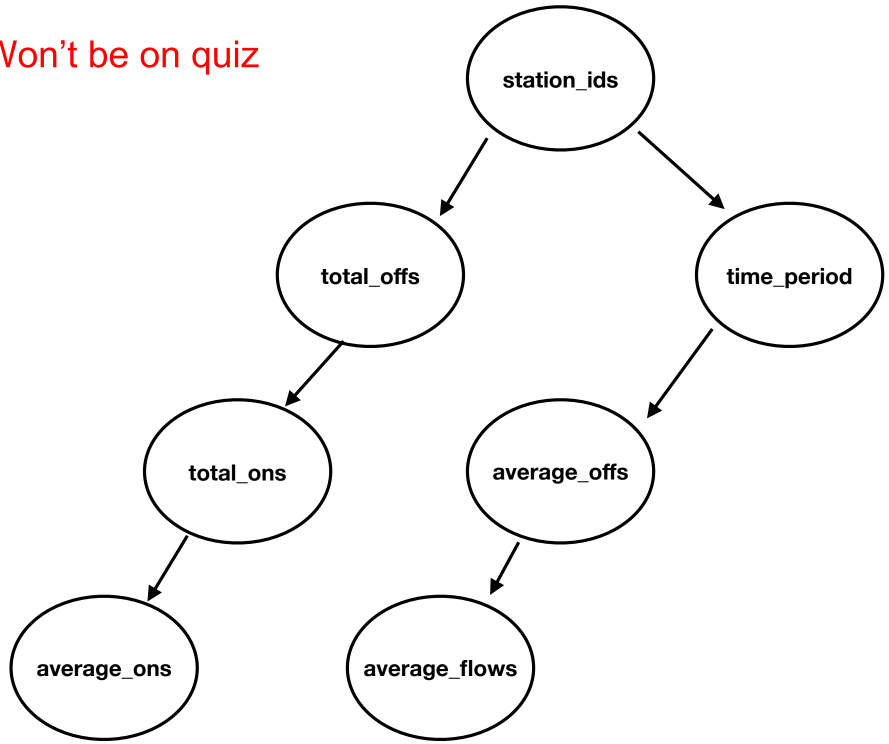
Given a dependency graph, an attribute is conditionally independent of all other attributes given its parent(s).

One 1-D histogram for each root (e.g. P(station_ids))

One 2-D histogram for each edge (e.g. P(average_ons|total_ons))

Probabilistic Graphical Models

Won't be on quiz



Dependency graph (tree) of rail_ridership

- PGMs provide a compact and accurate approach to build multiple 2-D histograms and use them for cardinality estimation.
- Each node will have at most one parent in a tree. $O(nbins^2)$ memory/inference complexity
- Tree-structured dependencies can preserve most correlations for many real-world data to provide accurate estimation on single table.
- A few DBMSes use PGM, such as ByConity from ByteDance

Chow, C. K.; Liu, C.N. (1968), "Approximating discrete probability distributions with dependence trees"
Lise Getoor and Daphne Taskar, Ben andKoller. (2001). "Selectivity estimation using probabilistic models"
Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, Jingren Zhou (2020), "Bayescard: Revitilizing bayesian frameworks for cardinality estimation"

Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances

Complex filter predicates

- String pattern matching
  - SELECT COUNT(*) from R WHERE X LIKE ‘%MIT%’;

- Complex mathematical expressions
  - SELECT COUNT(*) from R WHERE SQRT(X*Y) – Z*3 > 0;

- User Defined Functions
  - SELECT COUNT(*) from R WHERE my_hash(X) = 0;

Complex filter predicates

- Cannot use histograms to estimate them
- Most DBMSes just assume some constant selectivity (e.g. 7%) for these predicates.
  - Can still use histograms on other predicates
- Sampling as cardinality estimation
  - Keep a sample (e.g. 1%) of R in memory
  - Run the filter on this sample
  - Pros: works for any filters
  - Cons: very expensive / or not accurate

Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances

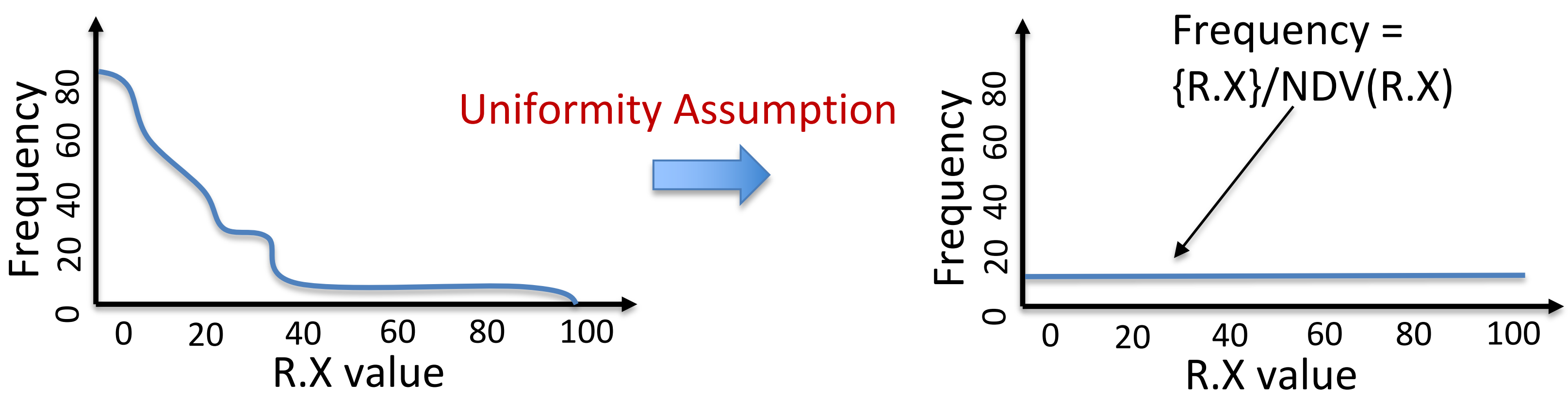
Estimating Join Cardinality

- Arguably the most crucial and most challenging part of query optimization
  - Good plans may execute in couple seconds while bad plans may execute for weeks.
  - Each join pattern imposes a unique data distribution and attribute correlation
- Objective for a desirable method:
  - Accurate
  - Lightweight (fast build time, low memory overhead)
  - Fast (low inference overhead)

Uniformity assumption

Assume all join keys are uniformly distributed

e.g. {R} = 500, NDV(R.X) = 100, so each value repeats exactly 5 times (number of distinct values)



Uniformity assumption

{R} = 500, NDV(R.X) = 100, so each value repeats exactly 5 times

{S} = 1000, NDV(S.Y) = 500, so each value repeats 2 times

At most how many unique values can there be in the result of the inner join R.X ⋈ S.Y? Min(100, 500) = 100

How many times can a value repeat in the result of the inner join R.X ⋈ S.Y? 5 * 2 = 10

^Card(R.X ⋈ S.Y)? 100 * 10 = 1000

$$\hat{}\mathrm{Card}(R.X \bowtie S.Y) = \underbrace{\min(\mathrm{NDV}(R.X)\,,\ \mathrm{NDV}(S.Y))}_{\text{Num. of distinct values in join result}} * \underbrace{\{R.X\}/\mathrm{NDV}(R.X) * \{S.Y\}/\mathrm{NDV}(S.Y)}_{\text{Each value will have this many repeats}}$$

^Card(R.X ⋈ S.Y) = {R} * {S} / max(NDV(R.X), NDV(S.Y)) (Lecture 5)

https://clicker.mit.edu/6.5830/ Uniformity assumption

Two tables R with {R} = 500, {S} = 1000, NDV(R.X) = 100, NDV(S.Y) = 500. Filter on R.A < 0 has selectivity of 20%. Filter on S.B > 0 has selectivity of 10%.

