Relational Model, Relational Algebra, and SQL

August 29, 2007

1 Relational Model

Data model. Set of conceptual tools for describing of data, data semantics, data relationships, and data integrity constraints.

Relational Model. A data model in which *relations* are used for all purposes!

Relation. A subset of $D_1 \times D_2 \ldots \times D_n$ where each D_i is a domain.

Each relation is a set of n-tuples $(a_1, \ldots, a_n)$ where $a_i \in D_i$.

We say that a relation, which is a subset of $D_1 \times D_2 \ldots \times D_n$, has n-attributes.

We associate a name to each attribute of a relation. The domain of an attribute is the set of possible values that it can take.

For n attributes $A_1, \ldots, A_n$, $R = (A_1, \ldots, A_n)$ is called a *relation schema*. A schema denotes a set of relations whose tuples are of the form $(a_1, \ldots, a_n)$ where a_i belongs to the domain of A_i. Each of these relations is called a relation on the schema R.

For a tuple $t = (v_1, \ldots, v_n)$ that belongs to the instance $r(R)$, $t[A_j]$ denotes v_j and $t[S]$, where $S \subseteq \{A_1, \ldots, A_n\}$, denotes a tuple created by the values of attributes belonging to S

$r(R)$ denotes a *relation instance* of R which is a relation on the schema

Key: (also called *superkey*) A set of attributes K of a schema R is a key if the values for K (or more precisely, the values for attributes in K) uniquely determined a tuple of each possible relation $r(R)$;

The definition of key also implies that no instance $r(R)$ could have two tuples whose values for K are identical.

Candidate key: a key which is minimal with respect to the set inclusion operator $\subseteq$.

Primary key: a designated candidate key.

Foreign key: a set of attributes of a relation that correspond to the primary key of another relation.

2 Database query language

Used for retrieving information stored in a database. Most important relational query language is SQL. SQL is declarative (only specifies what should be in the answer and not how to compute it). See the difference between SQL and C, C++?

Relational Algebra. An important form of query language for the relational model. Provided the necessary background for designing complex queries. Consists of a small number of basic operators. A query is simply an expression in relational algebra.

2.1 Basic operators of the relational algebra

They are divided into the following classes:

1. *Set operators*: union ($\cup$), intersection ($\cap$), and set difference ($-$)
2. *Operators that remove part of the relation*: projection or selection (π or σ)
3. *Operators that combine part of the relation*: Cartesian product or joint ($\times$ or $\rhd\lhd$)
4. *Renaming operators*: rename the schema of the relation (ρ)

Set operators

Union ($\cup$): $r \cup s$ represents a new relation whose tuples are either tuples belonging to r, to s, or both.

Set difference ($-$): $r - s$ represents a new relation whose tuples are tuples belonging to r but not to s.

Intersection ($\cap$) : $r \cap s$ represents a new relation whose tuples are tuples belonging to r and s.

Note: Set operators only work on relations with the same schema.

Example 1 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

and s be the following relation instance:

A	*B*	*C*
2	*3*	*4*
2	*1*	*1*
1	*3*	*4*
4	*3*	*4*

Then: $r \cup s$ is

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*
2	*3*	*4*
1	*3*	*4*
4	*3*	*4*

$r \cap s$ is

A	*B*	*C*
2	*1*	*1*

and $r - s$ is

A	*B*	*C*
1	*2*	*3*
3	*2*	*2*
4	*1*	*3*

NOTE: *If the schema of s changes, say $s(A1, B1, C1)$ then all of the three operations ($\cup$, $\cap$, $\setminus$) will yield the empty relation.*

Operators that remove part of the relation

Projection (π) This operation removes some columns from a relation. We write $\pi_{A_1,\ldots,A_k}(r)$ and the result is a new relation that has only the columns $A_1, \ldots, A_k$ of r and whose schema is the set of attributes $\{A_1, \ldots, A_k\}$ of the relation r. For the relation instance r in Example 1,

$$\pi_{A,B}(r)$$

is the relation instance:

A	B
1	2
2	1
3	2
4	1

Selection (σ) The operation selects some tuples from the current relation and defines a new relation. $\sigma_C(r)$ represents a new relation that has tuples belonging to r and each of them satisfying the conditional expression (or condition) C.

C is a conditional expression constructed from attributes of r or constants and/or basic logical operations such as AND, OR, and NOT. For example,

$$A < B + C$$

would be a valid conditional expression with respect to the relation r in Example 1; another example is

$$(A < B + C)\ OR\ B = C.$$

For the relation r in Example 1, $\sigma_{A<B+C}(r)$ is the following relation instance:

A	B	C
3	2	2

Operators that combine relations

Cartesian Product ($\times$) Given two relations r and s. The Cartesian product $r \times s$ is the set of pairs that can be formed by choosing the first element of the pair to be any element of r and the second an element of s.

