Review of Notation

Recall from Section 15.1.3 the following size

query plan. Even an inaccurate size-estimation method will serve that purpose well if it errs consistently, that is, if the size estimator assigns the least cost to the best physical query plan, even if the actual cost of that plan turns out to be different from what was predicted.

16.4.2 Estimating the Size of a Projection

The extended projection of Section 5.2.5 is a bag projection and does not eliminate duplicates. We shall treat a clasical, duplicate-eliminating projection as a bag-projection followed by a δ. The extended projection of bags is different from the other operators, in that the size of the result is computable exactly. Normally, tuples shrink during a projection, as some components are eliminated. However, the extended projection allows the creation of new components that are combinations of attributes, and so there are situations where a π operator actually increases the size of the relation.

Example 16.20: Suppose $R(a, b, c)$ is a relation, where a and b are integers of four bytes each, and c is a string of 100 bytes. Let tuple headers require 12 bytes. Then each tuple of R requires 120 bytes. Let blocks be 1024 bytes long, with block headers of 24 bytes. We can thus fit 8 tuples in one block. Suppose $T(R) = 10{,}000$; i.e., there are 10,000 tuples in R. Then $B(R) = 1250$.

Consider $S = \pi_{a+b\to x,c}(R)$; that is, we replace a and b by their sum. Tuples of S require 116 bytes: 12 for header, 4 for the sum, and 100 for the string. Although tuples of S are slightly smaller than tuples of R, we can still fit only 8 tuples in a block. Thus, $T(S) = 10{,}000$ and $B(S) = 1250$.

Now consider $U = \pi_{a,b}(R)$, where we eliminate the string component. Tuples of U are only 20 bytes long. $T(U)$ is still 10,000. However, we can now pack 50 tuples of U into one block, so $B(U) = 200$. This projection thus shrinks the relation by a factor slightly more than 6. □

16.4.3 Estimating the Size of a Selection

When we perform a selection, we generally reduce the number of tuples, although the sizes of tuples remain the same. In the simplest kind of selection, where an attribute is equated to a constant, there is an easy way to estimate the size of the result, provided we know, or can estimate, the number of different values the attribute has. Let $S = \sigma_{A=c}(R)$, where A is an attribute of R and c is a constant. Then we recommend as an estimate:

- $T(S) = T(R)/V(R, A)$

This rule surely holds if the value of A is chosen randomly from among all the possible values.

The size estimate is more problematic when the selection involves an inequality comparison, for instance, $S = \sigma_{a<10}(R)$. One might think that on the average, half the tuples would satisfy the comparison and half not, so $T(R)/2$

The Zipfian Distribution

In estimating the size of a selection $\sigma_{A=c}$ it is not necessary to assume that values of A appear equally often. In fact, many attributes have values whose occurrences follow a *Zipfian distribution*, where the frequencies of the ith most common values are in proportion to $1/\sqrt{i}$. For example, if the most common value appears 1000 times, then the second most common value would be expected to appear about $1000/\sqrt{2}$ times, or 707 times, and the third most common value would appear about $1000/\sqrt{3}$ times, or 577 times. Originally postulated as a way to describe the relative frequencies of words in English sentences, this distribution has been found to appear in many sorts of data. For example, in the US, state populations follow an approximate Zipfian distribution. The three most populous states, California, Texas, and New York, have populations in ratio approximately 1:0.62:0.56, compared with the Zipfian ideal of 1:0.71:0.58. Thus, if `state` were an attribute of a relation describing US people, say a list of magazine subscribers, we would expect the values of `state` to distribute in the Zipfian, rather than uniform manner.

As long as the constant in the selection condition is chosen randomly, it doesn't matter whether the values of the attribute involved have a uniform, Zipfian, or other distribution; the *average* size of the matching set will still be $T(R)/V(R,a)$. However, if the constants are also chosen with a Zipfian distribution, then we would expect the average size of the selected set to be somewhat larger than $T(R)/V(R,a)$.

would estimate the size of S. However, there is an intuition that queries involving an inequality tend to retrieve a small fraction of the possible tuples.[3] Thus, we propose a rule that acknowledges this tendency, and assumes the typical inequality will return about one third of the tuples, rather than half the tuples. If $S = \sigma_{a<c}(R)$, then our estimate for $T(S)$ is:

- $T(S) = T(R)/3$

The case of a "not equals" comparison is rare. However, should we encounter a selection like $S = \sigma_{a\neq 10}(R)$, we recommend assuming that essentially all tuples will satisfy the condition. That is, take $T(S) = T(R)$ as an estimate. Alternatively, we may use $T(S) = T(R)\big(V(R,a)-1\big)/V(R,a)$, which is slightly less, as an estimate, acknowledging that about fraction $1/V(R,a)$ tuples of R will fail to meet the condition because their a-value *does* equal the constant.

