IID Sampling over Joins

Based on: Joins on Samples: A Theoretical Guide for Practitioners, PVLDB 2019

and

Random Sampling over Join Revisited, SIGMOD 2018.

Motivating Example

- Predicting the return flag of an item shipped to a customer
  - Using features of both the item and another item shipped to the same customer

Label | **Features**

Flag	CustId	Region	Total	Discount	Flag2	Total2	Discount2
1	10	2	100	0.2	0	20	0.5
0	20	1	200	0.0	0	100	0.1
0	20	1	500	0.1	0	300	0.2
……	……						

Motivating Example

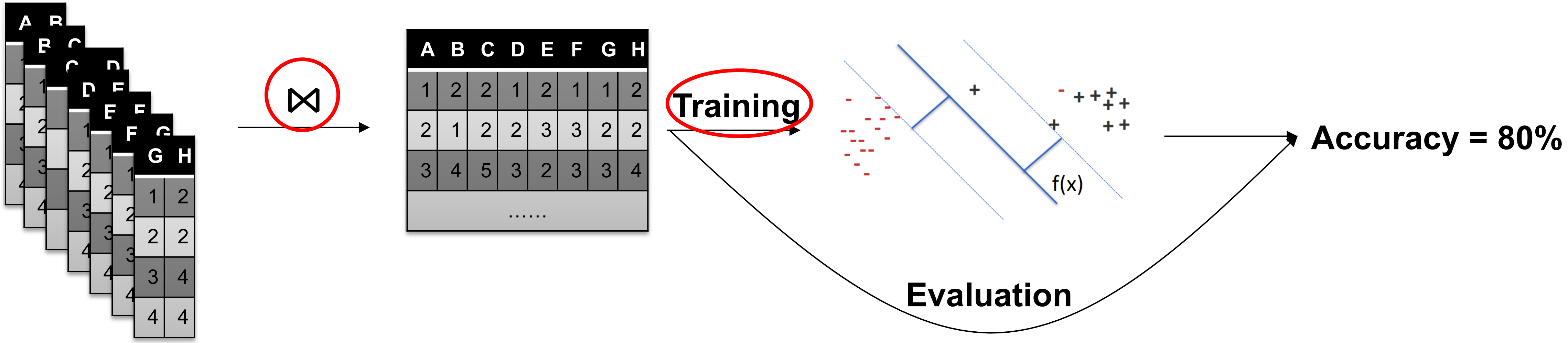
Joining 7 Tables from TPC-H

```
SELECT
    l1.l_returnflag, n_regionkey, s_acctbal,
    l1.l_quantity, l1.l_extendedprice, l1.l_discount,
    l1.l_shipdate, o1.o_totalprice, o1.o_orderpriority,
    l2.l_quantity, l2.l_extendedprice, l2.l_discount,
    l2.l_returnflag, l2.l_shipdate
FROM nation, supplier, lineitem l1, orders o1,
     customer, orders o2, lineitem l2
WHERE   s_nationkey = n_nationkey
    AND s_suppkey = l1.l_suppkey
    AND l1.l_orderkey = o1.o_orderkey
    AND o1.o_custkey = c_custkey
    AND c_custkey = o2.o_custkey
    AND o2.o_orderkey = l2.l_orderkey;
```

In order to predict the return_flag of an item ℓ1 shipped to a customer c, we may want to look at another item ℓ2 shipped to the same customer c and include the return_flag of ℓ2 as a feature

Motivating Example

- Training a classifier using SVM on a join over 7 tables
  - Full join takes more than 12 hours to compute.
  - Training runs forever without down-sampling.



I.I.D Sampling over Join

- In many applications a random sample of the join results often suffices
  - Estimating aggregates like COUNT, SUM, AVG, medians and quantiles, statistical inference, clustering, regression, classification, etc.
  - Training the model with a random sample on a join can bring great savings for both join computation and model training, while incurring a small and bounded loss in accuracy.

- Given two T_1 and T_2, a sampling algorithm A is iid, if tuples returned by A all have the same sampling probability and the appearance of two tuples in the join result are independent events.

