Notation and Terminology

Neville Churcher

Cosc265 2021

I **have had** requests for more detailed information on some of the notation that is used in this part of the course as not everybody is has encountered it elsewhere or is confident in its use. Here's my attempt at clarification. Keep in mind that notation is intended to help us by being concise and precise — not to make our lives harder. If anything's still not clear then please feel free to approach your tutors in labs — they are always happy to help.

Sets

Sets are widely used in the relational model. Schemas are sets of attributes, relations are sets of tuples, a minimal cover is a set of functional dependencies, and so on.

Here are some of the concept and operators we'll encounter:

Sets & elements We'll denote a set by listing its elements inside braces like this $MySet = \{A, B, C, D\}$. For convenience, if we're dealing with single-letter element names, we will often use the equivalent shorthand form $MySet = ABCD$. In both cases, the elements are unordered so $BDDA$ and $\{C, A, D, B\}$ are also equivalent.

We'll often encounter sets of functional dependencies $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ and relation schemes $R = ABCD$.

Exercise 1 (Realtions or tables?)
A relation is a set of tuples — so they don't have an order. When we line the tuples up to make a table, does it still behave like a set? If you perform the same `select` query several times, does your DBMS always return the resulting tuples in the same order?

Membership If A is a member of $MySet$ then we can write $A \in MySet$. Similarly, we write $E \notin MySet$ to say that E is not in $MySet$. We'll often say things like $f_i \in \mathcal{F}$or $X \notin K$.

Subsets & Supersets We often want to say that everything in one set, S_1, is also a member of another, S_2. We write $S_1 \subset S_2$ to denote that S_1 is a subset of S_2. Similarly, $S_2 \supset S_1$ — S_2 is a superset of S_1 — describes the same situation. In our work we often want to include the case where the subset includes all members — we write $S_1 \subseteq S_2$ to show that. For example, if we have a scheme $R = ABCDE$ and K is a key for R then we would write $K \subseteq R$ because the key might include all attributes.

Predicates Sometimes we want to define a set by specifying a condition for membership. The condition is called the *predicate*. We pronounce $\{x|P\}$ as "the set of x where P is true". If this reminds you of the SQL `SELECT` statement then hold that thought ...

Quantifiers

These are handy in a number of situations. They often turn up in relational calculus expressions, query predicates and definitions.

Universal quantifier ($\forall$) This is pronounced "for all". When we were defining functional dependencies $X \rightarrow Y$, we wanted to say "For all values, x that X could have, if we select the tuples of r where $X = x$, and then project out the Y column, then we'll have a relation with at most one tuple." We expressed that as: "$\forall x, \pi_Y(\sigma_{X=x}(r))$ has at most one tuple."

Existential quantifier ($\exists$) This is pronounced "there exists" — and we also have $\nexists$ for "doesn't exist". When we were defining functional dependencies $X \rightarrow Y$, we wanted to say "There are no two tuples in relation r that have the same values for attribute(s) X but different values for attribute(s) Y." We expressed that as: "$\nexists u, v \in r$ where $u[X] = v[X]$ and $u[Y] \neq v[Y]$."

Relational algebra

There's a lot that can be said about algebras — we'll leave that to the mathematicians. For our purposes, the main thing is that the relational algebra is closed under a number of operations. In other words, if we have some relations and we do things to them with combinations of the relational algebra operators, then the results will be relations.

Exercise 2
When an SQL query involving an aggregate function — such as `SELECT COUNT(*) FROM employee` — is executed, does your DBMS really give you the result as a relation?

We can consider the operators in several categories:

1. Some operations produce resulting relations with the same schema as their operands. Here r and s are *union compatible* — they have the same schemas.

   - σ Selection: $\sigma_c(r)$ has tuples of r matching condition c
   - $\cup$ Union: $r \cup s$ has tuples that are in r, in s or are in both.
   - $\cap$ Intersection: $r \cap s$ has tuples that are in *both* r and s
   - $-$ Difference: $r - s$ has tuples that are in r but not in s.

2. Some operations produce result relations which are "narrower" — have fewer attributes — than their operands.

   - π Projection: if $X \subset R$ then $\pi_X r$ has scheme X and may have fewer tuples than r since identical tuples will be removed.
   - $\div$ Division: If we have $q(Q)$ and $r(R)$ where $R \subseteq Q$ then $q \div r$ has scheme $Q - R$ and tuples that appear in q in combination with every tuple in r. Division can also be expressed using other relational operators.

3. Some operations produce "wider" relations. If we have $r(R)$ and $s(S)$ then:

   $\otimes$ Cartesian product: The tuples of $r \otimes s$ are all possible combinations of the tuples of the operands.

   $\bowtie$ You have encountered various forms of join (theta, equi, natural) elsewhere so we won't repeat them here.

4. There are some additional operators that can be handy at times.

   $\leftarrow$ Assignment: This is useful if we want to assign a value to a variable e.g. $r \leftarrow \sigma_{x<4^2}(s \bowtie p)$

   δ Renaming: There are two common situations where this is useful.

   - Distinguishing columns in joins where the same attribute name appeared in the original relations
   - Foreign keys: For example, we could say $Dog \leftarrow \delta_{P\#\leftarrow Owner}(Dog)$ to specify a more meaningful FK name.

Pictorial representations of most of these operators are included in the 'Set 1: Review' lecture notes on Learn.

Inference

- We often use the convention of using $\mathcal{F}$ to denote a set of functional dependencies.
- Given some FDs, we can infer others. We have seen how to do this using Armstrong's axioms as well as by using attribute closures. We use the symbol '$\models$' — pronounced "logically implies — when we want to say that a dependency can be inferred from one or more other dependencies. There are examples of this in the inference rules reference card on Learn.
- Closures, indicated by a superscript $^+$, have the feel of "keep going until you've got everything" about them. We've seen closures of sets of functional dependencies and we've also seen attribute closures.
- If $\mathcal{F}$ is a set of FDs, then we can talk about "the set of all FDs, f, which are logically implied by $\mathcal{F}$. We can express this as $\{f | \mathcal{F} \models f\}$ and we call it $\mathcal{F}$.
- Similarly, if X is an attribute, or set of attributes, then X^+ is the set of all attributes functionally determined by X using the FDs in $\mathcal{F}$.

Examples

Try going back through your notes and confirm that you can grok expressions like these:

- $A \rightarrow B \in \mathcal{F}$
- $\nexists Y \subseteq X, Y \rightarrow A_1 A_2 \ldots A_n \in \mathcal{F}^+$
- $\forall Z \subseteq Y, X \rightarrow Y \models X \rightarrow Z$
- $\forall i, i \in \{1 \ldots n\}, X \rightarrow A_i$ so $Y \subseteq X^+$