When the selection condition C is the `AND` of several equalities and inequalities, we can treat the selection $\sigma_C(R)$ as a cascade of simple selections, each of

[3]For instance, if you had data about faculty salaries, would you be more likely to query for those faculty who made *less* than \$200,000 or *more* than \$200,000?

which checks for one of the conditions. Note that the order in which we place these selections doesn't matter. The effect will be that the size estimate for the result is the size of the original relation multiplied by the *selectivity* factor for each condition. That factor is 1/3 for any inequality, 1 for $\neq$, and $1/V(R,A)$ for any attribute A that is compared to a constant in the condition C.

Example 16.21: Let $R(a,b,c)$ be a relation, and $S = \sigma_{a=10 \text{ AND } b<20}(R)$. Also, let $T(R) = 10{,}000$, and $V(R,a) = 50$. Then our best estimate of $T(S)$ is $T(R)/(50 \times 3)$, or 67. That is, 1/50th of the tuples of R will survive the $a = 10$ filter, and 1/3 of those will survive the $b < 20$ filter.

An interesting special case where our analysis breaks down is when the condition is contradictory. For instance, consider $S = \sigma_{a=10 \text{ AND } a>20}(R)$. According to our rule, $T(S) = T(R)/3V(R,a)$, or 67 tuples. However, it should be clear that no tuple can have both $a = 10$ and $a > 20$, so the correct answer is $T(S) = 0$. When rewriting the logical query plan, the query optimizer can look for instances of many special-case rules. In the above instance, the optimizer can apply a rule that finds the selection condition logically equivalent to FALSE and replaces the expression for S by the empty set. □

When a selection involves an OR of conditions, say $S = \sigma_{C_1 \text{ OR } C_2}(R)$, then we have less certainty about the size of the result. One simple assumption is that no tuple will satisfy both conditions, so the size of the result is the sum of the number of tuples that satisfy each. That measure is generally an overestimate, and in fact can sometimes lead us to the absurd conclusion that there are more tuples in S than in the original relation R.

A less simple, but possibly more accurate estimate of the size of

$$S = \sigma_{C_1 \text{ OR } C_2}(R)$$

is to assume that C_1 and C_2 are independent. Then, if R has n tuples, m_1 of which satisfy C_1 and m_2 of which satisfy C_2, we would estimate the number of tuples in S as $n\big(1-(1-m_1/n)(1-m_2/n)\big)$. In explanation, $1-m_1/n$ is the fraction of tuples that do not satisfy C_1, and $1-m_2/n$ is the fraction that do not satisfy C_2. The product of these numbers is the fraction of R's tuples that are *not* in S, and 1 minus this product is the fraction that are in S.

Example 16.22: Suppose $R(a,b)$ has $T(R) = 10{,}000$ tuples, and

$$S = \sigma_{a=10 \text{ OR } b<20}(R)$$

Let $V(R,a) = 50$. Then the number of tuples that satisfy $a = 10$ we estimate at 200, i.e., $T(R)/V(R,a)$. The number of tuples that satisfy $b < 20$ we estimate at $T(R)/3$, or 3333.

The simplest estimate for the size of S is the sum of these numbers, or 3533. The more complex estimate based on independence of the conditions $a = 10$ and $b < 20$ gives $10000\big(1-(1-200/10000)(1-3333/10000)\big)$, or 3466. In this case, there is little difference between the two estimates, and it is very unlikely

that choosing one over the other would change our estimate of the best physical query plan. □

The final operator that could appear in a selection condition is NOT. The estimated number of tuples of R that satisfy condition NOT C is $T(R)$ minus the estimated number that satisfy C.

16.4.4 Estimating the Size of a Join

We shall consider here only the natural join. Other joins can be handled according to the following outline:

1. The number of tuples in the result of an equijoin can be computed exactly as for a natural join, after accounting for the change in variable names. Example 16.24 will illustrate this point.
2. Other theta-joins can be estimated as if they were a selection following a product. Note that the number of tuples in a product is the product of the number of tuples in the relations involved.

We shall begin our study with the assumption that the natural join of two relations involves only the equality of two attributes. That is, we study the join $R(X,Y) \bowtie S(Y,Z)$, but initially we assume that Y is a single attribute although X and Z can represent any set of attributes.

