SELECTIVE PARTIAL ACCESS TO A DATABASE[+]

Richard Conway and David Strip

TR 76-269

March 1976

Department of Computer Science
Cornell University
Ithaca, New York 14853

[+]This work was supported in part by Grant GJ-41115 of the National Science Foundation.

Selective Partial Access to a Database

Richard Conway and David Strip
Cornell University

Database systems are generally intended to be used for multiple functions, and consequently by different users. The managing system should be able to restrict each user to precisely that subset of the information that is required to perform his legitimate tasks. In general, this means restricting both the subset of logical records that can be accessed, and the subset of the fields of the accessible records. Each user's authorized access to fields can be specified by an access function: $A(i,j) = 1$; if user i is allowed access to the jth field of the record; 0, otherwise. While most database systems do not yet generally offer this degree of flexibility, there are existing systems that demonstrate its feasibility. IBM's IMS, and a generalized dbms ASAP both have this type of capability (references 2,3,6). The DBTG proposal also includes this capability (ref 4).

Our present purpose is to explore the possibility of generalizing the access function to allow a third alternative -- something between absolute denial and complete access to a particular field. Roughly speaking, we will examine alternatives that would allow a user some form of partial access to certain fields -- access that would be sufficient for him to perform various statistical and summarizing tasks, without his being able to examine the exact values associated with the given fields in a particular record.

For example, consider a user who should not be allowed access to the "salary" field in a personnel database. It might still be useful to allow this user to be able to do such things as

a. compute the average salary for the employees of a given department,

b. find the maximum salary for any employee whose highest earned degree is a BS,

c. print an alphabetical list of employees whose salary exceeds $20,000.

For each of these examples, if the database system denies any access to the salary field, one of two things must happen. Either this user must be granted complete access to the salary field so he can do his assigned job, or the job must be reassigned to some other user whose job already requires him to have access to salary information.

+This work was supported in part by Grant GJ-41115 of the National Science Foundation.

Three Models for Partial Access

We will examine three different strategies by which partial access might be allowed:

1. Distortion: the actual field values are altered by some "random" quantity before being presented to the user.

2. Dissociation: actual values are presented, but only after they have been dissociated from the actual records in which they occur.

3. Value-class membership: possible values are partitioned into a set of disjoint, mutually-exhaustive classes, and the user is permitted to know only into which class the actual value falls.

In each case we will examine the potential utility of the strategy -- the type of query it would permit the user to perform; the vulnerability of the strategy to penetration by a determined and persistent user; and various alternative schemes by which the strategy might be implemented.

We will limit our consideration to retrieval tasks -- that is, tasks that only require "read access" to the database. For convenience and consistency we will imagine the tasks to be presented in an ASAP-like language. Each task will involve specification of some subset of the logical records of the database, and description of some process to be performed with the information from those selected records. The general form is

```
FOR ALL <record-name> WITH <condition>, <task>.
```

For example, the tasks listed on page 1 would be written in the following way:

```
FOR ALL EMPLOYEES WITH DEPT = 'SALES' AND LOCN = 'NYC',
    SUMSAL = SUMSAL + SALARY; NUMSAL = NUMSAL + 1;
    TO FINISH: AVGSAL = SUMSAL/NUMSAL; PRINT AVGSAL.

FOR ALL EMPLOYEES WITH HIGHDEGREE = 'BS',
    IF SALARY > MAXBSSALARY, MAXBSSALARY = SALARY;
    TO FINISH: PRINT MAXBSSALARY.

FOR ALL EMPLOYEES WITH SALARY > 20000,
    PRINT A LIST OF NAME, DEPT, JOB_TITLE, ORDERED BY
    NAME.
```

In general, we must be concerned with fields that occur in the condition by which records are selected -- which we will call "selection fields", as well as fields that occur in the task performed upon those records -- which we will call "projection fields".

We will assume that these strategies for partial access are implemented by a "secure sub-system" so that a user cannot readily circumvent the managing system to avoid the restriction upon his access. This is only a modestly unrealistic assumption. While it is true that general-purpose operating systems in general use today are dismayingly penetrable, one can achieve acceptable levels of system security by dedicating a system to a particular application. Moreover, apparently some new virtual-machine architectures provide a satisfactory approximation to a dedicated machine (ref 1). For example, an ASAP system, running in its own virtual machine under VM/370, would appear to be an adequately secure subsystem for most commercial applications.

