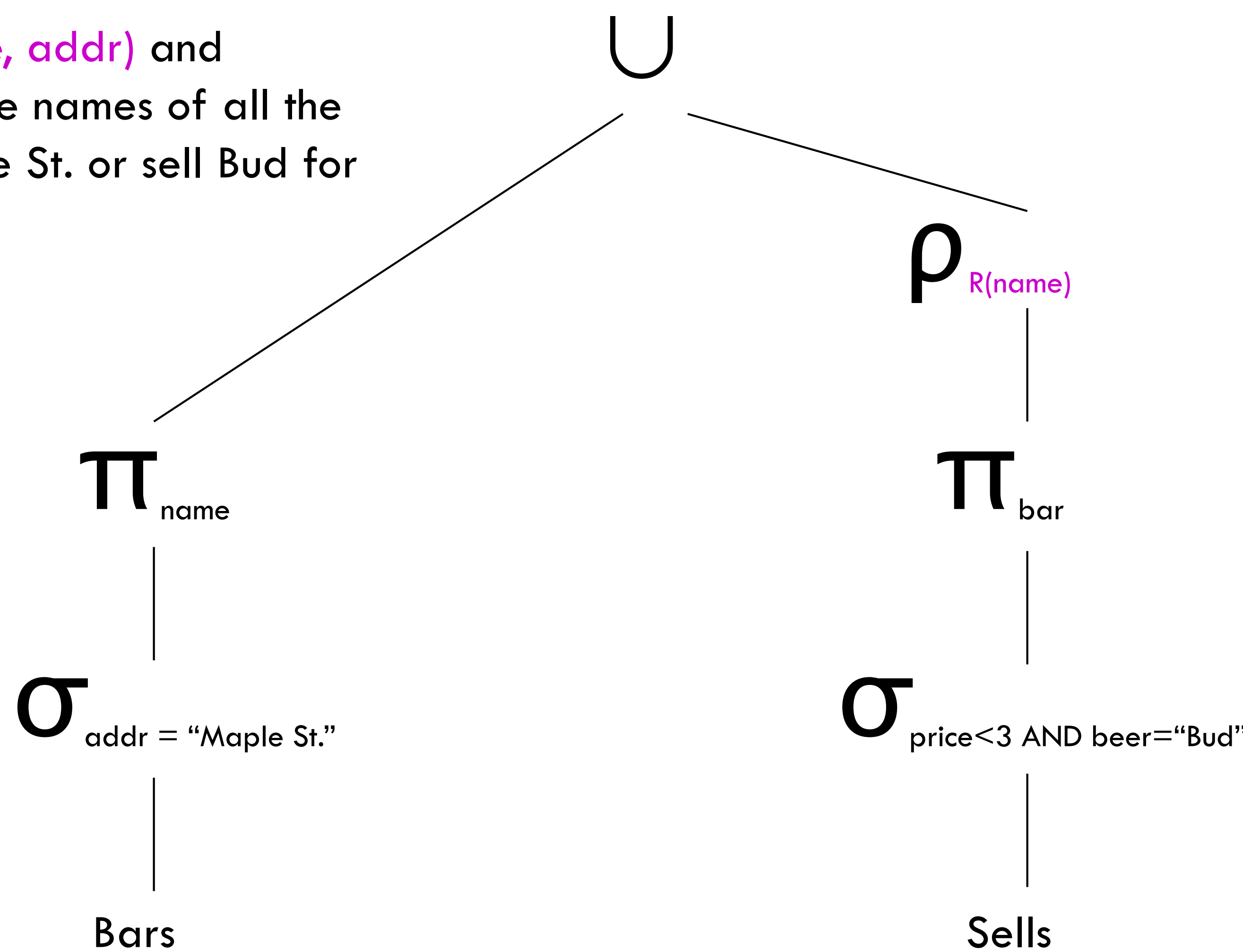
Example: Tree for a Query

- Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.

As a Tree:

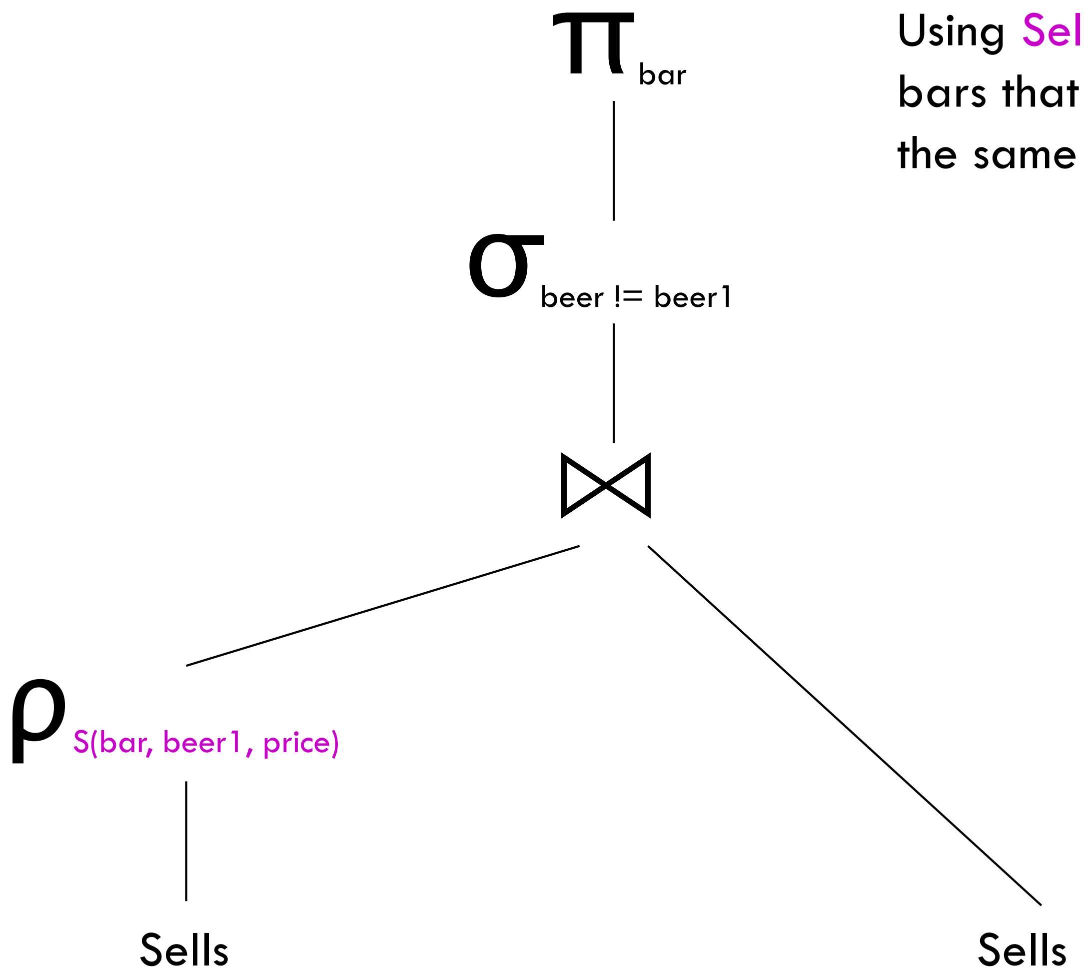
Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.



Example: Self-Join

- Using Sells(bar, beer, price), find the bars that sell two different beers at the same price.
- Strategy: by renaming, define a copy of Sells, called S(bar, beer1, price). The natural join of Sells and S consists of quadruples (bar, beer, beer1, price) such that the bar sells both beers at this price.

The Tree



Using Sells(bar, beer, price), find the bars that sell two different beers at the same price.

Schemas for Results

- Union, intersection, and difference: the schemas of the two operands must be the same, so use that schema for the result.
- Selection: schema of the result is the same as the schema of the operand.
- Projection: list of attributes tells us the schema.

Schemas for Results

- Product: schema is the attributes of both relations.
  - Distinguish two attributes with the same name.
- Theta-join: same as product.
- Natural join: union of the attributes of the two relations. Keep only one copy of the equated attributes.
- Renaming: the operator tells the schema.

Quiz 1

Assume set semantics

R

A	B
1	1
2	1
3	3

S

C	D
1	2
3	4
3	5

T

B	C
1	1
2	1
3	3

$S \bowtie T$

B	C	D
1	1	2
2	1	2
3	3	4
3	3	5

$\pi_B(R) \cap \rho_{T(B)}(\pi_C(S))$:

B
1
3

$R \bowtie (S \bowtie \rho_{T(B,C)}(R))$

A	B	C	D
1	1	1	2
2	1	1	2
3	3	3	4
3	3	3	5

The Extended Algebra

δ = eliminate duplicates from bags.

τ = sort tuples.

γ = grouping and aggregation.

Outerjoin : avoids "dangling tuples" = tuples that do not join with anything.

Duplicate Elimination

- R1 := δ(R2).
- R1 consists of one copy of each tuple that appears in R2 one or more times.

Example: Duplicate Elimination

R = (

A	B)
1	2
3	4
1	2

$\delta_{(R)}$ =

A	B
1	2
3	4

Sorting

- R1 := τ_L (R2).
  - L is a list of some of the attributes of R2.
- R1 is the list of tuples of R2 sorted first on the value of the first attribute on L, then on the second attribute of L, and so on.
  - Break ties arbitrarily.

Example: Sorting

R =

A	B
1	2
3	4
5	2

τ_B (R) =

A	B
5	2
1	2
3	4

Aggregation Operators

- Aggregation operators are not formally operators of relational algebra.
- Rather, they apply to entire columns of a table and produce a single result.
- The most important examples: SUM, AVG, COUNT, MIN, and MAX.