- Q1: What is ^Card(R.X $\bowtie$ S.Y AND R.A < 0 AND S.B > 0), under uniformity assumption?
- Q2: Suppose the actual cardinality is larger than your estimation, what is the maximumly possible estimation error? (in terms of Card/^Card)
  - (a) 1-10x underestimation
  - (b) 10-100x underestimation
  - (c) 100-1000x underestimation
  - (d) more than 1000x underestimation

We denote the estimation of Card(X) as ^Card(X)

Uniformity assumption

- Q1: What is ^Card(R.X $\bowtie$ S.Y AND R.A < 0 AND S.B > 0)?
  - 1000 x 0.1 x 0.2 = 20
- Q2: Suppose the actual cardinality is larger than your estimation, what is the maximum estimation error?
  - After R.A < 0, R will have 100 rows; after S.B > 0, S will have 100 rows
  - If distribution is highly skewed after the filter, max(card) = 100 * 100 = 10000
  - 500x estimation error

We denote the estimation of Card(X) as ^Card(X)

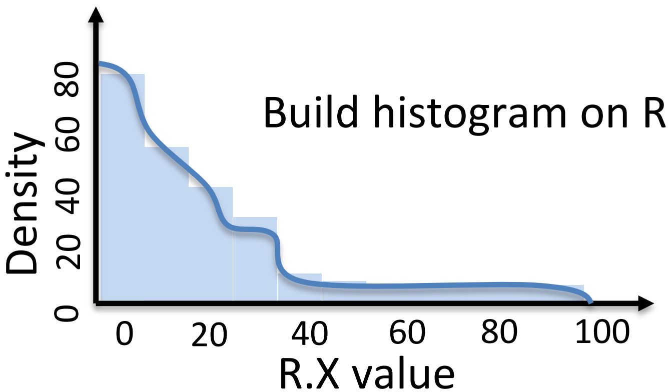
Uniformity assumption

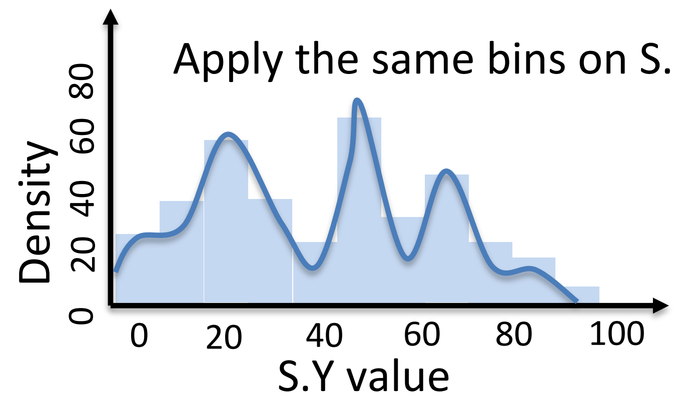
- Most DBMSes use this assumption
- Pros: lightweight, fast
  - #distinct can be read-off from index (if available)
  - Negligible memory/computation overhead
- Cons: very inaccurate
  - Real-world data are highly-skewed
  - Error will accumulate exponentially w.r.t. number of tables

Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances

Joining Histograms





Bin No.	bin	Density of R.X	Density of S.Y	Density of join
0	0-10	80	30	240
1	10-20	50	40	200
2	20-30	40	60	240
3	30-40	30	40	120
…	…	…	…	…
10	90-100	2	10	2
-	All	-	-	887

Uniformity assumption only in each bin
80 * 30 / max(10, 10) = 240

Sum up all bins

Relaxed version of uniformity assumption

Yannis Ioannidis (2003), "The History of Histograms (abridged)"

Joining Histograms

- A few DBMSes use this approach (e.g., Oracle)
- More expensive but more accurate than join uniformity assumption.
- Drawbacks
  - Cannot account for correlation between filtered attributes and join keys.
  - The same bins must be applied to the join keys
    - A set of bins that works well on R.X may not be optimal for S.Y

Roadmap

- Estimating the cardinality on single table
  - Histograms (used by PostgreSQL)
  - Handling correlated columns
  - Special filter types and estimation methods
- Estimating cardinality of joins
  - Uniformity assumption
  - Joining histograms
  - Recent advances (optional content)