We further assume that a user knows the nature of the restriction that has been placed upon his access, and is aware in at least a general way of the manner in which this restriction works. This is necessary for him to know how to interpret the information he receives, and in any event, security systems should not be dependent upon secrecy about the nature of their implementation.

We ignore the real possibility of collusion between users, and also the risk of masquerading, wiretapping and other more esoteric threats, since these seem to have no special relevance to our problem of partial access.

Value Distortion

The simplest and most obvious strategy, which has often been suggested, would be the modify the value of a restricted field by some random quantity before returning the value to the query. That is

$$V_p = V_f + r_d$$

where V_p is the projected value, V_f is the actual value of the field in the database record, and r_d is a random deviate with distribution d. The plus sign in the formula implies that this strategy is applicable only to fields with arithmetic values. One can imagine random distortions of character-valued fields (perhaps by random displacement in the collating sequence), but it is hard to imagine the resulting V_p being of any use whatever.

The distribution d should presumably be chosen to have an expected value of zero, so that V_p is an unbiased estimator of the true value V_f. However, beyond that, it is not always obvious what would constitute an appropriate distribution. If the population of values V_f in the database is well-behaved and symmetric, then d should probably also be symmetric, with a variance chosen to obscure but not overwhelm the V_f values. But if the population of V_f is highly skewed, which is a common occurrence, then the choice of d is much more difficult. A variance chosen so as not to overwhelm the large number of relatively small V_f will be quite ineffective in disguising the rare large values of V_f. If d is not fixed for a particular field over the entire database, then there is further difficulty -- but this is easier to understand in the terms presented in the next section, so we postpone our discussion until then.

The entire discipline of statistical inference -- estimation, confidence intervals, ranking, tests of hypotheses, non-parametric statistics, regression, correlation -- is available to determine how to use and interpret the values V_p. Models such as the one given above are what one normally assumes in statistical problems, where the random component is imposed by nature. It is the usual task of the statistician to draw inferences about the population of V_f values, from a sample of observations of the V_p. The difference is that in this case the random component is artificially introduced to make a process that is actually deterministic appear to be stochastic. This deliberate introduction of what the statistician works to overcome seems like mathematical masochism, but it accomplishes the required objective -- individual values are obscured, while any aggregative task involving a reasonable number of records is essentially unimpeded.

It is simplest to understand this strategy when the protected field is a projection field, but obviously a field must also be protected when it is used in record selection, since selection is, in effect, just an indirect form of projection. For example, without protection of selection fields, a sequence of queries such as the following could come arbitrarily close to determining the salary value V_f of an individual employee record:

```
FOR ALL EMPLOYEES WITH SALARY >= 20000 AND SALARY < 20500,
    PRINT A LIST OF NAME, SOC_SEC_NBR, JOB_TITLE.
```

The user should understand that <u>every</u> use of a protected field will be distorted by the application of r_d, so the query above would yield a subset of records whose membership is altered with respect to that of the true selection set.

It is an important question as to whether or not the values protected in this manner should be <u>repeatable</u>. That is, if a record is selected on a second query, and if there has been no intervening update of the field, should the value V_p be identically the same as on the initial query? That is equivalent to asking whether the random deviate will be repeated, or whether a new value is used each time the particular record is selected. It is probably easier to implement a system in which values are not repeated, but to do so makes the system vulnerable to assault by simply repeating a query. For example, repetition of the query:

```
FOR ALL EMPLOYEES WITH LASTNAME = 'DAVIES',
    PRINT SALARY.
```

would yield a sequence of observations of the values of Davies' salary. Even without knowledge of the distribution d one can easily estimate the width of a confidence interval on the sample mean, and continue sampling (repeating the query) until that interval is satisfactorily small. Unless there is threat monitoring that could detect such repetition (which is highly unlikely) it is imperative that for a particular record V_p be exactly repeated as long as the true value V_f is unchanged.