Example: 2-table Join Sampling

A	B
1	1
2	1
3	1
4	2
5	2
6	2
7	3

B	C
2	1
2	2
3	3
3	4
3	5
3	6

A	B	C
4	2	1
4	2	2
5	2	1
5	2	2
6	2	1
6	2	2
7	3	3
7	3	4
7	3	5
7	3	6

$$\boldsymbol{R_1(A,B) \quad \bowtie \quad R_2(B,C) \quad = \quad R(A,B,C)}$$

Goal: sample $t \in R$ with probability $\frac{1}{10}$

Join Size

T1

A	B
a	1
b	2

T2

C	D
a	3
a	4
b	5
b	6

50% Uniform Sample

50% Uniform Sample

S1

A	B
a	1

$\bowtie$
A=C

S2

C	D
a	3
b	5

S1 $\bowtie$ S2

A	B	C	D
a	1	a	3

T1 $\bowtie$ T2

A	B	C	D
a	1	a	3
a	1	a	4
b	2	b	5
b	2	b	6

50% Uniform Sample

S(T1 $\bowtie$ T2)

A	B	C	D
a	1	a	3
b	2	b	5

Bernoulli/Random Sampling

- Offline setting
- Random sampling: for sample size k, each element in the underlying population is picked with equal probability; repeat k times independently. w/ or w/o replacement
  - Expensive for taking a large sample w/ replacement

- Join samples taken from tables based on Bernoulli sampling
- Bernoulli sampling: each tuple is included in the sample independently, with a fixed sampling probability p.
  - What join size do we expect?
  - Is the result a random/uniform sample?
  - Is the result an independent sample?

Bernoulli/Random Sampling

- Bernoulli sampling: each tuple is included in the sample independently, with a fixed sampling probability p.
  - p^2 of joined tuples. Quadratically fewer output tuples.
  - Uniform: Consider an arbitrary tuple of the join (t_1,t_2), where t_1 is from the first table and t_2 is from the second. The probability of this tuple appearing in the join of the samples is p^2.
  - Not independent: consider (t_1 , t'_2) where t'_2 joins with t_1. If (t_1 , t_2) in the output, the probability of (t_1 , t'_2) also appearing becomes p instead of p^2.

Universe Sampling

- Offline setting
- Given a column J, a (perfect) hash function $h : J \rightarrow [0, 1]$, and a sampling rate p, this strategy includes a tuple t in the sample if $h(t.J) \leq p$.
  - Often used for equi-joins (the same p value and hash function h are applied to the join columns in both tables). Why?
- What join size do we expect?
- Is the result a random/uniform sample?
- Is the result an independent sample?

Universe Sampling

- Given a column J, a (perfect) hash function $h : J \rightarrow [0, 1]$, and a sampling rate p, this strategy includes a tuple t in the table if $h(t.J) \leq p$.
  - Often used for equi-joins (the same p value and hash function h are applied to the join columns in both tables). Why?
- The join result size of two universe samples of rate p produces p fraction of the original join output *in expectation*.
- Uniform: each join tuple appears with the same probability p.
- Not Independent: Consider two join tuples (t_1, t_2) and (t'_1, t'_2) where t_1, t'_1, t_2, t'_2 all share the same join key. Then, if (t_1, t_2) appears, the probability of (t'_1, t'_2) also appearing will be 1. Likewise, if (t_1, t_2) does not appear, the probability of (t'_1, t'_2) appearing will be 0.

Stratified Sampling

- Offline setting
- The goal of stratified sampling is to ensure that minority groups are sufficiently represented in the sample.
- Groups are defined according to one or multiple columns, called the *stratified columns*. A group (a.k.a. a stratum) is a set of tuples that share the same value under those stratified columns.
- Given a set of stratified columns C and an integer parameter k, a stratified sampling guarantees at least k tuples are sampled uniformly at random from each group. When a group has fewer than k tuples, all of them are retained.

Sampling Summary

- The sampling operation cannot be pushed down through a join operator

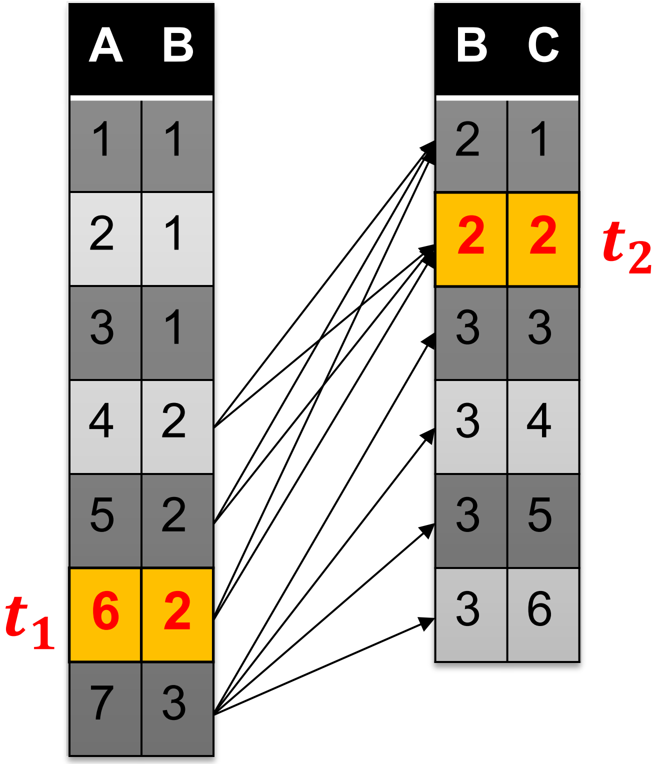
  sample(R) $\bowtie$ sample(S) $\neq$ sample(R $\bowtie$ S).

