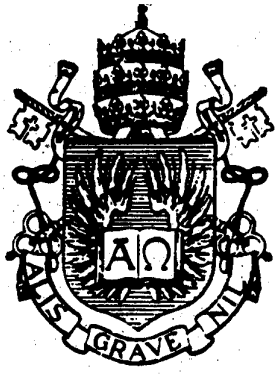


PUC

Series: Monografias em Ciência da Computação
Nº 26/77

COMPLETE AND COMPATIBLE SETS OF UPDATE OPERATIONS

by

K.C. Sevcik
A.L. Furtado

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 – ZC-19
Rio de Janeiro – Brasil

Series: Monografias em Ciência da Computação
Nº 26/77

Series Editor: Michael F. Challis December 1977

COMPLETE AND COMPATIBLE SETS OF UPDATE OPERATIONS*

by

K.C. Sevcik **
A.L. Furrado

* This research has been sponsored by FINEP and CNPq.

**Visiting associate Professor from the University of Toronto, Canada, under the sponsorship of the Canadian International Development Agency, and the National Research Council of Canada

For copies contact:

Rosane T.L. Castilho
Set. Doc. Inf.
Deptº Informática - PUC/RJ
Rua Marquês de São Vicente, 209 - Gávea
20.000 - Rio de Janeiro - RJ - Brasil.

ABSTRACT

We examine the problem of defining data base updates on user views so that all anticipated enterprise activities can be represented in the data base, but that certain declared constraints, which delimit the set of acceptable data base states, are preserved. In order to verify that a set of operations on views meets these criteria, it is necessary to consider the interactions of all users. Some updates must have side-effects not observable by the user requesting the update in order to preserve certain constraints.

KEYWORDS

Data bases, relational model, quotient relations, update operations, constraints, graph grammars.

RESUMO

Examinamos o problema de definir atualizações em bancos de dados através de visões dos usuários de modo que todas as atividades previstas da empresa possam ser representadas no banco de dados, mas que certas restrições declaradas, que delimitam o conjunto de estados aceitáveis do banco de dados, sejam preservadas. Para verificar que um conjunto de operações sobre visões atende a esses critérios, é necessário considerar as interações de todos os usuários. Algumas atualizações devem ter efeitos colaterais não observáveis pelo usuário solicitando a atualização para preservar certas restrições.

PALAVRAS CHAVES

Banco de dados, modelo relacional, relações quocientes, operações de atualização, restrições, gramáticas de grafos.

CONTENTS

1. INTRODUCTION

Various constant properties of an enterprise are reflected in its data base in that certain constraints are satisfied by the values of any acceptable configuration of the data base. The execution of update operations is governed by the requirement to obey such constraints in that each kind of update is valid only under certain conditions, and the update transforms the data base so that different conditions hold. Several devices have been suggested for handling these problems, such as views, assertions, triggers, request modifications, and transactions [1,2,3]. Here, we consider them together in an attempt to identify a complete, compatible set of update operations for a particular simplified data base environment.

To each data base user who needs to update the data base, we provide one or more views, which consist of derived relations, created on demand from the base relations, the representations of which are actually stored. Besides permitting the user to see only relevant portions of the data base, the views also act as screens on the update operations expressed through them. Acceptable update operations are allowed to change the stored representations of the base relations, while updates that would lead to the violation of some declared constraint are rejected. Re-derivation of a view after an acceptable update operation reveals that the intended change has occurred [4].

Starting with a description of the structure and operation of an enterprise, our goal is to define a complete and compatible set of update operations, along with views through which they are expressed. We consider a set of update operations that reflect the activities of an enterprise to be complete and consistent if and only if

(1) in any legal data base configuration, application of any update whose pre-requisite conditions are satisfied leads to another legal data base configuration,

and

(2) from any legal data base configuration, there is some sequence of allowable updates that leads to each other legal data base configuration.