The problem is that we don't know how the Y-values in R and S relate. For instance:

1. The two relations could have disjoint sets of Y-values, in which case the join is empty and $T(R \bowtie S) = 0$.
2. Y might be the key of S and the corresponding foreign key of R, so each tuple of R joins with exactly one tuple of S, and $T(R \bowtie S) = T(R)$.
3. Almost all the tuples of R and S could have the same Y-value, in which case $T(R \bowtie S)$ is about $T(R)T(S)$.

To focus on the most common situations, we shall make two simplifying assumptions:

- *Containment of Value Sets.* If Y is an attribute appearing in several relations, then each relation chooses its values from the front of a fixed list of values $y_1, y_2, y_3, \ldots$ and has all the values in that prefix. As a consequence, if R and S are two relations with an attribute Y, and $V(R,Y) \leq V(S,Y)$, then every Y-value of R will be a Y-value of S.
- *Preservation of Value Sets.* If we join a relation R with another relation, then an attribute A that is not a join attribute (i.e., not present in both relations) does not lose values from its set of possible values. More precisely, if A is an attribute of R but not of S, then $V(R \bowtie S, A) = V(R, A)$.

Assumption (1), containment of value sets, clearly might be violated, but it *is* satisfied when Y is a key in S and the corresponding foreign key in R. It also is approximately true in many other cases, since we would intuitively expect that if S has many Y-values, then a given Y-value that appears in R has a good chance of appearing in S.

Assumption (2), preservation of value sets, also might be violated, but it is true when the join attribute(s) of $R \bowtie S$ are a key for S and the corresponding foreign key of R. In fact, (2) can only be violated when there are "dangling tuples" in R, that is, tuples of R that join with no tuple of S; and even if there *are* dangling tuples in R, the assumption might still hold.

Under these assumptions, we can estimate the size of $R(X,Y) \bowtie S(Y,Z)$ as follows. Suppose r is a tuple in R, and S is a tuple in S. What is the probability that r and s agree on attribute Y? Suppose that $V(R,Y) \geq V(S,Y)$. Then the Y-value of s is surely one of the Y values that appear in R, by the containment-of-value-sets assumption. Hence, the chance that r has the same Y-value as s is $1/V(R,Y)$. Similarly, if $V(R,Y) < V(S,Y)$, then the value of Y in r will appear in S, and the probability is $1/V(S,Y)$ that r and s will share the same Y-value. In general, we see that the probability of agreement on the Y value is $1/\max\big(V(R,Y),V(S,Y)\big)$. Thus:

- $T(R \bowtie S) = T(R)T(S)/\max\big(V(R,Y),V(S,Y)\big)$

That is, the estimated number of tuples in $T(R \bowtie S)$ is the number of pairs of tuples — one from R and one from S, times the probability that such a pair shares a common Y value.

Example 16.23: Let us consider the following three relations and their important statistics:

$R(a,b)$	$S(b,c)$	$U(c,d)$
$T(R) = 1000$	$T(S) = 2000$	$T(U) = 5000$
$V(R,b) = 20$	$V(S,b) = 50$	
	$V(S,c) = 100$	$V(U,c) = 500$

Suppose we want to compute the natural join $R \bowtie S \bowtie U$. One way is to group R and S first, as $(R \bowtie S) \bowtie U$. Our estimate for $T(R \bowtie S)$ is $T(R)T(S)/\max\big(V(R,b),V(S,b)\big)$, which is $1000 \times 2000/50$, or 40,000.

We then need to join $R \bowtie S$ with U. Our estimate for the size of the result is $T(R \bowtie S)T(U)/\max\big(V(R \bowtie S,c),V(U,c)\big)$. By our assumption that value sets are preserved, $V(R \bowtie S,c)$ is the same as $V(S,c)$, or 100; that is no values of attribute c disappeared when we performed the join. In that case, we get as our estimate for the number of tuples in $R \bowtie S \bowtie U$ the value $40{,}000 \times 5000/\max(100,500)$, or 400,000.

We could also start by joining S and U. If we do, then we get the estimate $T(S \bowtie U) = T(S)T(U)/\max\big(V(S,c),V(U,c)\big) = 2000 \times 5000/500 = 20{,}000$.

By our assumption that value sets are preserved, $V(S \bowtie U, b) = V(S, b) = 50$, so the estimated size of the result is

$$T(R)T(S \bowtie U)/\max\big(V(R,b), V(S \bowtie U, b)\big)$$

which is $1000 \times 20{,}000/50$, or 400,000. □

16.4.5 Natural Joins With Multiple Join Attributes

When the set of attributes Y in the join $R(X,Y) \bowtie S(Y,Z)$ consists of more than one attribute, the same argument as we used for a single attribute Y applies to each attribute in Y. That is:

- The estimate of the size of $R \bowtie S$ is computed by multiplying $T(R)$ by $T(S)$ and dividing by the larger of $V(R,y)$ and $V(S,y)$ for each attribute y that is common to R and S.