- Why iid sampling?

Join Sampling Requirements

- Online setting
- The problem of join sampling is to return each tuple from $J = R_1 \bowtie \cdots \bowtie R_n$ with probability $1/|J|$. When one sample is not enough, continuously sample until a desired sample size k is reached. Join sampling requires that these samples are totally independent.

Olken's Algorithm for 2-table Joins



$\boldsymbol{R_1(A,B)} \quad \bowtie \quad \boldsymbol{R_2(B,C)}$

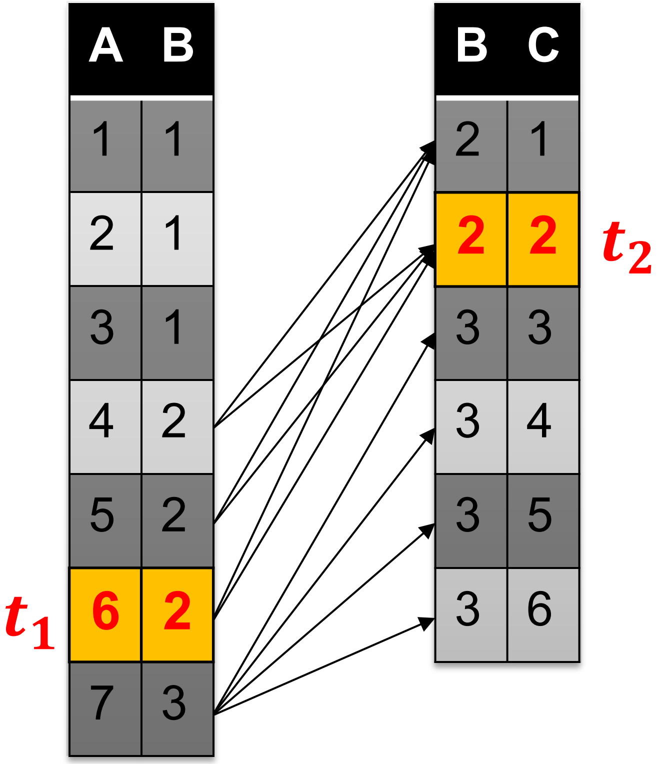
- Degree of value b in R_i: $d_B(b, R_i)$
- Maximum degree of B in R_i: $M_B(R_i)$

1. Uniformly sample $t_1 \in R_1$
2. Uniformly sample $t_2 \in t_1 \rtimes R_2 = \{t_2 \in R_2 | \pi_B R_2 = \pi_B(t_1)\}$
3. With probability, $\alpha = ?$ accept the sample. Reject otherwise. Show this algorithm guarantees iid.

$$\Pr(t_1, t_2 \wedge accepted) = \Pr(t_1) \quad \times \quad \Pr(t_2) \quad \times \quad \alpha \quad =$$

$$\Rightarrow \Pr(t_1, t_2 | accepted) = \frac{\Pr(t_1, t_2 \wedge accepted)}{\Pr(accepted)}$$

Olken's Algorithm for 2-table Joins



$\boldsymbol{R_1(A,B)} \quad \bowtie \quad \boldsymbol{R_2(B,C)}$

- Degree of value b in R_i: $d_B(b, R_i)$
- Maximum degree of B in R_i: $M_B(R_i)$

1. Uniformly sample $t_1 \in R_1$
2. Uniformly sample $t_2 \in t_1 \rtimes R_2 = \{t_2 \in R_2 | \pi_B R_2 = \pi_B(t_1)\}$
3. With probability $\alpha = \frac{d_B(\pi_B(t_1), R_2)}{M_B(R_2)}$, accept the sample.

   Reject otherwise. Show this algorithm guarantees iid.