The enterprise description from which we start includes

(1) the information to be stored in the data base, expressed as a set of base relations,

(2) the set of users who need to update the data base,

and (3) a textual description of the activities of the enterprise that necessitate data base updates, the conditions required for each activity to occur, and the specific responsibilities of each data base user.

In section 2, we present an example description of an enterprise, and in section 3, we specify a complete and compatible set of fifteen update operations, and the views through which they can be expressed. We briefly suggest how views that help preserve constraints can be implemented. In section 4, we argue the compatibility and completeness of the set of fifteen operations. Section 5 includes discussion and conclusions.

2. A SIMPLE DATA BASE ENVIRONMENT

As an example data base environment, we consider the personnel segment of a small manufacturing enterprise. While the example is intended to suggest realism, it is highly simplified, and certainly does not cover the breadth of situations that may arise in more detailed enterprise descriptions.

A. The Base Relations

The attributes treated in our example are:

N - name of employee
S - salary
J - job title
K - skill
T - task
P - project
L - leader of a project

The base relations which represent the relationships among entities with these attributes are:

EMP(N,S,J) - employee's name, salary and job title

REQ(T,K) - requirement of a skill to do a task

ASN(N,T,P) - assignment of an employee in a project to a task

MNG (P,L) - management of a project by a leader

CAP(N,K) - capabilities (skills) possessed or acquired by employees

Figure 1 indicates the presence of attributes in relations diagramatically.

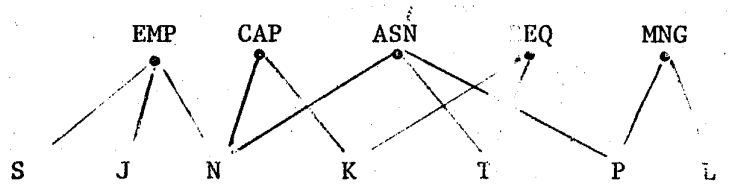


Figure 1. The base relations and their attributes.

B. The Users

The users authorized to perform update operations are the employees holding the positions

personnel manager,
engineering manager,
training manager,
and leader of some project.

C. The Activities of the Enterprise

The personnel manager hires employees by associating a salary (at least the minimum wage), and a job title with their name, and may fire employees, but only if they are not currently assigned to any project.

The engineering manager initiates new projects by specifying their names and the name of the initial leader of each. He may replace the leader of a project, or suspend a project by leaving it with no leader. No employees may continue to be assigned to a project that is suspended.

A suspended project may be permanently terminated by the engineering manager, or may be restarted by assigning a new leader.

Various tasks compose each project, and the engineering manager is responsible for indicating what skills are required of an employee to perform each task. The engineering manager also associates employees with projects (but not suspended ones), and terminates such associations.

Employees acquire new skills through training, but lose old skills through lack of use or changing technology. The training manager is responsible for recording the skills currently possessed by each employee.

Each leader of a project determines how employees associated with his project are assigned to tasks. An employee must possess all the skills required for each task assigned to him.

3. THE VIEWS AND UPDATE OPERATIONS

While it is impossible to anticipate all future queries in most data base environments, it is possible, and necessary for reasons of integrity, to enumerate the set of all allowable update operations. Although subject to occasional change, this set is generally quite static, despite the creation of new applications for the data base and the disappearance of old applications.

In this section, we identify a set of views and update operations expressed in terms of those views for the example data base environment described in section 2. The set of views and operations are designed to guarantee the preservation of all indicated constraints on legal data base configurations, while still permitting each user enough latitude to fulfill his responsibilities.

