Project(R,A)

Create a new relation that retains only the attributes A taken from R.

Select(R,C)

Create a new relation including only tuples from R that satisfy C

DupElim(R)

Create a new relation from R by including each unique tuple exactly once

GroupMin(R,A,B)

Create a new relation consisting of unique tuples of the attributes A and the minima of the attributes B over the corresponding grouped sets of tuples.

GroupMax(R,A,B)

Create a new relation consisting of unique tuples of the attributes A and the maxima of the attributes B over the corresponding grouped sets of tuples.

GroupCount(R,A)

Create a new relation consisting of unique tuples of the attributes A and counts of the sizes of corresponding grouped sets of tuples.

GroupSum(R,A,B)

Create a new relation consisting of unique tuples of the attributes A and the sums of the attributes B over the corresponding grouped sets of tuples.

GroupAvg(R,A,B)

Create a new relation consisting of unique tuples of the attributes A and the averages of the attributes B over the corresponding grouped sets of tuples.

SetUnion(R,S)

Create a new relation containing each of the unique tuples found in either R or S.

BagUnion(R,S)

Create a new relation containing each of the tuples found in either R or S (including duplicates).

SetIntersection(R,S)

Create a new relation containing each of the unique tuples found in both R and S.

BagIntersection(R,S)

Create a new relation containing each tuple found in both R and S, repeated the lesser of their number of occurrences in each.

SetDifference(R,S)

Create a new relation containing each unique tuple found in R but not in S

BagDifference(R,S)

Create a new relation containing each unique tuple found in R more often than S, as many times as there are excess appearances in R

Product(R,S)

Create a new relation containing every possible concatenation of a tuple from R with a tuple from S.

NaturalJoin(R,S)

Create a new relation containing concatenations of a tuple from R with a tuple from S, where the tuples match on shared attributes.

SortUnion(R,S)

Uses merge sort to take the union of large relations R and S

HashIntersection(R,S)

Uses hashing to take the intersection of large relations R and S

SortIntersection(R,S)

Uses merge sort to take the intersection of large relations R and S

HashDifference(R,S)

Uses hashing to take the set difference of large relations R and S

SortDifference(R,S)

Uses merge sort to take the set difference of large relations R and S

HashDupElim(R)

Uses hashing to eliminate duplicates in large relation R

SortDupElim(R)

Uses merge sort to eliminate duplicates in large relation R

HashGroupAgg(R,A,G)

Uses hashing to compute some aggregated property G of tuples from large relation R, as grouped by attributes A

SortGroupAgg(R,A,G)

Uses merge sort to compute some aggregated property G of tuples from large relation R, as grouped by attributes A

HashJoin(R,S)

Uses hashing to produce a join of large relations R and S

SortJoin(R,S)

Uses merge sort to produce a join of large relations R and S

SortedIndexJoin

Uses a sorted index to produce a join of large relations R and S

HashUnion(R,S)

Uses hashing to take the union of large relations R and S

Sort(R)

Uses multiway merge sort on large relation R