A word now about **bags**

- Bag semantics are used in **select-from-where** statements
- However:
  - The default semantics for union, intersection and difference is set semantics
  - Meaning: duplicates are eliminated when these operations are applied.
- We can force semantics one way or the other!

A word now about **bags**

- Why bother with bags? or sets?
- Projection is faster if we can avoid eliminating duplicates
  - DBMS simply chugs through each of the tuples in turn, one at a time
  - Hence bags are used for projection.
- Intersection and difference are faster / more efficient if we sort beforehand
  - ... and if we have the data sorted, then throwing out duplicates is trivial.
  - Hence sets are used for intersection and difference

Eliminating / enabling duplicates

- To force result to be a set:
  - use **select distinct**
- To force the result to be a bag
  - use **all** with the set operator as in **union all**
- Two examples:
  - Find all the different prices charged for beers
  - List patrons who frequent a larger number of pubs than the number of beers that they like.

Example

```
select distinct price
from   sells;
```

Without **distinct** each price would be listed as many times as there were pubs / beers at that price.

```
(select patron from Frequents)
    except all
(select patron from Likes);
```

Without **all** we would would not be able to use tuple frequencies as required in the original question statement.

join expressions

- SQL provides several versions of bag joins
- These can be stand-alone expressions...
  - ... or they can be used in place of relations in a from clause
- Natural join: R **natural join** S
- Product: R **cross join** S
- Theta join: R **join** S **on** <condition>
- Note that R or S or both can be a subquery

Example

```
select * from Patrons
     join Frequents on name = patron;
```

Gives us all (name, address, phone, patron, pub) tuples such that a patron lives at the address and frequents the pub.

Some new RA operators

- δ
  - Eliminate duplicates from bags
- τ
  - Sort sets of tuples
- γ
  - Grouping and aggregation of tuples
- Outerjoin
  - Do not worry about the symbol – it is a little finnicky
  - This operation avoids **dangling tuples** (i.e., avoids having tuples that do not join with anything else)

Duplicate elimination

- Usage: R2 = δ(R1)
- R2 contains a single copy of each tuple appearing in R2 one or more times.

R1

A	B
1	2
3	4
1	2
3	2
3	2

R2 = δ (R1)

A	B
1	2
3	4
3	2

Sorting

- Usage: τ_L(R1)
  - L is a list of some of the attributes in R1
- Result is the list of R1's tuples...
  - ... sorted on the first attribute in L ...
  - ... then on the second attribute of L, etc.
  - Break ties arbitrarily
- τ is the only operator whose result is a list (i.e., not a set, not a bag)

R1

A	B
1	2
3	4
3	2

τ_B(R1) = [(3,2), (1,2), (3,4)]

Aggregation operators

- These are not really RA operators
- They apply to entire columns of a table...
  - ... and produce a single result
- Most obvious examples of such operators
  - sum
  - avg
  - count
  - min
  - max
- However, to apply the operators, we sometimes must specify a bit more about tuple groups themselves

Aggregation example

R1

A	B
1	3
3	4
3	2

- sum(A) = 7
- count(A) = 3
- max(B) = 4
- avg(B) = 3

Grouping operator

- Usage: R2 = γ_L(R1)
- L is a list of elements that are either:
  - **Individual** (i.e., grouping) attributes
  - **AgOp(A)** where AgOp is one of the aggregation operators and A is an attribute
- An **arrow** and a **new attribute name** provides a way to label the new resulting column/ attribute

Grouping operator: application of γ_L(R)

- Group R according to all the attributes listed in L
  - i.e., form one group for each distinct list of values for those attributes in R
- Within each group:
  - Compute AgOp(A) for each aggregation on list L
- Resulting relation has **one tuple for each group**
  - The grouping attributes plus
  - Their group's aggregate values

Example

R1

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

$$R2 = \gamma_{A,\, B,\, AVG(C) \rightarrow X}(R1)$$

First group R1 by A and B

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

Then average C within groups

R2 =

A	B	X
1	2	4
4	5	6