This repetition of value could be achieved is several ways. The deviate r_d will presumably be obtained from some programmed pseudo-random sequence. An algorithm that based the "seed" of that sequence on some function of V_f could produce identically the same r_d (and hence V_p) each time the process was applied. Alternatively, one could generate r_d only once, whenever a new value V_f is stored, and then store V_p directly in the record. This method would require additional space in the database, since there would be fields for both V_f and V_p in the record, but it would permit operation with no loss of efficiency for retrieval queries.

Value Dissociation

An interesting alternative to distorting the value of a field is to return an actual field value, but a value from the same field in some record different from the one selected by the query. For example, the query:

```
FOR ALL EMPLOYEES WITH DEPT = 'SALES',
     PRINT A LIST OF NAME, SALARY.
```

would produce a list of names and salaries, as expected. The salaries listed would be actual salaries of some employee -- but not, in general, the employee whose name is listed on the same line. The salary value is dissociated from the actual record it represents.

The process can be viewed in the following way:

$$V_p(i) = V_f(j)$$

where $V_p(i)$ is the value projected for the i^{th} record, and $V_f(j)$ is the actual field value for the j^{th} record. In general, $i \neq j$. This could be rewritten:

$$V_p(i) = V_f(i) + \{V_f(j) - V_f(i)\}$$

to emphasize the similarity to the value distortion model. If the record j is in some sense chosen randomly, then the difference

$$\{V_f(j) - V_f(i)\}$$

is the random deviate used to distort the true field value.

The dissociation strategy has some significant advantages over simple distortion:

1. The distribution from which the random deviates are obtained is automatically natural and appropriate in relation to the distribution of values of V_f.

2. Since actual values are returned, special properties of those values are preserved. For example, if all salaries are in even thousands of dollars, this fact would be preserved under dissocation.

3. The method is equally applicable to character-valued as well as arithmetic fields.

However, there are difficulties. Consider the set J from which record j is obtained. Presumably J should also contain record i. Consider two possible definitions for J:

1. J = C, the set of all records in the database.

2. J = Q, the set of all records selected by this particular query.

Suppose that J = C. In this case the selection condition of the query is essentially ineffective. It determines the size of the set of projected values, but has no effect whatever on their value distribution, since that is the distribution for the entire database regardless of the query. Therefore, regardless of the selection criterion specified, the expected value of the projection is identically the same. Hence, for all practical purposes, J must be Q, or some function of Q. Note that since the composition of Q is known only when a query is processed, this precludes any implementation in which a secondary dissociated value is stored in the record, as was suggested for the distortion model.

Essentially the same C versus Q issue exists for the distortion model also, in the choice of the distribution d. The analog to C is to have d fixed for a particular field for the entire database. The Q version would have d depend in some way upon the population of values V_f for the particular query. But in either case projected values are based on the V_f values for the selection set, so the selection criterion is effective. To simplify the implementation one would probably decide to hold d fixed over the database.

The repeatability question is also quite different under dissociation and distortion. Suppose that record j is selected from the finite population Q without replacement. Then, since the size of the projected set is exactly equal to the size of Q, the projected values are precisely an order-permutation of the values of Q. This would mean that this system would just be a minor inconvenience, but no real restriction upon the access capability of the user. For example, to determine Davies' salary in the face of such "protection", one would need only run two queries:

```
FOR ALL EMPLOYEES WITH DEPT = 'SALES' AND NAME ¬= 'DAVIES',
     PRINT A LIST OF NAME, SALARY.

FOR ALL EMPLOYEES WITH DEPT = 'SALES',
     PRINT A LIST OF NAME, SALARY.
```

The additional salary value produced by the second query, regardless of the name with which it is associated on the list, would be Davies' salary. Hence the sampling by which j is selected from J must be with replacement, so that values are not exactly repeatable. Once a particular record j has been selected from J, it must remain eligible for subsequent selection. This means that the set of projected values would not be exactly the same values as those of J, but instead some values would be repeated and some would be omitted. If there are n records in J, the probability that some particular record will appear in the projection set is given by:

$$1 - \left(\frac{n-1}{n}\right)^n$$

If n = 100, for example, this probability is only approximately 0.6. Penetration is no longer trivial, since the second query above can no longer be assumed to return the same values as the first query, plus one additional value. By chance, the second query could have fewer distinct values than the first. In general, the vulnerability of the dissociation model, with J = Q, and with selection from Q with replacement, is not easily assessed. We will examine the question in detail in a subsequent paper.