High rejection rate if $M_B(R_i)$ is much larger than typical $d_B(b, R_i)$

$$\Pr(t_1, t_2 \wedge accepted) = \Pr(t_1) \times \Pr(t_2) \times \alpha = \frac{1}{7} \times \frac{1}{2} \times \frac{2}{4} = \frac{1}{28}$$

$$\Rightarrow \Pr(t_1, t_2 | accepted) = \frac{\Pr(t_1, t_2 \wedge accepted)}{\Pr(accepted)} = \frac{1/28}{10/28} = \frac{1}{10}$$

Chaudhuri et al.'s Algorithm for 2-table Joins

A	B
1	1
2	1
3	1
4	2
5	2
6	2
7	3

t_1 = (6, 2)

B	C
2	1
2	2
3	3
3	4
3	5
3	6

t_2 = (2, 2)

$\boldsymbol{R_1(A,B)} \quad \bowtie \quad \boldsymbol{R_2(B,C)}$

- Degree of value b in R_i: $d_B(b, R_i)$

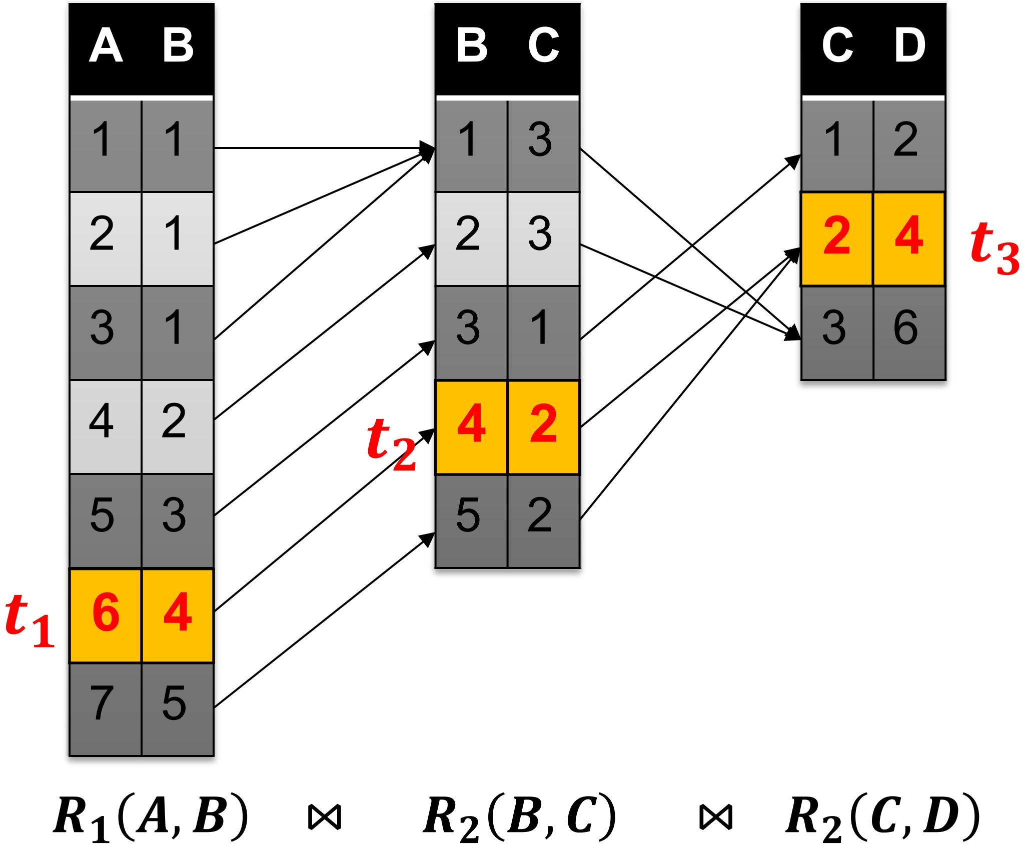
1. Sample $t_1 \in R_1$ with probability $\propto d_B(b, R_i)$
2. Uniformly sample $t_2 \in t_1 \rtimes R_2 = \{t_2 \in R_2 | \pi_B R_2 = \pi_B(t_1)\}$
3. Always accept the sample

Acceptance rate = 1

Both Olken's algorithm and Chaudhuri et al.'s algorithm can be implemented if indexes are available on the join attribute B. If not, a full scan on both relations is needed.

