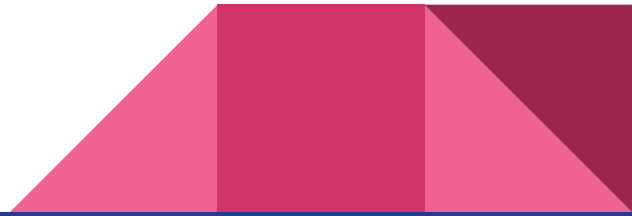
Approximate Aggregation

Outline

- Introduction to Approximate Query Processing(AQP)
- Related Work on AQP (over Joins)
- What about Non-Foreign-Key Joins?
- Weighted Sampling for AQP



Introduction to AQP

Approximate Query Processing

- Primarily for aggregate queries
  - count, sum, avg
- Exact answers are **NOT** *always* required
  - Exploratory queries in Desion Support Systems(DSS)
  - Interactive response times (in several seconds)
  - Big data visualization
  - Incomplete or noisy data
- Answering queries based on **samples**
  - Sampling can handle more general queries
  - Although other techniques exist: histograms, wavelets

```
SELECT sum(price)
FROM   orders
WHERE  region = 'Asia'

Exact: $12345

AQP  : $12300 ± 100
       with 95% confidence
```

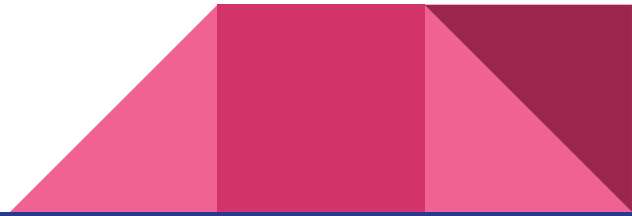
Sampling for AQP

- A relatively small random sample can well-represent the entire data
- Example: approximate answer for ***SUM()*** queries
  - e.g., S is a 1% sample of *orders*
  - Q: `SELECT sum(price) FROM orders WHERE region = 'Asia'`
  - X = the result of evaluating Q on S
  - Final answer = $(X * 100) \pm \Delta$ with $xx\%$ confidence
- Confidence Intervals
  - Most widely used: CLT-based confidence interval
  - Central Limit Theory
    - $\{X_1, X_2, \ldots, X_n\}$: a sequence of i.i.d. random variables
    - $E[X_i] = \mu$, $Var[X_i] = \sigma^2$, $S_n = (X_1+X_2+\ldots+X_n) / n$
    - $sqrt(n) * (S_n - \mu)$ converge to $N(0, \sigma^2)$

Related Work

Notation & Terminology

- Grouping/Aggregate Columns
  - e.g., Q: `SELECT sum(price) FROM orders GROUP BY region`
  - grouping column: ***region***, aggregate column: ***price***



Sampling Methods

- Stratified Sampling
  - divide the population into homogeneous subgroups
  - simple random sampling in each subgroup
- Cluster Sampling
  - divide the population into clusters
  - random sampling the clusters(, then the elements in them)
- PPS Sampling
  - probability-proportional-to-size sampling

$$\hat{\tau} = \frac{1}{n} \sum_{i=1}^{n} \frac{y_i}{p_i}, \quad var(\hat{\tau}) = \frac{1}{n} \sum_{i=1}^{N} p_i \left(\frac{y_i}{p_i} - \tau \right)^2$$

Cardinality Estimation

- CE is a traditional research area of DBMS
- Equivalent to $COUNT(*)$ queries in AQP
- Similar techniques
  - histogram, wavelet, sketch, sampling
  - sampling is not widely adopted due to its significant overhead
  - recent research shows the benefits of sampling in join size estimation

Uniform Sampling over Joins

- For hard cases, uniform sampling on either side is infeasible
- Attach weights based on the frequency of join attribute values
  - weights proportional to the frequency in the other side
- Consider $T_1 \bowtie T_2$
  - Let $F_i(a_j)$ be the frequency of join attribute value a_j in table T_i
  - S_1: weighted sampling on T_1
    - pick each (a_1) with probability proportional to $F_2(a_1) = 1$
    - pick (a_2) with probability proportional to $F_2(a_2) = n$
  - For each tuple in S_1:
    - choose one joinable tuple from T_2 randomly