Example 2 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

and s be the following relation instance:

E	*D*
2	*3*
2	*1*

Then: $r \times s$ is

A	*B*	*C*	*E*	*D*
1	*2*	*3*	*2*	*3*
2	*1*	*1*	*2*	*3*
3	*2*	*2*	*2*	*3*
4	*1*	*3*	*2*	*3*
1	*2*	*3*	*2*	*1*
2	*1*	*1*	*2*	*1*
3	*2*	*2*	*2*	*1*
4	*1*	*3*	*2*	*1*

NOTE: When r and s share some attributes, the name of the shared attributes *need* to be redefined. Two common ways: (i) attaching the relation name to the attributes of the result; (ii) using the renaming operator.

Suppose s is given by

A	D
2	3
2	1

Then: $r \times s$ is

r.A	B	C	s.A	D
1	2	3	2	3
2	1	1	2	3
3	2	2	2	3
4	1	3	2	3
1	2	3	2	1
2	1	1	2	1
3	2	2	2	1
4	1	3	2	1

Alternatively, we can write $r \times s[A, B, C, D, E]$ and the result is

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

2.2 Additional Relational Algebra Operations

Join (also called *theta* join) The join operator combines two relations r, and s but using only tuples that are matching in some way. The general join operator has the form:

$$r \bowtie_{join\_cond} s$$

where $join\_cond$ is an expression of the form

$$r.A_1\ op_1\ s.B_1\ AND\ r.A_2\ op_2\ s.B_2 \ldots\ AND\ r.A_n\ op_n\ s.B_n$$

op_i's are either $=, \neq, >, \leq, <, \geq$, $r.A_i$s' are attributes of r, and $s.B_j$'s are attributes of s. (a conjunction of comparisons between attributes of r and s)

Example 3 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

and s be the following relation instance:

B	*D*
2	*3*
3	*1*

Then: $r \rhd\lhd_{r.B=s.B} s$ is

A	*B*	*C*	*s.B*	*D*
1	*2*	*3*	*2*	*3*
3	*2*	*2*	*2*	*3*

The first tuple of $r \rhd\lhd_{r.B=s.B} s$ is combined from the 1^{st} tuple of r and the 1^{st} tuple of s. The second tuple of $r \rhd\lhd_{r.B=s.B} s$ is combined from the 3^{st} tuple of r and the 1^{st} tuple of s. For the second tuple of s, the value of B is 3 and none of the tuples in r has $B = 3$. So, no new tuple for $r \rhd\lhd_{r.B=s.B} s$ can be formed using the second tuple of s.

Special cases: *Equi-join* is a join between r and s whose join condition is a conjunction of equality comparisons; *natural join* is an equi-join whose join condition is the conjunction of all equality comparisons between shared attributes of r and s. Often, a join without join-condition is understood as a natural join.

Join is a derived operator : $r \rhd\lhd_C s$ is given by

$$r \rhd\lhd_C s = \sigma_C(r \times s).$$

Renaming (ρ): $\rho_{s(A_1,\ldots,A_k)}(r)$ creates a new relation, named s, and attributes $\{A_1, \ldots, A_k\}$ with all the tuples of r.

Example 4 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

Then, $\rho_{T(M,N,P)}(r)$ is a new relation T with the following instance

M	*N*	*P*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

NOTE: $\rho_s(r)$ only changes the name of the relation.

Division operator For r with the set of attributes $\{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ and s with the set of attributes $\{B_1, \ldots, B_m\}$ (or $r(R)$ where $R = (A_1, \ldots, A_n, B_1, \ldots, B_m)$ and $s(S)$ where $S = (B_1, \ldots, B_m)$) $r \div s$ is a relation over the set of attributes $\{A_1, \ldots, A_n\}$ (or $R - S$) with the set of tuples $\langle t \rangle$ where $\{\langle t \rangle\} \times s \subseteq r$. This is equivalent to say that t belongs to $r \div s$

1. t is in Π_{r-s}
2. for each t_s in s there exists a tuple t_r in r such that
   (a) $t_r[S] = t_s[S]$ and
   (b) $t_r[R - S] = t$

Example 5 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
1	*3*	*1*

and s be the following relation instance:

B	*C*
2	*3*
3	*1*

Then: $r \div s$ is

A
1

Combining Operations to Form Queries. The operators of relational algebra allows us to form expressions of arbitrary complexity by applying operators to either a given relation or to the relations that are the result of applying operators to relations.

Example like the theta-join operator, or something like $\pi_{A,B}(\sigma_{C>D}(r \times s))$ etc.

Assignment. Uses for storing temporary results. $r \leftarrow exp$ where exp is a relational expression indicates that r is the relation resulting from exp. For example, we write

$$r' \leftarrow \Pi_{A,B}(r)$$
$$s' \leftarrow \Pi_{A,B}(s)$$
$$result = r' \times s'$$

to represent the complex expression

$$\Pi_{A,B}(r) \times \Pi_{A,B}(s)$$

2.3 Extended Relational Operations

This includes the generalized projection, aggregate functions, and outer joints.