A. The Update Operations

For each operation, we specify the operation, the (only) person authorized to use the operation, the view through which the operation is expressed, and the result of the operation, including any conditions on which the permissibility of the operation depends and any additional actions (side-effects) caused ("triggered"[1]) by the operation. Also, we give a diagram that indicates the effect of the operation. The diagrams show the initial configuration and the resulting situation on each side of the arrow. A $\emptyset$ on the left or right means respectively that there are no requirements of the initial configuration, or that all tuples shown in the initial configuration are deleted in the final configuration. Relation names with arcs to attribute values represent tuples in the relations. Ordinary arcs mean that a defined attribute value must be there, while slashed arcs are used for attributes that must not be there. Missing arcs, when there is not an arc for every attribute, lead to arbitrary values, and dotted arcs connect to tuples that may or may not be present; slashed dotted arcs are for tuples (again that may or may not be present) not connected to the given attribute value. Whenever possible, attribute values are indicated by the lower case of the attribute name. In ambiguous cases, the arcs are labelled by an attribute name. Primes indicate distinct attribute values. Converging lines from two tuples on a single attribute value indicate that the value must be the same in both tuples, curly brackets show sets of tuples, and the "undefined" value is denoted by "*".

1. Hire employee n with salary s and job title j

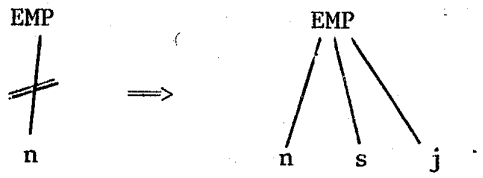
Authorized User: Personnel Manager

View: V_{hire}, which is EMP restricted to permit only salaries that meet or exceed the minimum wage.

Operation: Insert (n,s,j) in to V_{hire}.

Result: If no other employee in EMP has name n and s is greater than or equal to the minimum wage, then the tuple is entered into EMP. Otherwise, the update is rejected.

Diagram:



(Note: The salary condition is not indicated in the diagram)

2. Initiate project p with leader ℓ

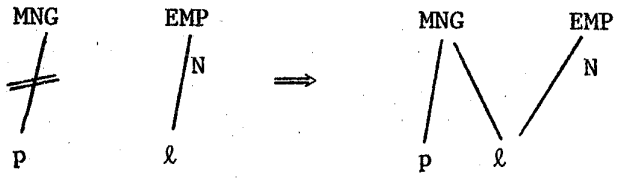
Authorized User: Engineering Manager

View: $V_{initiate}$, which is MNG

Operation: Insert (p,ℓ) into $V_{initiate}$

Result: If ℓ is an employee and there is no project in MNG with name p, then the tuple is entered into MNG. Otherwise, the update is rejected.

Diagram:



3. Require skill k for task t

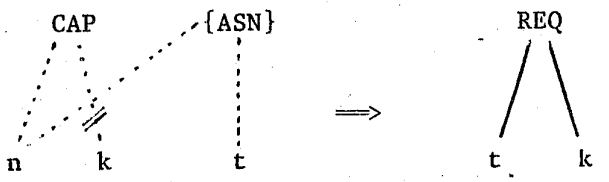
Authorized User: Engineering Manager

View: $V_{require}$, which is REQ

Operation: Insert (t,k) into $V_{require}$

Result: The tuple is inserted into REQ unless it is already there, and any assignment of an employee not possessing skill k to task t is deleted from ASN.

Diagram:



4. Acquire skill k by employee n

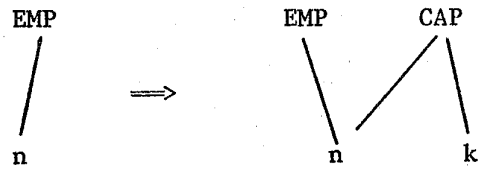
Authorized User: Training Manager

View: $V_{acquire}$, which is CAP

Operation: Insert (n,k) into $V_{acquire}$

Result: If n is an employee and the tuple is not already present, the tuple is inserted into CAP. Otherwise, the update is rejected.

Diagram:



5. Associate employee n with project p

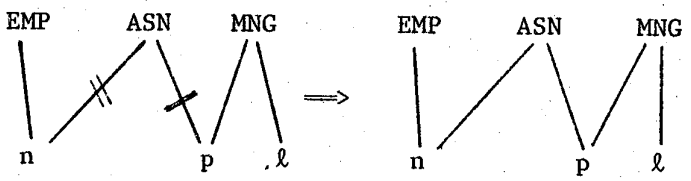
Authorized User: Engineering Manager

View: $V_{associate}$, which is ASN projected on N and P.