T_1

A
a_1
a_1
...
a_1
a_2

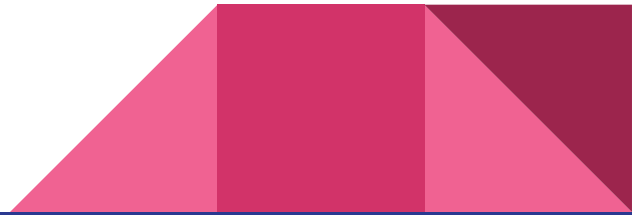
$\bowtie$

T_2

A	B
a_1	1
a_2	2
a_2	3
...	...
a_2	n+1

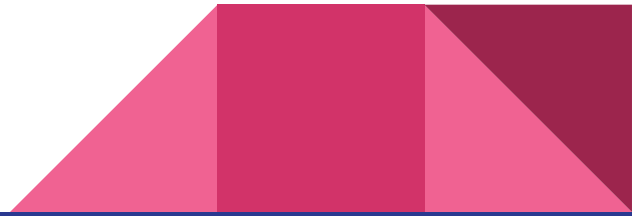
AQP over a single relation

- ***Value skew*** in aggregate columns
  - precompute ***SUM(column)*** and **scale factors** of tuples
  - outlier index
- ***Frequency skew*** among groups
  - congressional sampling: divide the sample size equally among groups
  - small group sampling: take all tuples of small groups into the sample
  - stratified sampling on grouping columns
- Highly selective filter conditions
  - workload-based weighted/stratified sampling
  - auxiliary index
- Non-correlated error among groups
  - precompute ***SUM(column)***



AQP over Joins

- Extend methods on a single relation to foreign-key joins
  - draw a sample from the fact table, then join it with the dimension table
- Draw simple random samples from both sides
  - ripple join
- Index-based correlated sampling
  - select random tuples of one side
  - sample tuples of the other side that can join with them
- Cluster sampling over join attributes
  - take simple random sample of join attribute values (using hash)
  - keep tuples whose join attribute values have been sampled



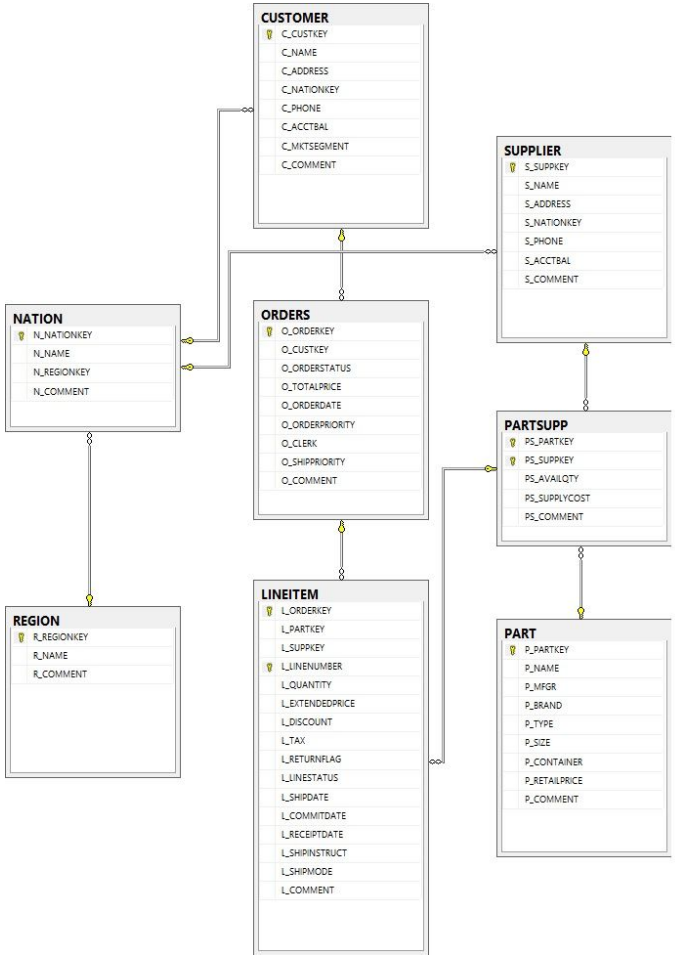
Non-Foreign-Key Joins?