Example 16.24: The following example uses the rule above. It also illustrates that the analysis we have been doing for natural joins applies to any equijoin. Consider the join

$$R(a,b,c) \bowtie_{R.b=S.d \text{ AND } R.c=S.e} S(d,e,f)$$

Suppose we have the following size parameters:

$R(a,b,c)$	$S(d,e,f)$
$T(R) = 1000$	$T(S) = 2000$
$V(R,b) = 20$	$V(S,d) = 50$
$V(R,c) = 100$	$V(S,e) = 50$

We can think of this join as a natural join if we regard $R.b$ and $S.d$ as the same attribute and also regard $R.c$ and $S.e$ as the same attribute. Then the rule given above tells us the estimate for the size of $R \bowtie S$ is the product 1000×2000 divided by the larger of 20 and 50 and also divided by the larger of 100 and 50. Thus, the size estimate for the join is $1000 \times 2000/(50 \times 100) = 400$ tuples. □

Example 16.25: Let us reconsider Example 16.23, but consider the third possible order for the joins, where we first take $R(a,b) \bowtie U(c,d)$. This join is actually a product, and the number of tuples in the result is $T(R)T(U) = 1000 \times 5000 = 5{,}000{,}000$. Note that the number of different b's in the product is $V(R,b) = 20$, and the number of different c's is $V(U,c) = 500$.

When we join this product with $S(b,c)$, we multiply the numbers of tuples and divide by both $\max\big(V(R,b), V(S,b)\big)$ and $\max\big(V(U,c), V(S,c)\big)$. This quantity is $2000 \times 5{,}000{,}000/(50 \times 500) = 400{,}000$. Note that this third way of joining gives the same estimate for the size of the result that we found in Example 16.23. □

16.4.6 Joins of Many Relations

Finally, let us consider the general case of a natural join:

$$S = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$$

Suppose that attribute A appears in k of the R_i's, and the numbers of its sets of values in these k relations — that is, the various values of $V(R_i, A)$ for $i = 1, 2, \ldots, k$ — are $v_1 \leq v_2 \leq \cdots \leq v_k$, in order from smallest to largest. Suppose we pick a tuple from each relation. What is the probability that all tuples selected agree on attribute A?

In answer, consider the tuple t_1 chosen from the relation that has the smallest number of A-values, v_1. By the containment-of-value-sets assumption, each of these v_1 values is among the A-values found in the other relations that have attribute A. Consider the relation that has v_i values in attribute A. Its selected tuple t_i has probability $1/v_i$ of agreeing with t_1 on A. Since this claim is true for all $i = 2, 3, \ldots, k$, the probability that all k tuples agree on A is the product $1/v_2v_3 \cdots v_k$. This analysis gives us the rule for estimating the size of any join.

- Start with the product of the number of tuples in each relation. Then, for each attribute A appearing at least twice, divide by all but the least of the $V(R, A)$'s.

Likewise, we can estimate the number of values that will remain for attribute A after the join. By the preservation-of-value-sets assumption, it is the least of these $V(R, A)$'s.

Example 16.26: Consider the join $R(a, b, c) \bowtie S(b, c, d) \bowtie U(b, e)$, and suppose the important statistics are as given in Fig. 16.26. To estimate the size of this join, we begin by multiplying the relation sizes; $1000 \times 2000 \times 5000$. Next, we look at the attributes that appear more than once; these are b, which appears three times, and c, which appears twice. We divide by the two largest of $V(R, b)$, $V(S, b)$, and $V(U, b)$; these are 50 and 200. Finally, we divide by the larger of $V(R, c)$ and $V(S, c)$, which is 200. The resulting estimate is

$$1000 \times 2000 \times 5000/(50 \times 200 \times 200) = 5000$$

We can also estimate the number of values for each of the attributes in the join. Each estimate is the least value count for the attribute among all the relations in which it appears. These numbers are, for a, b, c, d, e respectively: 100, 20, 100, 400, and 500. □

Based on the two assumptions we have made — containment and preservation of value sets — we have a surprising and convenient property of the estimating rule given above.