Example: Aggregation

R = (

A	B
1	3
3	4
3	2

)

SUM(A) = 7
COUNT(A) = 3
MAX(B) = 4
AVG(B) = 3

Grouping Operator

- R1 := γ_L (R2). *L* is a list of elements that are either:
  1. Individual (*grouping*) attributes.
  2. AGG(*A*), where AGG is one of the aggregation operators and *A* is an attribute.
     - An arrow and a new attribute name renames the component.

Applying $\gamma_L(R)$

- Group *R* according to all the grouping attributes on list *L*.
  - That is: form one group for each distinct list of values for those attributes in *R*.
- Within each group, compute AGG(*A*) for each aggregation on list *L*.
- Result has one tuple for each group:
  1. The grouping attributes and
  2. The group's aggregations.

Example: Grouping/Aggregation

R =

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

$\gamma_{A,B,AVG(C) \rightarrow X}$ (R) = ??

First, group *R* by *A* and *B* :

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

Then, average C within groups:

A	B	X
1	2	4
4	5	6

Recall: Outerjoin

- Suppose we join $R \bowtie_C S$.
- A tuple of R that has no tuple of S with which it joins is said to be *dangling*.
  - Similarly for a tuple of S.
- Outerjoin preserves dangling tuples by padding them NULL.

Example: Outerjoin

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

(1,2) joins with (2,3), but the other two tuples are dangling.

R FULL OUTERJOIN S =

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

Outer Join – Example

- instructor

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

teaches

ID	course_id
10101	CS-101
12121	FIN-201
76766	BIO-101

instructor $\bowtie$ *teaches*

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201

- Left Outer Join

  instructor $\bowtie_{\text{LEFT}}$ *teaches*

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	*NULL*

Outer Join – Example

- instructor

ID	*name*	*dept_name*
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

teaches

ID	*course_id*
10101	CS-101
12121	FIN-201
76766	BIO-101

- *instructor* $\bowtie_{\text{RIGHT}}$ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	null	null	BIO-101

- *instructor* $\bowtie_{\text{FULL}}$ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	*null*
76766	null	null	BIO-101

Operations on Bags

A **bag** = a set with repeated elements

All operations need to be defined carefully on bags

- $\{a,b,b,c\} \cup \{a,b,b,b,e,f,f\} = \{a,a,b,b,b,b,b,c,e,f,f\}$

- $\sigma_C(R)$: preserve the number of occurrences
- $\Pi_A(R)$: no duplicate elimination
- Cartesian product, join: no duplicate elimination

Important! Relational Engines work on bags, not sets!

Why Bags?

- SQL, the most important query language for relational databases, is actually a bag language.
- Some operations, like projection, are more efficient on bags than sets.

Operations on Bags

- Selection applies to each tuple, so its effect on bags is like its effect on sets.
- Projection also applies to each tuple, we do not eliminate duplicates.
- Products and joins are done on each pair of tuples, so duplicates in bags have no effect on how we operate.

Example: Bag Selection

R

A	B
1	2
5	6
1	2

$\sigma_{A+B<5}$ (R) =

A	B
1	2
1	2

Example: Bag Projection

R

A	B
1	2
5	6
1	2

$\pi_A(R) =$

A
1
5
1

Example: Bag Product

R

A	B
1	2
5	6
1	2

S

B	C
3	4
7	8

R X S =

A	R.B	S.B	C
1	2	3	4
1	2	7	8
5	6	3	4
5	6	7	8
1	2	3	4
1	2	7	8

Example: Bag Theta-Join

R(A, B)

A	B
1	2
5	6
1	2

S(B, C)

B	C
3	4
7	8

$R \bowtie_{R.B<S.B} S =$

A	R.B	S.B	C
1	2	3	4
1	2	7	8
5	6	7	8
1	2	3	4
1	2	7	8

Bag Union

- An element appears in the union of two bags the sum of the number of times it appears in each bag.
- Example: $\{1,2,1\} \cup \{1,1,2,3,1\} = \{1,1,1,1,1,2,2,3\}$

Bag Intersection

- An element appears in the intersection of two bags the minimum of the number of times it appears in either bag
- Example: $\{1,2,1,1\} \cap \{1,2,1,3\} = \{1,1,2\}$.

Bag Difference

- An element appears in the difference $A - B$ of bags as many times as it appears in A, minus the number of times it appears in B.
- Example: $\{1,2,1,1\} - \{1,2,3\} = \{1,1\}$.

Beware: Bag Laws != Set Laws

- Some, but *not all* algebraic laws that hold for sets also hold for bags.
- Example: the commutative law for union ($R \cup S = S \cup R$) *does* hold for bags.
  - Since addition is commutative, adding the number of times x appears in R and S doesn't depend on the order of R and S.

Example: A Law That Fails

- Set union is *idempotent*, meaning that $S \cup S = S$.
- However, for bags, if x appears n times in S, then it appears $2n$ times in $S \cup S$.
- Thus $S \cup S$!= S in general.
  - e.g., {1} ∪ {1} = {1,1} != {1}.

What about Intersection?

Intersection is idempotent for sets and bags.