Many-to-Many Joins

- Most joins are foreign-key joins (star/snowflake schema)
- Self-join

```
SELECT   avg(abs(A.salary - B.salary))
FROM     employee A
JOIN     employee B USING (department)
WHERE    A.id < B.id
GROUP BY department
```

- What are the ***semantics*** of aggregations after many-to-many joins?


CUSTOMER
C_CUSTKEY
C_NAME
C_ADDRESS
C_NATIONKEY
C_PHONE
C_ACCTBAL
C_MKTSEGMENT
C_COMMENT
SUPPLIER
S_SUPPKEY
S_NAME
S_ADDRESS
S_NATIONKEY
S_PHONE
S_ACCTBAL
S_COMMENT
NATION
N_NATIONKEY
N_NAME
N_REGIONKEY
N_COMMENT
ORDERS
O_ORDERKEY
O_CUSTKEY
O_ORDERSTATUS
O_TOTALPRICE
O_ORDERDATE
O_ORDERPRIORITY
O_CLERK
O_SHIPPRIORITY
O_COMMENT
PARTSUPP
PS_PARTKEY
PS_SUPPKEY
PS_AVAILQTY
PS_SUPPLYCOST
PS_COMMENT
REGION
R_REGIONKEY
R_NAME
R_COMMENT
LINEITEM
L_ORDERKEY
L_PARTKEY
L_SUPPKEY
L_LINENUMBER
L_QUANTITY
L_EXTENDEDPRICE
L_DISCOUNT
L_TAX
L_RETURNFLAG
L_LINESTATUS
L_SHIPDATE
L_COMMITDATE
L_RECEIPTDATE
L_SHIPINSTRUCT
L_SHIPMODE
L_COMMENT
PART
P_PARTKEY
P_NAME
P_MFGR
P_BRAND
P_TYPE
P_SIZE
P_CONTAINER
P_RETAILPRICE
P_COMMENT

Weighted Sampling for AQP

Value-Weighted Sampling

- Idea: ***SUM()*** can be converted to ***COUNT(*)***
  - e.g., ***SUM(b)***
  - convert table ***T*** into a virtual table T_v
  - for each tuple ***t***, repeat itself ***t.b*** times and set ***t.b = 1***

T

a	b
a1	1
a2	2
a3	3

T_v

a	b
a1	1
a2	1
a2	1
a3	1
a3	1
a3	1

Value-Weighted Sampling

- Idea: ***SUM()*** can be converted to ***COUNT(*)***
  - e.g., ***SUM(b)***
  - convert table ***T*** into a virtual table T_v
  - for each tuple ***t***: repeat ***t.b*** times & set ***t.b = 1***

- Weighted sampling in ***T***
  - equivalent to **uniform sampling in** T_v
  - attach each tuple ***t*** a weight proportional to ***t.b***

T

p	a	b
1/6	a1	1
2/6	a2	2
3/6	a3	3

→

T_v

a	b	p
a1	1	1/6
a2	1	1/6
a2	1	1/6
a3	1	1/6
a3	1	1/6
a3	1	1/6

Drawbacks

- Aggregate Expressions
  - ***sum(l_extendedprice * (1 - l_discount) * (1 + l_tax))***
  - column-stores may materialize these expressions
- Multi-aggregation

```
SELECT
     sum(l_quantity) as sum_qty,
     sum(l_extendedprice) as sum_base_price,
     sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
     sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
     avg(l_quantity) as avg_qty,
     avg(l_extendedprice) as avg_price,
     avg(l_discount) as avg_disc
FROM lineitem ...
```

- Negative values

Frequency-Weighted Sampling

- Consider $T_1 \bowtie T_2$
  - Let $F_i(a_j)$ be the frequency of join attribute value a_j in table T_i
  - S_1: weighted sampling on T_1
    - pick each (a_1) with probability proportional to $F_2(a_1) = 1$
    - pick (a_2) with probability proportional to $F_2(a_2) = n$
  - For each tuple in S_1:
    - choose one joinable tuple from T_2 randomly