- No matter how we group and order the terms in a natural join of n relations, the estimation rules, applied to each join individually, yield the

$R(a,b,c)$	$S(b,c,d)$	$U(b,e)$
$T(R) = 1000$	$T(S) = 2000$	$T(U) = 5000$
$V(R,a) = 100$		
$V(R,b) = 20$	$V(S,b) = 50$	$V(U,b) = 200$
$V(R,c) = 200$	$V(S,c) = 100$	
	$V(S,d) = 400$	
		$V(U,e) = 500$

Figure 16.26: Parameters for Example 16.26

same estimate for the size of the result. Moreover, this estimate is the same that we get if we apply the rule for the join of all n relations as a whole.

Examples 16.23 and 16.25 form an illustration of this rule for the three groupings of a three-relation join, including the grouping where one of the "joins" is actually a product.

16.4.7 Estimating Sizes for Other Operations

We have seen two operations — selection and join — with reasonable estimating techniques. In addition, projections do not change the number of tuples in a relation, and products multiply the numbers of tuples in the argument relations. However, for the remaining operations, the size of the result is not easy to determine. We shall review the other relational-algebra operators and give some suggestions as to how this estimation could be done.

Union

If the bag union is taken, then the size is exactly the sum of the sizes of the arguments. A set union can be as large as the sum of the sizes or as small as the larger of the two arguments. We suggest that something in the middle be chosen, e.g., the larger plus half the smaller.

Intersection

The result can have as few as 0 tuples or as many as the smaller of the two arguments, regardless of whether set- or bag-intersection is taken. One approach is to take the average of the extremes, which is half the smaller.

Difference

When we compute $R - S$, the result can have between $T(R)$ and $T(R) - T(S)$ tuples. We suggest the average as an estimate: $T(R) - T(S)/2$.

Duplicate Elimination

If $R(a_1, a_2, \ldots, a_n)$ is a relation, then $V(R, [a_1, a_2, \ldots, a_n])$ is the size of $\delta(R)$. However, often we shall not have this statistic available, so it must be approximated. In the extremes, the size of $\delta(R)$ could be the same as the size of R (no duplicates) or as small as 1 (all tuples in R are the same).[4] Another upper limit on the number of tuples in $\delta(R)$ is the maximum number of distinct tuples that could exist: the product of $V(R, a_i)$ for $i = 1, 2, \ldots, n$. That number could be smaller than other estimates of $T\big(\delta(R)\big)$. There are several rules that could be used to estimate $T\big(\delta(R)\big)$. One reasonable one is to take the smaller of $T(R)/2$ and the product of all the $V(R, a_i)$'s.

Grouping and Aggregation

Suppose we have an expression $\gamma_L(R)$, the size of whose result we need to estimate. If the statistic $V(R, [g_1, g_2, \ldots, g_k])$, where the g_i's are the grouping attributes in L, is available, then that is our answer. However, that statistic may well not be obtainable, so we need another way to estimate the size of $\gamma_L(R)$. The number of tuples in $\gamma_L(R)$ is the same as the number of groups. There could be as few as one group in the result or as many groups as there are tuples in R. As with δ, we can also upper-bound the number of groups by a product of $V(R, A)$'s, but here attribute A ranges over only the grouping attributes of L. We again suggest an estimate that is the smaller of $T(R)/2$ and this product.

16.4.8 Exercises for Section 16.4

Exercise 16.4.1: Below are the vital statistics for four relations, W, X, Y, and Z:

$W(a,b)$	$X(b,c)$	$Y(c,d)$	$Z(d,e)$
$T(W) = 100$	$T(X) = 200$	$T(Y) = 300$	$T(Z) = 400$
$V(W,a) = 20$	$V(X,b) = 50$	$V(Y,c) = 50$	$V(Z,d) = 40$
$V(W,b) = 60$	$V(X,c) = 100$	$V(Y,d) = 50$	$V(Z,e) = 100$

Estimate the sizes of relations that are the results of the following expressions:

(a) $W \bowtie X \bowtie Y \bowtie Z$ (b) $\sigma_{a=10}(W)$ (c) $\sigma_{c=20}(Y)$
(d) $\sigma_{c=20}(Y) \bowtie Z$ (e) $W \times Y$ (f) $\sigma_{d>10}(Z)$
(g) $\sigma_{a=1 \text{ AND } b=2}(W)$ (h) $\sigma_{a=1 \text{ AND } b>2}(W)$ (i) $X \bowtie_{X.c<Y.c} Y$

Exercise 16.4.2: Here are the statistics for four relations E, F, G, and H:

[4] Strictly speaking, if R is empty there are no tuples in either R or $\delta(R)$, so the lower bound is 0. However, we are rarely interested in this special case.