A Quick Introduction to Relational Algebra

Class Notes for COS 480/580 and MAT 500

Sudarshan S. Chawathe

University of Maine

October 2, 2006

Relational Databases For simplicity, we assume all constants in our database are drawn from a single underlying *domain* which we call $\mathcal{D}$. For concreteness, we will use $\mathcal{D} = \mathbb{R}$, the set of real numbers. A *k-tuple* is an element of the set $\mathcal{D} \times \mathcal{D} \times \ldots \mathcal{D}$, where the Cartesian product is taken $k - 1$ times. We say k is the *arity* of the tuple, and that the tuple has k *attributes*. A *relation* of *arity* k, or a k-ary relation, is a set of k-tuples. A relational database consists of a collection of *relations*. Relations are often depicted as tables, with tuples as rows and attributes as columns. For example, we may depict a 5-ary relation $R = \{(1,1,1,1,1), (1,1,2,1,2), (2,1,3,3,1)\}$ as follows:

1	1	1	1	1
1	1	2	1	2
2	1	3	3	1

Projection The projection operator is a unary operator that is parameterized by a list L of positive integers, and is denoted π_L. When applied to a k-ary relation R, we require the elements of L to be in the range $1, \ldots k$, with no duplicates. The result of applying π_L to R is the set of tuples formed by restricting the tuples in R to the attributes in L. For example, for a 5-ary relation R,

$$\pi_{1,2,4} R = \{(a,b,d) \mid \exists c, e : (a,b,c,d,e) \in R\}$$

Selection The selection operator is a unary operator that is parameterized by a *simple predicate* θ, and is denoted σ_θ. When applied to a k-ary relation R, the predicate θ is of the form $\alpha c \beta$ or $\alpha C \lambda$, where α and β are indices of attributes in R (thus $\alpha, \beta \in 1, 2, \ldots, k$), $\lambda \in \mathcal{D}$ is a literal, and C is a comparison operator over $\mathcal{D}$. For concreteness with $\mathcal{D} = \mathbb{R}$, we require C to be one of $=$, $<$, $>$, $\neq$, $\leq$, and $\geq$, with the usual semantics. The result of applying σ_θ to R is the set of tuples in R that satisfy the predicate θ. For example, for a 5-ary relation R,

$$\sigma_{\#2<5} = \{(a,b,c,d,e) \in R \mid b < 5\}$$

where we use the notational convention of prefixing attribute indices with the # character to distinguish them from literals.

Cross Product The cross product operator in relational algebra, $\times$, is a binary operator with a definition that is very similar to the standard definition of a cross product of two sets. The only difference is that we flatten the each ordered pair of tuples resulting from a cross product of two relations. For example, for a 5-ary relation R and a 3-ary relation S,

$$\begin{aligned} R \times S \quad = \quad & \{(a,b,c,d,e,f,g,h) \mid \\ & (a,b,c,d,e) \in R \wedge (f,g,h) \in S\} \end{aligned}$$

Union The union operator in relational algebra is identical to the standard set operator. It is a binary operator that can only be applied to operands that are *union compatible*: they must have the same arity and the corresponding attributes must have identical types. (The latter condition is always true with our assumption of a single domain $\mathcal{D}$.) For example, if S and T are two union-compatible relations (say, of arity 3), then their union is a 3-ary relation

$$S \cup T = \{s \mid s \in S \vee s \in T\}$$

Difference As with union, this relational-algebra operator is identical to the standard set difference operator. Its operands are also required to be union compatible. For example, with the relations S and T above, we have

$$S \cup T = \{s \mid s \in S \wedge s \notin T\}$$

Named Attributes Above, we have identified attributes using their indices (i.e., positions) within tuples and relations. For example, $\pi_{1,3,5}$ projects on to the first, third, and fifth attributes. It is often convenient to use a slightly different scheme in which attributes are assigned identifying names. For example, we may use the scheme $R(A, B, C, D, E)$ for the 5-ary relation R introduced earlier. In a graphical representation, it is conventional to use the attribute names as column headings:

A	B	C	D	E
1	1	1	1	1
1	1	2	1	2
2	1	3	3	1

The projection $\pi_{A,B,D}R$ is equivalent to the projection $\pi_{1,2,4}R$ introduced earlier. It is conventional to omit the commas from lists of attribute names when there is no danger of confusion; thus this projection may written more concisely as $\pi_{ABD}R$. We shall refer to this variant of the algebra as the *named algebra* and the earlier one as the *unnamed algebra*.

Renaming In the named algebra, it is sometimes necessary to rename attributes using the renaming operator $\rho_{\mathcal{S}}$, where the subscript $\mathcal{S}$ is a relational scheme (e.g., $T(P, Q, R)$). The arity of this scheme must be the same as the arity of the operand to which this operator is applied. For example, given $R(A, B, C, D, E)$, the following expression produces a relation with the scheme $T(P, Q, R, S, T)$ containing exactly the tuples in R:

$$\rho_{T(P,Q,R,S,T)}R$$

An example of the use of this operator appears in the description of the natural join below.

Derived Operators It is not difficult to prove that each of the six operators introduced above (projection, selection, cross product, union, difference, and renaming) is *basic*, in the sense that it is not expressible using any combination of the others. That is, it is not possible to achieve the effect of one of these operators using some combination of the others. In contrast, we may define any number of *derived* operators, which are operators defined using one or more of the basic operators. Below, we introduce two such derived operators: intersection and join.

Intersection The intersection of two relations that are union compatible is, as usual, the set of tuples that are contained in both:

$$S \cup T = \{s \mid |s \in S \wedge s \in T\}$$

An easy argument reveals that we may express intersection using difference:

$$S \cup T = S - (S - T) = T - (T - S)$$

Join Recall that the cross product of two relations is obtained by pairing every tuple from one relation with every tuple from the other. Intuitively, the join operator restricts such pairings to those that satisfy some predicate, called join predicate. More precisely, the *predicate join* (or *theta join*) $\bowtie_\theta$ is a binary operator that is parameterized by a predicate θ that is a selection predicate over the cross product of its operand:

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

Natural Join In the named algebra, it is often convenient to use a join predicate that equates like-named attributes of its operands and discards one attribute from each pair of like-named attributes. This operator is called the *natural join* and denoted by $\bowtie$. Thus the absence of a predicate below the $\bowtie$ symbol does not denote a vacuously satisfied predicate as one may expect; rather, it denotes a predicate that equates the like-named attributes of the operands. For example, with relation schemes $R(A, B, C, D, E)$ and $S(D, E, F)$,

$$R \bowtie S = \pi_{ABCDEF}\sigma_{R.D=S.D \,\wedge\, R.E=S.E}(R \times S)$$

The predicate of the select operator uses a dotted notation to disambiguate the like-named attributes. This notation is shorthand for the following, more precise, expression of the right-hand-side using the renaming operator:

$$\pi_{ABCDE}\sigma_{D=D' \wedge E=E'}(R \times \rho_{S'(D',E',F')}S)$$