$$\Pr(t_1) \times \Pr(t_2) = \frac{2}{10} \times \frac{1}{2} = \frac{1}{10}$$

Acharya et al.’s Algorithm for Multi-way Foreign-key Joins



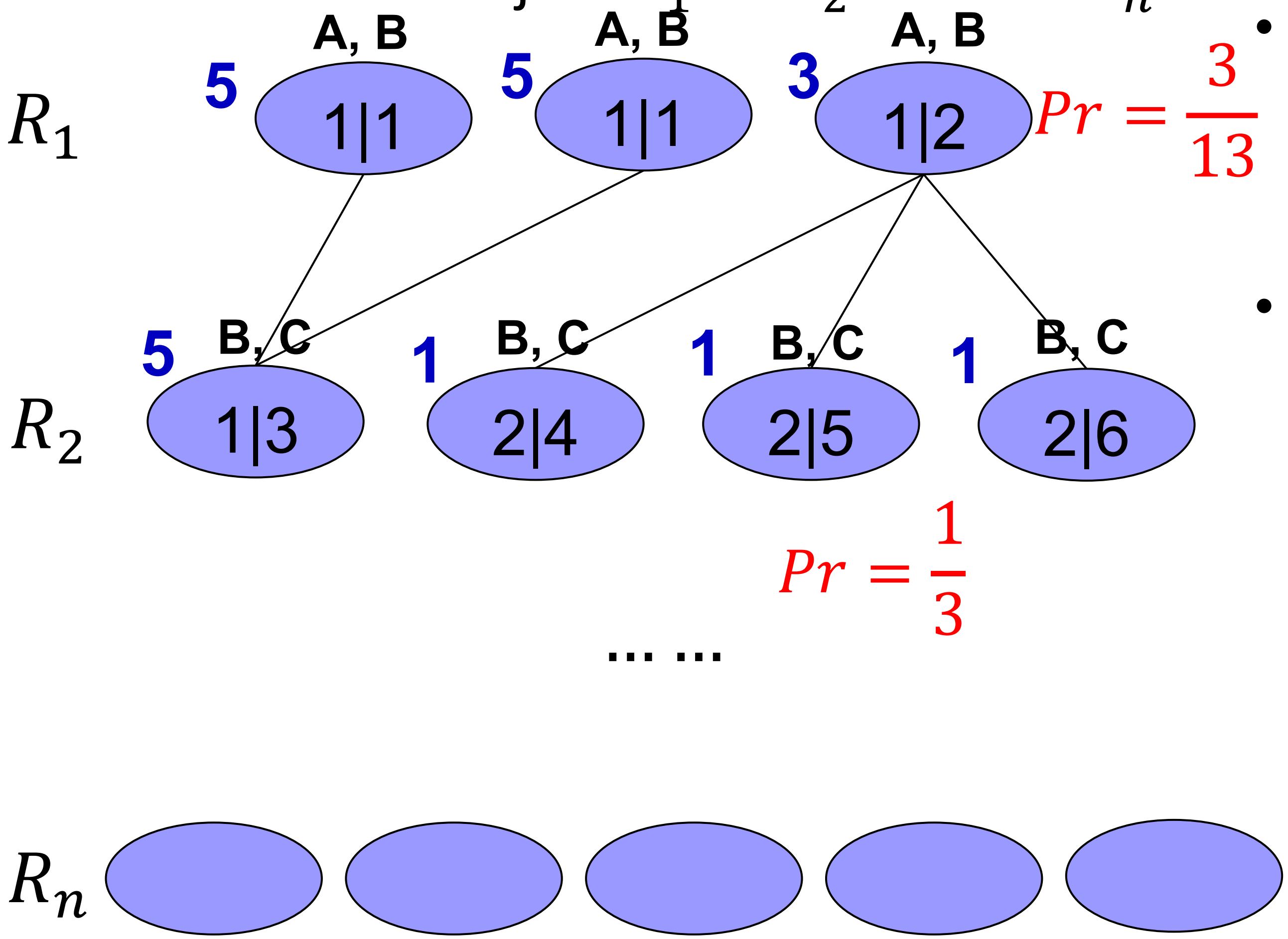
- Acyclic joins
- Joins are on foreign keys and primary keys

