LAB 4 Notes

The Relational Algebra:

- Any questions on the project (Discuss)?
- In the previous lab we discussed the Conceptual Database Design Phase and the ER Diagram
- Today we will mainly discuss how to convert an ER model into the Relational model of a specific database.

Relational Algebra Operators:

1. Selection

The **selection** operation selects tuples from a relation that fit some criteria, creating a new relation with the selected tuples. We will use the notation

$\sigma_c(R) = \{ t \mid C \text{ is true for } t \}$

where σ is the selection operator, ***C*** is the **selection condition**, and ***R*** is a relation. The selection condition is a well-formed logical expression built from the following rules:

- a comparison operation between attribute names or attribute values, and
- the standard logical connectives: **AND**, **OR**, and **NOT**.

Some example conditions are given below for a relation with `Name` and `Age` attributes.

```
Name = 'Sue'
Name = 'Sue' AND Age > 23
NOT (Name = 'Sue' AND Age > 23)
```

Let's look at some example of selection, and the meaning will become clear. Consider the relation `Professions`.

Professions

```
Name | Job
------------------
Joe  | Garbageman
Sue  | Doctor
Joe  | Surfer
```

Now consider the following selections and their results.

$\sigma_{\texttt{Name = Sue}}(\texttt{Professions}) = \{t \mid t.\text{Name} = \text{Sue}\} = \{(\text{Sue, Doctor})\}$

$\sigma_{\texttt{Name = Sue}\ \textbf{OR}\ \texttt{Job = Surfer}}(\texttt{Professions}) = \{t \mid t.\text{Name} = \text{Sue}\ \textbf{OR}\ \$t.\text{Job} = \text{Surfer}\} = \{(\text{Sue, Doctor}), (\text{Joe, Surfer})\}$

$\sigma_{\texttt{Name = Sue}\ \textbf{AND}\ \texttt{Job = Surfer}}(\texttt{Professions}) = \{t \mid t.\text{Name} = \text{Sue}\ \textbf{AND}\ t.\text{Job} = \text{Surfer}\} = \{\}$

What does selection do in terms of the table metaphor? It merely selects those rows from the table that satisfy the selection condition, ignoring the rest. Note that the selected rows form a new table (possibly an empty table).

2. Projection

The **projection** operation projects out a list of attributes from a relation. For example, suppose we have a relation with the schema $R(A_1, A_2, ..., A_N)$ and we want only the first M attributes

$\pi_{A1, A2, ..., AM}(R) = \{ (t[A1], t[A2], ... t[AM]) \mid t \in R \}$

where π is the projection operator, *A1*, *A2*, ..., *AM* is a list of the first ***M*** attributes, and ***R*** is a relation. In general, we can project any of the attributes in a relation in any order. Let's look at some examples from the `Professions` relation depicted above.

$\pi_{\texttt{Job}}(\texttt{Professions})$ would produce the following relation.

```
Job
--------------
Garbageman
Doctor
Surfer
```

$\pi_{\texttt{Name}}(\texttt{Professions})$ would produce the following relation (assuming we retain duplicates)

```
Name
--------
Joe
Sue
Joe
```

or this table (assuming we eliminate duplicates)

```
Name
--------
Joe
Sue
```

3. Cartesian product of relations

The **Cartesian product** operation is similar to that for sets. Basically the Cartesian product produces a relation consisting of all possible pairings of tuples as follows. Assume we have relations $R(A_1, A_2, ..., A_N)$ and $S(B_1, B_2, ..., B_M)$ Then

$R \times S = \{((a1, a2, ..., aN, b1, b2, ..., bM) \mid (a1, a2, ..., aN) \in R$ **AND** $(b1, b2, ..., bM) \in S \}$

Note that $R \times S$ is not the same as $S \times R$ because the order of attributes differs. Let's look at an example. Assume that in addition to the `Professions` relation, we have a `Salaries` relation.

Salaries

```
Job         | Pays
------------------------
Garbageman | 50000
Doctor     | 40000
Surfer     | 6500
```

The result of `Professions` × `Salaries` is depicted below.

```
Name | Job         | Job         | Pays
-------------------------------------------
Joe  | Garbageman | Garbageman | 50000
Joe  | Garbageman | Doctor     | 40000
Joe  | Garbageman | Surfer     | 6500
Sue  | Doctor     | Garbageman | 50000
Sue  | Doctor     | Doctor     | 40000
Sue  | Doctor     | Surfer     | 6500
Joe  | Surfer     | Garbageman | 50000
Joe  | Surfer     | Doctor     | 40000
Joe  | Surfer     | Surfer     | 6500
```

Note that we have two attributes now with the same name, `Job`, we will assume that one of the attributes is renamed appropriately.

4. Union, Intersection, Difference

Since a relation is just a set (or multiset), the set (or multiset) algebra operations, **union**, **intersection**,

and **difference**, are also present in the relational algebra, with one constraint. These operations are only permitted between relations that are **union compatible**. Two relations are union compatible if they have the same number of attributes, and if the ith attribute in each relation has the same domain. Basically, the two relations must have the same schemas, modulo renaming of the attributes, which makes a lot of sense since you really do not want two completely different kinds of tuples in the same relation.

A complete set of operations

We now have a complete set of relational algebra operations. Any other operator that we might introduce, such as a *join*, is merely for our notational convenience.

Joins:

In general, a **join** is an operation that glues relations together. There are several kinds of joins.

5. Theta-join

The **theta-join** operation is the most general join operation. We can define theta-join in terms of the operations that we are familiar with already.

$R \bowtie_\theta S = \sigma_\theta(R \times S)$

So the join of two relations results in a subset of the Cartesian product of those relations. Which subset is determined by the **join condition** θ. Let's look at an example. The result of

$\texttt{Professions} \bowtie_{\texttt{Job = Job}} \texttt{Careers}$

is shown below.

```
Name | Job         | Job         | Pays
-------------------------------------------
Joe  | Garbageman | Garbageman | 50000
Sue  | Doctor     | Doctor     | 40000
Joe  | Surfer     | Surfer     | 6500
```

6. Equi-join

The join condition θ can be any well-formed logical expression, but usually it is just the conjunction of equality comparisions between pairs of attributes, one from each of the joined relations. This common case is called an **equi-join**. The example given

above is an example of an equi-join.

7. Natural join

Note that in the result of an equi-join, the join attributes are duplicated. A **natural join** is an equi-join that projects away duplicated $\bowtie$ attributes. If θ is omitted from a we will assume that the operation is a natural join. Let

$R = (A1,...,An,X1,...,Xm)$

and

$S = (X1,...,Xm,B1,...,Bk)$

Then

$R \bowtie S = \sigma_{A1,...,An,X1,...,Xm,B1,...,Bk}(R \bowtie_{X1 = X1 \textbf{ AND } ... \textbf{ AND } Xm = X1} S)$

(We assume that the join attributes have been made distinct via renaming appropriately.)

Let's look at an example. The result of

`Professions` $\bowtie$ `Careers`

is shown below.

```
Name | Job         | Pays
---------------------------------
Joe  | Garbageman | 50000
Sue  | Doctor     | 40000
Joe  | Surfer     | 6500
```