Operation: Insert (n,p) into $V_{associate}$

Result: If p is a project that currently has a leader and n is an employee, and no tuple of ASN currently has name n and project p, then the tuple (n,*,p) is entered into ASN. Otherwise, the update is rejected.

Diagram:



6. Assign to employee n task t in project p

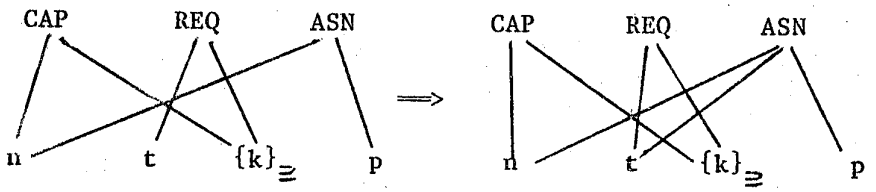
Authorized User: Leader of project p

View: $V_{(assign,p)}$, which is ASN restricted to retain only those tuples with project p and then projected on N and T.

Operation: Insert (n,t) into $V_{(assign,p)}$

Result: If some tuple of ASN has name n and project p, and the set of skills associated with t in REQ is a subset of the set of skills associated with n in CAP, then the tuple (n,t,p) is inserted into ASN (and if the tuple (n,*,p) is in ASN, it is removed). Otherwise, the update is rejected.

Diagram:



where $\{k\}_{\geq}$ indicates that that the set of values of k in REQ must be a subset of the set of values of k in CAP.

7. Replace the leader of project p with employee ℓ

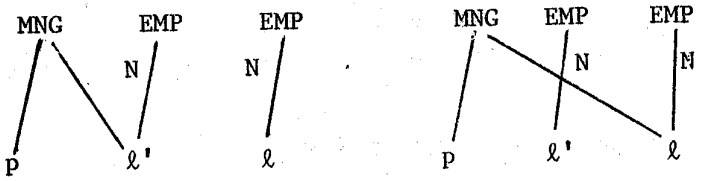
Authorized User: Engineering Manager

View: $V_{replace}$, which is MNG

Operation: Modify any tuple with project p in $V_{replace}$ by L ← ℓ

Result: If p is a project with a leader and ℓ is an employee, then the tuple for project p in MNG is modified so that ℓ is the leader. Otherwise, the update is rejected.

Diagram:



8. Suspend project p

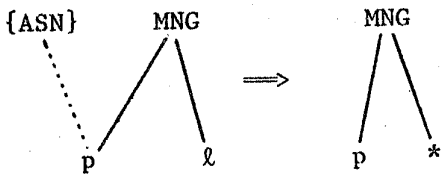
Authorized User: Engineering Manager

View: $V_{suspend}$, which is MNG

Operation: Modify any tuple with project p in $V_{suspend}$ by L ← '*'

Result: If p is a project with a leader, then the tuple for project p in MNG is changed to (p,*), and all tuples with project p in ASN are deleted. Otherwise, the update is rejected.

Diagram:



9. Restart project p with leader ℓ

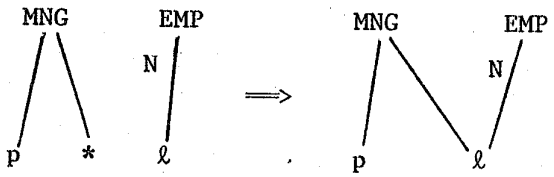
Authorized User: Engineering Manager

View: $V_{restart}$, which is MNG

Operation: Insert (p,ℓ) into $V_{restart}$

Result: If p is a leader-less project (so that (p,*) appears in MNG) and ℓ is an employee, then the tuple (p,ℓ) replaces the tuple (p,*). Otherwise, the operation is rejected.