=> 1-to-1 mapping between $R_1 \bowtie R_2 \bowtie R_3$ and R_1

1. Uniformly sample $t_1 \in R_1$
2. Use the foreign key to look up matching tuples in $R_2, \ldots, R_n$

A General Sampling Framework for Multi-way Joins

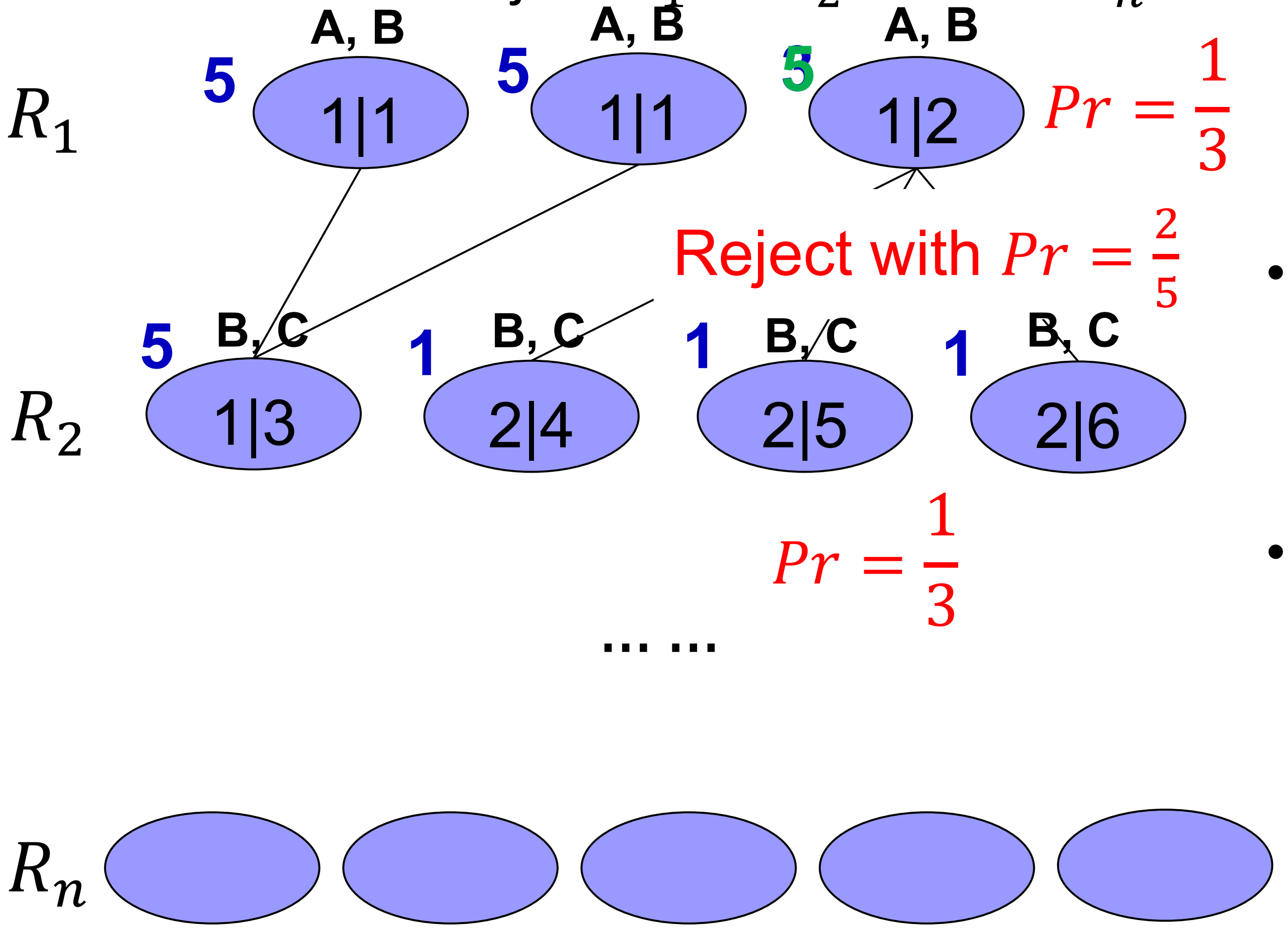
Consider a chain join $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$



- Model a join as a DAG
  - Vertices: tuples
  - Edges: if two tuples join
- Weight of a tuple $w(t)$: # join results starting from it
  - Sample proportional to weight