Outer-, Left Outer-, and Right Outer-join The join operator might remove some tuples of the relations involved in the join. To retain the 'dangling' tuples (for different purposes), outer-, left outer-, or right outer-join can be used. They are defined as follows:

Outer join $r \rhd\lhd_C^{outer} s$ consists of (i) (the tuples in) $r \rhd\lhd_C s$, (ii) the tuples in r that do not join with any tuple in s (NULL valued padded to the part of s), and (iii) the tuples in s that do not join with any tuple in r (NULL valued padded to the part of r).

Left outer join $r \rhd\lhd_C^{left} s$ consists of tuples in (i) and (iii) (of outer join)

Right outer join $r \rhd\lhd_C^{right} s$ consists of tuples in (i) and (ii) (of outer join)

Example 6 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

and s be the following relation instance:

B	*D*
2	*3*
3	*1*

Then: $r \rhd\lhd_{r.B=s.B} s$ *is*

A	*B*	*C*	*s.B*	*D*
1	*2*	*3*	*2*	*3*
3	*2*	*2*	*2*	*3*

$r \rhd\lhd_{r.B=s.B}^{outer} s$ *is*

A	*B*	*C*	*s.B*	*D*
1	*2*	*3*	*2*	*3*
3	*2*	*2*	*2*	*3*
2	*1*	*1*	*NULL*	*NULL*
4	*1*	*3*	*NULL*	*NULL*
NULL	*NULL*	*NULL*	*3*	*1*

$r \rhd\lhd_{r.B=s.B}^{right} s$ *is*

A	*B*	*C*	*s.B*	*D*
1	*2*	*3*	*2*	*3*
3	*2*	*2*	*2*	*3*
2	*1*	*1*	*NULL*	*NULL*
4	*1*	*3*	*NULL*	*NULL*

$r \rhd\lhd_{r.B=s.B}^{left} s$ *is*

A	*B*	*C*	*s.B*	*D*
1	*2*	*3*	*2*	*3*
3	*2*	*2*	*2*	*3*
NULL	*NULL*	*NULL*	*3*	*1*

Generalized Projection. $\Pi_{F_1,\ldots,F_n}(E)$ where E is a relational algebra expression and F_i is either an arithmetic expression involving the attributes and constants in the schema of E. The result is a relation with the schema of n attributes. $t = (v_1, \ldots, v_n)$ is a tuple belonging to the result if there exists a tuple e in E such that v_i is the value of F_i with respect to e.

Example 7 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*

and $\Pi_{A+B,B-C}(r)$ is the following relation instance:

A + B	*B - C*
3	*-1*
3	*0*
5	*0*
5	*3*

We can also use $\Pi_{F_1 \textbf{ as } A_1,\ldots,F_n \textbf{ as } A_n}(E)$ to specify the generalized projection. The attributes of the result relation will be $A_1, \ldots, A_n$ instead of $F_1, \ldots, F_n$.

Aggregates. For a sequence of numbers S (also a multiset of numbers, a set in which elements can be repeated), $\mathbf{sum}$, $\mathbf{avg}$, $\mathbf{min}$, and $\mathbf{max}$ returns the sum, average, min, or max. An aggregate operation $\mathcal{G}$ has the following form

$$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_m(A_m)}(E)$$

where

- E is a relational algebra expression
- $G_1, \ldots, G_n$ is a list of attributes
- F_i is an aggregate function
- A_i is attribute name

The relation resulting from the above aggregate operation is computed in the following steps

- $r \leftarrow E$
- Divide r into different relations $r_1, \ldots, r_k$ where
  1. for each pair of tuples t and t' in r_i $t_{r_i}[G_1, G_2, \ldots, G_n] = t'_{r_i}[G_1, G_2, \ldots, G_n]$
  2. for each pair of tuples t and t' where t is in r_i and t' is in r_j $t_{r_i}[G_1, G_2, \ldots, G_n] \neq t'_{r_j}[G_1, G_2, \ldots, G_n]$
- For each relation r_i, compute the value $F_j(A_j)$ and create a tuple $(g_1, \ldots, g_n, v_{F_1(A_1)}(r_i), \ldots, v_{F_m(A_m)}(r_i))$ for the resulting relation.

Example 8 *Let r be the following relation instance:*

A	*B*	*C*
1	*2*	*3*
2	*1*	*1*
3	*2*	*2*
4	*1*	*3*
1	*1*	*3*

and $\mathcal{G}_{\mathbf{sum}(A)}(r)$ *is the following relation instance:*

sum(*A*)
11

${}_{B}\mathcal{G}_{\mathbf{sum}(A)}(r)$ *is the following relation instance:*

B	**sum**(*A*)
2	*5*
1	*6*

We can also use "**as** ..." to specify the new attribute name in aggregate operations. For example, ${}_{B}\mathcal{G}_{\mathbf{sum}(A)\ \mathbf{as}\ sA}(r)$ is the following relation instance (r refers to the relation in the previous example):

B	sA
2	5
1	6