Diagram:



10. Release employee n from task t in project p

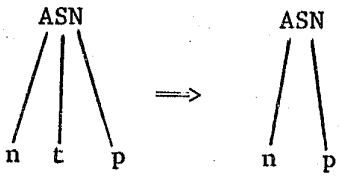
Authorized User: Leader of project p

View: $V_{(release,p)}$, which is ASN restricted to tuples for project p then projected on N and T.

Operation: Delete (n,t) from $V_{(release,p)}$

Result: If (n,t,p) is a tuple in ASN, then it is deleted. If no other tuple in ASN has employee n and project p, then the tuple (n,*,p) remains in ASN. Otherwise the operation is rejected.

Diagram:



11. Disassociate employee n from project p

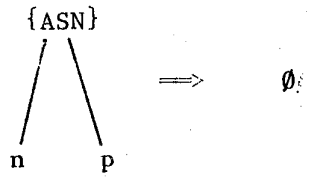
Authorized User: Engineering Manager

View: $V_{disassociate}$, which is ASN projected on N and P.

Operation: Delete (n,p) from $V_{disassociate}$.

Result: If there are any tuples in ASN with employee n and project p, then they are all deleted. Otherwise, the update is rejected.

Diagram:



12. Lose skill k of employee n

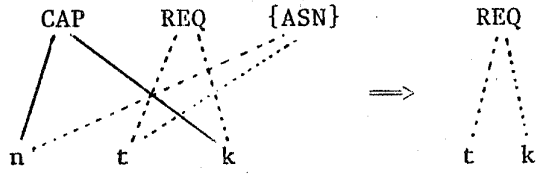
Authorized User: Training Manager

View: V_{lose}, which is CAP

Operation: Delete (n,k) from V_{lose}

Result: If (n,k) is a tuple in CAP, then it is deleted, and any tuple in ASN with employee n and a task that is associated with k in REQ is also removed. Otherwise, the update is rejected.

Diagram:



13. Remove the requirement of skill k for task t

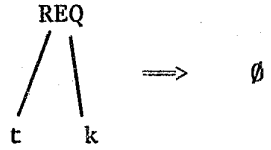
Authorized User: Engineering Manager

View: V_{remove}, which is REQ

Operation: Delete (t,k) from V_{remove}

Result: If tuple (t,k) is in REQ, then it is deleted. Otherwise, the update is rejected.

Diagram:



14. Terminate project p

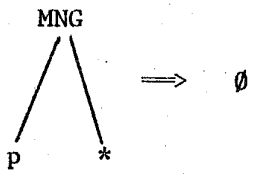
Authorized User: Engineering Manager

View: $V_{terminate}$, which is MNG

Operation: Delete any tuple with project p from $V_{terminate}$

Result: If the tuple (p,*) is in MNG (i.e., p is a leader-less project), then the tuple is deleted. Otherwise , the update operation is rejected.

Diagram:



15. Fire employee n

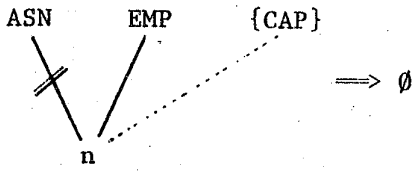
Authorized User: Personnel Manager

View: V_{fire}, which is EMP projected on N

Operation: Delete (n) from V_{fire}

Result: If a tuple with name n is in EMP, then it is deleted, together with his skills. Otherwise, the update is rejected.

Diagram:



The definitions above are reminiscent of the notion of abstract data types, which suggests that some mechanism resembling clusters might be used in an implementation [9].

B. The Views and Their Implementation

While the previous section defined a different view for each update operation, each user actually updates and observes only a few distinct derived relations. Figure 2 summarizes the use of derived relations by the four users. The notation in which the views are expressed is the standard notation of Codd's relational algebra: square brackets containing attribute names denote projection on those attributes, and square brackets containing a condition denote restriction if after a single relation name and denote join if between two relation names.