A General Sampling Framework for Multi-way Joins

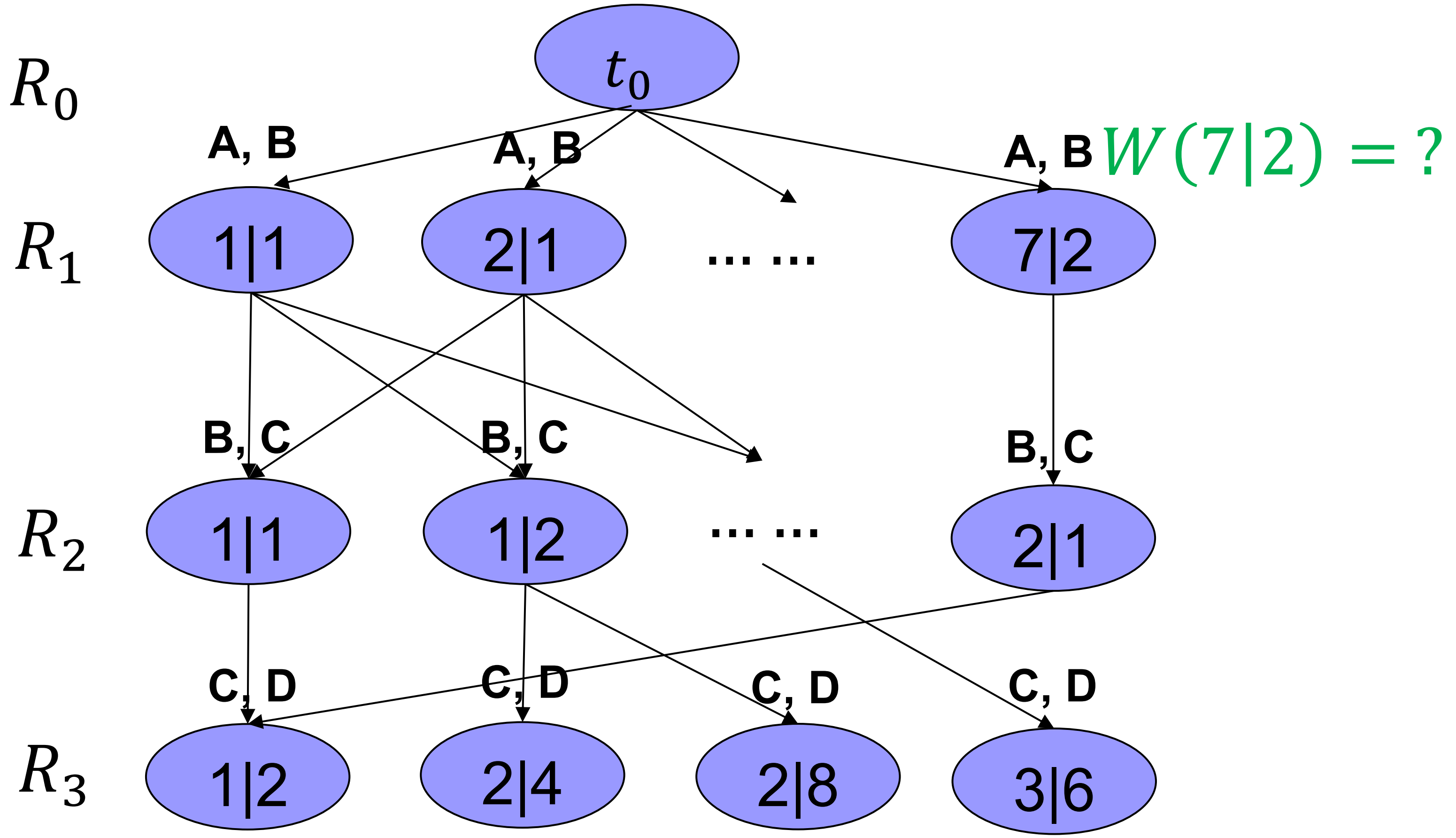
Consider a chain join $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$



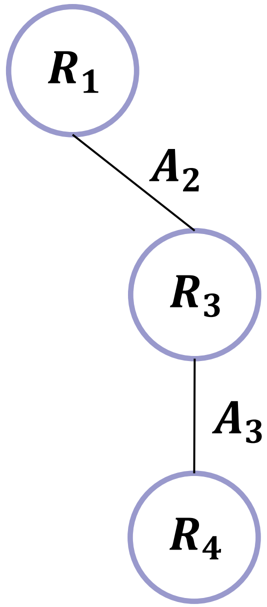
- We model join results as a DAG
  - Vertices: tuples
  - Edges: if two tuples join
- Weight of a tuple $w(t)$: # join results starting from it
  - Sample proportional to weight
- Use a surrogate of weight $W(t)$ if $w(t)$ is not available. $W(t)$: upper bound of $w(t)$
  - Reject with prob. $\frac{W(t) - \sum_{t' \in ch(t)} W(t')}{W(t)}$.