Recent advances in cardinality estimation

Won't be on quiz

- Very active ongoing field of research
  - ~20 papers in SIGMOD/VLDB per year in the last 5 years.
- Two directions
  - Data-driven: build stats by analyzing the data
    - Everything you have seen so far
    - Use sophisticated statistical/ML models to understand distribution
  - Query-driven: do not analyze data, analyze query
    - Map query to its actual cardinality from execution feedback
    - Featurize the query and use ML/DL-based regression models

Data-driven: Denormalize

Won't be on quiz

- Join tables together (denormalize) and treat the denormalized result as a single table

R.X	R.A
1	12
2	62
2	-1
3	-7
4	99
…	…

$\bowtie$

S.Y	S.B
1	0.2
1	1.4
2	-9.1
2	-1.1
3	8.3
…	…

$\Rightarrow$

R.X	R.A	S.B
1	12	0.2
1	12	1.4
2	62	-9.1
2	62	-1.1
2	-1	-9.1
…	…	…

Can accurately answer any query such as (R.X $\bowtie$ S.Y AND R.A < 0 AND S.B > 0)

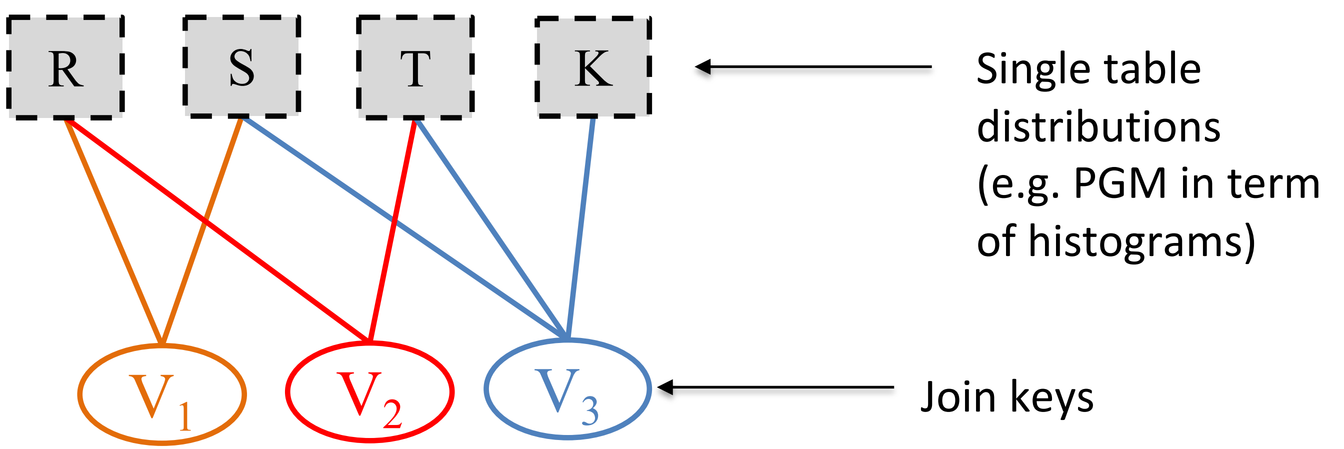
- Very accurate but very heavy-weight and slow
  - There can be exponential number of possible joins in a database with n tables
  - Need to understand the data distribution for each one

Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, Ion Stoica (2020), "NeuroCard: One Cardinality Estimator for All Tables"
Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, Carsten Binnig (2019), "DeepDB: Learn from Data, not from Queries!"
Rong Zhu*, Ziniu Wu*, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Bin Cui (2020), "FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation"
Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, Jingren Zhou (2020), "Bayescard: Revitilizing bayesian frameworks for cardinality estimation"

Data-driven: FactorJoin

Won't be on quiz

- Build a factor graph to generalize the joining histograms approach to accurately estimate any join with filters.
- Only need to understand the data distribution in each single table, combining single-table probabilities into probabilities on the denormalized (joined) tables using factor graph.