Value-Class Partition

A third strategy, unrelated to distortion and dissociation except for intent, is to partition the set of all possible values of a field into k disjoint, mutually-exhaustive classes. The type of partition that would be most common and useful in this context can be specified by a strictly increasing sequence of k values:

$$a_1 < a_2 < \ldots < a_k$$

with classes defined:

$$\text{Class } a_i: \quad a_i <= V_f < a_{i+1}$$

Classes can be defined for character-valued fields as well as arithmetic fields, since the system collating sequence provides a well-defined ordering over all possible values of the field.

A reasonable implementation would be to have the class-name (the lower-bound value) of the particular class into which the actual field value falls, returned instead of that field value.

For example, if the partition included a class:

Class 15000: $15000 <= V_f < 20000$

then in a projection field a salary of 17250 would be reported as 15000. For a selection field, the condition:

SALARY = 16000

would be interpreted:

"true if the value class of SALARY is the same as the value class of 16000; false otherwise."

The query:

FOR ALL EMPLOYEES WITH SALARY < 16000,
PRINT A LIST OF NAME, SALARY.

would be interpreted:

"For all employees whose salary falls in a value class lower than the class containing value 16000, print a list giving employee name and the class name of the class containing the salary value."

The degree to which actual field values would be masked from the user under such a system would depend upon the number of classes k, and the manner in which the classes are defined. The extremes would be k = 1, which corresponds to denial of access, and k such that each possible value is in a unique class, which corresponds to unrestricted access.

Implementation could be interpretive, involving a table-look-up over the class definitions each time the field is accessed. To avoid this execution overhead, an alternative implementation would be to determine the class membership once, whenever a new value is assigned, and to store the class name directly in the record. This would probably require that the partition definition be associated with a field and be common to all restricted users of that field, rather than allowing a separate definition for each user.

There does not appear to be any way, even with repeated queries, that unauthorized information could be extracted from a field protected in this manner.

Conclusions

The basic idea of partial access seems to be quite feasible and useful. Users could be granted sufficient access to perform various useful tasks, without compromising the security of individual field values. If such partial access is not provided, then either unnecessarily broad access will have to be granted to allow these tasks to be performed, or conversely, the tasks will have to be performed by unnecessarily high-level personnel.

Implementation of any of the strategies described here seems quite straightforward. For example, any of these strategies could be readily incorporated in the ASAP system, which already includes the capability of controlling the subset of fields accessible to each individual user. Presumably as this capability appears in other database systems, a partial access alternative could be included.

The most obvious strategy for partial access is distortion, which makes a query a statistical rather than deterministic process. No restriction against small selection sets is necessary. If values are repeatable, the system is invulnerable to assault by persistance. Very efficient implementations are possible, requiring only a secondary copy of the protected field, or the presence of a pseudo-random distorting routine in the access function. The most serious limitation is that the method is applicable only to arithmetic fields.

Dissociation is essentially a variation of distortion that has the minor advantage of displaying actual values, dissociated from the actual records, and the major advantage of being applicable to character-valued as well as arithmetic fields. However, the system must be carefully implemented with non-repeatable values or it will be trivially penetrable, and at best it is still vulnerable to persistent assault. The dissociation scheme must depend upon the particular query, and therefore must be implemented interpretively at some cost in execution efficiency.

The value-class partitioning strategy is simple and effective. It would be useful for a somewhat different type of query than either distortion or dissociation. It can be implemented either interpretively, or with a secondary field for the class-name in the record. In either case, the strategy is invulnerable to penetration.

References:

1. Attanasio, C. R., P. W. Markstein, and R. J. Phillips, "Penetrating an Operating System: A Study of VM/370 Integrity", IBM Systems Journal, Vol. 15, No. 1, 1976
2. ASAP System Reference Manual, Compuvisor Inc., Ithaca, N. Y., 1971
3. Conway, R, W. Maxwell, and H. Morgan, "Selective Security Capabilities in ASAP", SJCC 1972, pg 1181
4. CODASYL Data Base Task Group, October 1969 Report
5. Hansen, M. H., "Insuring Confidentiality of Individual Records in Data Storage and Retrieval for Statistical Purposes", FJCC 1971, pg 579
6. IBM: Information Management System, GH20-0765