Instantiation of the Join Sampling Framework

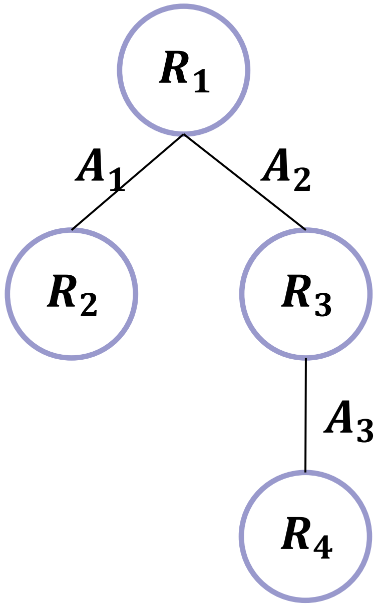
- Different instantiation of $W(t)$ => different sampling algorithms
  - How to efficiently compute a tight upper bound $W(t)$ for any tuple t in an online fashion?



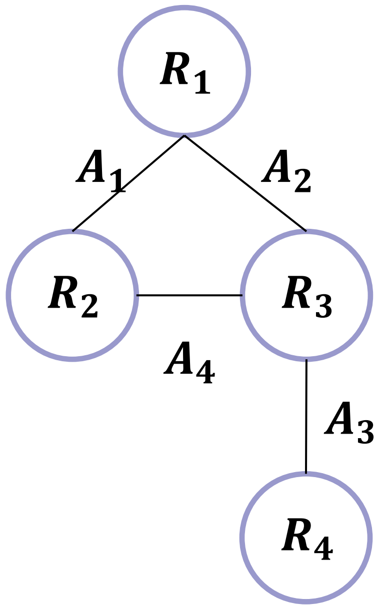
General Join Cases



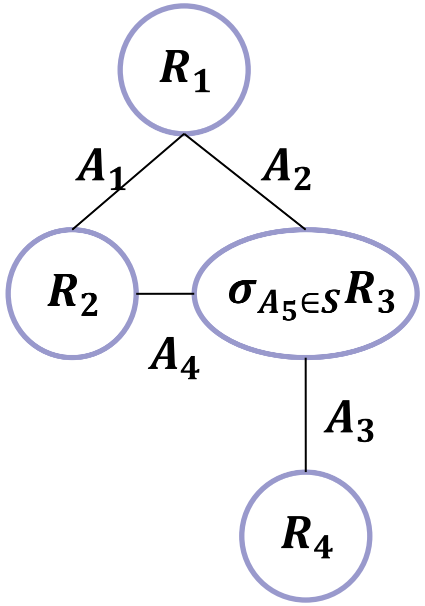
Chain Join



Acyclic Join



Cyclic Join



Join
w/ Selection Predicate

Project I

- Given sources $L = \{D_1,..., D_n\}$ with their costs $\{C_1,..., C_n\}$, and count requirements $\{Q_1, . . . , Q_m\}$ on groups $\{G_1, . . . , G_m\}$, our goal is to query different sources in L, in a sequential manner, in order to collect samples that fulfill the count requirement, while the expected total query cost is minimized.
- Generalize the problem to
  - fixed > 1 number of samples at each iteration
  - arbitrary number of samples at each iteration
  - count requirements on multiple groups (e.g. 100 of gender=F and 100 of gender=M as well as 100 of race=W and 100 of race=NW)
  - overlapping sources
- Prove of cost optimality when possible.
- Evaluate the designed algorithms in terms of cost/number of samples.
- Compare to a baseline/ existing work.

Project II

- We are given multiple (chain) join paths $J_1, \ldots, J_m$ with more than two tables, where each $J_i = T_1 \bowtie \ldots \bowtie T_k$. Note different join paths contain various number of tables. All join paths incur the same result schemas. Design an *efficient* algorithm for iid sampling from the union (set and multiset semantics) of $J_1, \ldots, J_m$. Suppose the following statistics are available/easy to compute.
  - Table sizes
  - The size of overlap of columns in table pairs
  - The join size of tables
- Prove the algorithm returns iid results.
- Empirically evaluate your algorithm in terms of efficiency and accuracy.
- Compare to a baseline/ existing work.
- https://github.com/InitialDLab/SampleJoin

Project III

- Literature review of threshold-based nearest neighbor search using containment
- Empirical evaluation of LSH Ensemble for containment search
- https://github.com/ekzhu/lshensemble
- Design complementary experiments to the paper to gain more insights.