Factor graph rep. of a join graph

Ziniu Wu, Parimarjan Negi, Mohammad Alizadeh, Tim Kraska, Samuel Madden (2023), "FactorJoin: A New Cardinality Estimation Framework for Join Queries"

Query-driven

Won't be on quiz

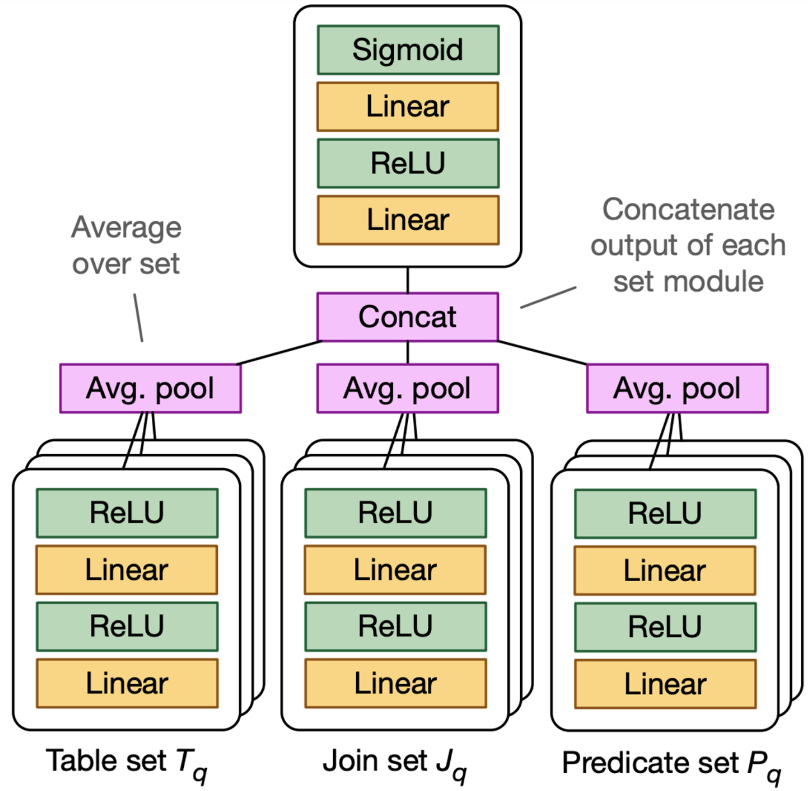
SELECT COUNT(*) FROM title t, movie_companies mc WHERE t.id = mc.movie_id AND t.production_year > 2010 AND mc.company_id = 5

Table set { [0 1 0 1 … 0], [0 0 1 0 … 1] } Join set { [0 0 1 0] } Predicate set { [1 0 0 0 0 1 0 0 0.72], [0 0 0 1 0 0 1 0 0.14] }

table id, samples, join id, column id, value, operator id

- Many DBMSes have execution history (with cardinality info)

- Featurize the queries (SQL)
- Train deep neural network to map query to its cardinality



Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, Alfons Kemper (2018), "Learned Cardinalities: Estimating Correlated Joins with Deep Learning"
Jie Liu, Wenqian Dong, Qingqing Zhou, and Dong Li (2021), "Fauce: fast and accurate deep ensembles with uncertainty for cardinality estimation"
Ji Sun, Guoliang Li (2019), "An end-to-end learning-based cost estimator"

Query-driven

Won't be on quiz

- Accuracy varies
  - Can be very accurate
  - Can be inaccurate if workload changes (training and testing queries mismatch) or data updates
- Can handle complex filter
  - Special query featurization for LIKE or user-defined functions
- Deep learning model can be expensive
  - Requires a large amount of training data
  - Large memory/computation overhead
  - Requires special hardware (e.g. gpu)

Summary

- Cardinality estimation is crucial and challenging
  - Simplified assumptions make this problem tractable and practical in DBMS, but can have huge estimation errors.
  - Advanced approaches makes it very accurate but more expensive to create/use.
  - Numerous ongoing research to find the sweet spot.