T_1

A
a_1
a_1
...
a_1
a_2

$\bowtie$

T_2

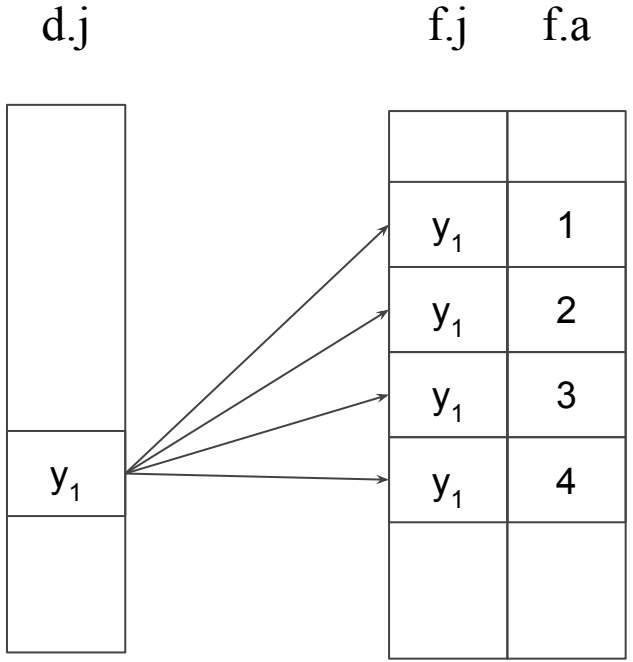
A	B
a_1	1
a_2	2
a_2	3
...	...
a_2	n+1

Value-Weighted & Frequency-Weighted

- Star/snowflake Schema
- Value-weighted: draw samples from the fact table
  - ***aggregate columns*** belong to the fact table
- Frequency-weighted: draw samples from dimension tables
  - fact → dimension: ***many-to-one***
- Frequency-weighted + value-weighted?
  - $\mathbf{t_d}$: dimension tuple, $\mathbf{f}$: fact table, $\mathbf{j}$: join attribute, $\mathbf{a}$: aggregate attribute

$$weight(t_d) = \sum_{i=0}^{F_f(t_d.j)} t_f^i.a$$

$$weight(t_f^i) = \frac{t_f^i.a}{weight(t_d)}$$



d.j	f.j	f.a
y_1	y_1	1
y_1	y_1	2
y_1	y_1	3
y_1	y_1	4

$$weight(t_d) = \sum_{i=0}^{F_f(t_d.j)} t^i_f.a$$

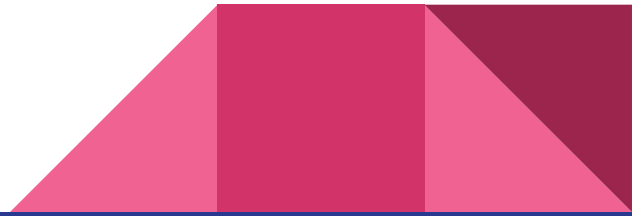
$$weight(t^i_f) = \frac{t^i_f.a}{weight(t_d)}$$

Cost Model for AQP?

- Uniform samples or weighted samples?
- Draw samples from fact tables or from dimension tables?
- Sampling-selection or selection-sampling?

```
SELECT sum(orders.price)
FROM   orders, customer
WHERE  orders.cid = customer.id
       AND customer.name = 'Bob'
```

- Sampling as an operator
- Factors: sample size, selectivity



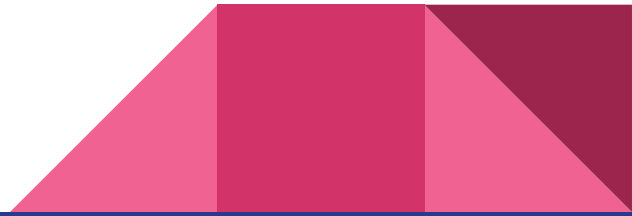
Backup Slides

Workload-Weighted Sampling

- Intuition: many queries only touch a portion of the data
  - latest events, constant predicts, …

```
SELECT count(*)
FROM   events AS ev
WHERE  ev.timestamp - current_timestamp
       < INTERVAL '24 hours'
```

- Idea: maintain a reservoir sample for ***latest seen*** tuples
- Idea: attach weights based on frequency of ***access*** to each tuple



Workload-Based AQP

- Analyze the given workload
- Maintain different types of samples
  - uniform sample, time-weighted sample, workload-weighted sample, …
- Classify incoming queries to use these samples

- Public Query Logs: Sloan Digital Sky Survey (www.sdss.org)
  - 4,000,000+ query logs per month
  - very few aggregate queries (0.01%) 😞