Relational algebra and relational calculus

•Relational algebra and relational calculus are formal languages associated with the relational model.
•Informally, relational algebra is a (high-level) procedural language and relational calculus a non-procedural language.
•However, formally both are equivalent to one another.
•A language that produces a relation that can be derived using relational calculus is <u>relationally complete</u>.

Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:
<u>Relational Algebra</u>: More operational, very useful for representing execution plans.
•Relational algebra operations work on one or more relations to define another relation without changing the original relations.
•Both operands and results are relations, so output from one operation can become input to another operation.
•Allows expressions to be nested, just as in arithmetic. This property is called <u>closure</u>.
•Five basic operations in relational algebra: Selection, Projection, Cartesian product, Union, and Set Difference.
•Basic operations:
•<u>*Selection*</u> ( ) Selects a subset of rows from relation. Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (*predicate*). *p* is called the selection predicate , r can be the name of a table, or another query

•<u>*Projection*</u> ( ) Deletes unwanted columns from relation. Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.

•<u>*Cross-product*</u> ( ) Allows us to combine two relations. •Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.
•For sets A, B, the Cartesian product, or cross product, of A and B is denoted by A × B and equals {(a, b) | a Œ A, b Œ B}
•Elements of A × B are ordered pairs. For (a, b), (c, d) Œ A × B , (a, b) = (c, d) if and only if a = c and b = d
Properties:
1.If A, B are finite, it follows from the rule of product that |A × B| = |A||B|
2.Although we generally will not have A × B = B × A, we will have |A×B|=|B×A|

•<u>*Set-difference*</u> ( ) Tuples in reln. 1, but not in reln. 2. •Defines a relation consisting of the tuples that are in relation R, but not in S.
•R and S must be union-compatible.

•<u>*Union*</u> (») Tuples in reln. 1 and in reln. 2. •R » S
•Union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated.
•R and S must be union-compatible.

•If R and S have *I* and *J* tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (*I* + *J*) tuples.

•Additional operations:
•Intersection, *join*, division, renaming: Not essential, but (very!) useful.
•Since each operation returns a relation, operations can be *composed*! (Algebra is “closed”.)
•These perform most of the data retrieval operations needed.
•Also have Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations

Join Operations

•Join is a derivative of Cartesian product.
•Equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations.
•One of the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMSs have intrinsic performance problems.

Theta join (q-join)

•Defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.
•The predicate F is of the form $R.a_i$ q $S.b_i$ where q may be one of the comparison operators (<, £, >, ≥, =, π).

Natural join

•An Equijoin of the two relations R and S over all common attributes *x*. One occurrence of each common attribute is eliminated from the result.

Outer join

•To display rows in the result that do not have matching values in the join column, use Outer join.
•(Left) outer join is join in which tuples from R that do not have matching values in common columns of S are also included in result relation.

Semijoin

•Defines a relation that contains the tuples of R that participate in the join of R with S.

Division

•R ∏ S
•Defines a relation over the attributes C that consists of set of tuples from R that match combination of *every* tuple in S.
•Expressed using basic operations:

T_1 ¨ $P_C(R)$
T_2 ¨ $P_C((S \times T_1) - R)$
T ¨ $T_1 - T_2$

•*Idea*: For *A*/*B*, compute all *x* values that are not `disqualified’ by some *y* value in *B*.
•*x* value is *disqualified* if by attaching *y* value from *B*, we obtain an *xy* tuple that is not in *A*.

Relational Calculus: Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

Relations

Let A = {0,1,2}, B = {1,2,3}. A x B = {(0,1), (0,2), (0,3), (1,1), (1,2), (1,3), (2,1), (2,2), (2,3)}
Let say an element x in A is related to an element y in B iff x is less than y. x R y: x is related to y
\0 R 1, 0 R 2, 0 R 3, 1 R 2, 1 R 3, 2 R 3
\The set of all ordered pair in A x B where elements are related {(0,1), (0,2), (0,3), (1,2), (1,3), (2,3)}
•For sets A, B, a (binary) relation R from A to B is a subset of A × B. Any subset of A × A is called a (binary) relation on A
•Given an ordered pair (a, b) in A x B, x is related to y by R   (x R y) iff (x, y) is in R
•In general, for finite sets A, B with |A| = m and |B|= n, there are 2^{mn} relations from A to B, including the empty relation as well as the relation A × B itself
Let A = {2, 3, 4}, B = {4, 5}. Then
A × B = {(2, 4), (2, 5), (3, 4), (3, 5), (4, 4), (4, 5)}.
The following are some of the relations from A to B.
i.∆
ii.{(2, 4)}
iii.{(2, 4), (2, 5)}
iv.{(2, 4), (3, 4), (4, 4)}
v.{(2, 4), (3, 4), (4, 5)}
vi.A × B
Since |A × B| = 6, there are 2^6 possible relations from A to B (for there are 2^6 possible subsets of A × B)

Aggregate Operations
•Applies aggregate function list, AL, to R to define a relation over the aggregate list.
•AL contains one or more (<aggregate_function>, <attribute>) pairs .
•Main aggregate functions are: COUNT, SUM, AVG, MIN, and MAX.

Grouping Operation
•Groups tuples of R by grouping attributes, GA, and then applies aggregate function list, AL, to define a new relation.
•AL contains one or more (<aggregate_function>, <attribute>) pairs.
•Resulting relation contains the grouping attributes, GA, along with results of each of the aggregate functions.