Note that figure 2 mentions only the base relations involved in the derivation of the views for each user. The scope of each operation includes all base relations involved in checking pre-conditions or completing triggered actions. The base relations in the scope of each operation are indicated in the diagrams given previously. Examples of derivation expressions capable of checking the pre-conditions are given elsewhere [5].

The views are designed to reveal to each user only the information he needs, and to permit him to change only what he is responsible for changing. Note that neither the Engineering Manager nor the Training Manager are able to learn the salaries of employees.Similarly, each Project Leader knows nothing of any employees other than those associated with his project. This effect is achieved with both the EMP and CAP relations by joining them to ASN on the N attribute, then eliminating tuples for other projects by a restriction, then projecting on the original attributes of the relation. While the Personnel Manager could fire employees with a more limited view containing only the N attribute, we choose to use the same view as for hiring in order to avoid the proliferation of views.

In order to use views to help preserve constraints, the transformations of updates on views to corresponding updates on the base relations must be carefully designed. The transformations are specific to the set of operations used to derive views. For the case in which the set of operations is that of the algebra of quotient relations [6], the problem of transforming updates has been studied in detail [5] . Here, we only summarize the general approach.

User	Purpose	Derived Relation	Action
Personnel Manager	Updates	EMP [S ≥ min.wage]	to HIRE and FIRE
	Observes	ASN [N,P]	to check that an employee is not associated with a project before firing
		CAP	to check skills before firing
Engineering Manager	Updates	MNG	to INITIATE, REPLACE, SUSPEND, RESTART, and TERMINATE
		REQ	to REQUIRE and REMOVE
		ASN [N,P]	to ASSOCIATE and DISASSOCIATE
	Observes	EMP [N,J]	to know job titles of employees
		CAP	to know skills of employees
Training Manager	Updates	CAP	to ACQUIRE and LOSE
	Observes	EMP [N,J]	to know job titles of employees for planning training
		ASN [N,T] REQ	to know assignments of employees and to learn what skill are being used
Project Leader (of project p)	Updates	(ASN [P=p]) [N,T]	to ASSIGN and RELEASE
	Observes	((EMP [N=N] ASN) [P=p]) [N,S,J]	to know job titles and salaries of employees associated with his project
		REQ	to know skills required for tasks
		((CAP [N=N] ASN) [P=p] [N,K]	to know skills of employees associated with his project

Figure 2. Derived Relations Updated and Observed By Each User.

Taking the operations by which views are derived one at a time, it is possible to specify appropriate base relation updates for each update operation expressed on a view that is derived by the chosen operation from the base relations. By applying such transformation rules iteratively, the base relation updates for update operations on any view can be determined. A transformation is considered to be appropriate only if re-derivation of the view through which the update is expressed reveals that the requested change in the view (and no other visible change) has occurred.

By carefully designing the sequence of operations by which a view is derived, the preservation of some constraints can be guaranteed. The constraint is either enforced directly by intercepting any operation that would lead to a violation of the constraint, or is checked retrospectively after the update is carried out, and, if necessary to preserve the constraint, the base relations are restored to their state before the update.

The constraint that all salaries must be at least the minimum wage can be enforced directly since the restriction on the Personnel Manager's view (see figure 2) makes it impossible for him to successfully insert a tuple with a salary less than the minimum wage into the EMP relation. The constraint that employee names (or project names) be unique can be checked using a view slightly more complex than those shown in figure 2. If the Engineering Manager's, view constrains the MNG relation to have at most one tuple with any single name, then an attempt to initiate a project with the name of an existing project would fail. (In the algebra of quotient relations, such a constraint is expressed as

$$V_{initiate} = (MNG/\{P\})\ [\#L=1]$$

which requires that, when projects of the same name are grouped together, there is exactly one project in each group).

4. THE COMPATIBILITY AND COMPLETENESS

In this section we argue that the operations and views defined in section 3 allow the data base users to carry out all their responsibilities, but prevent them from inadvertently or maliciously causing the data base to violate one of the stated constraints. Figure 3 shows how the four users depend on and influence one another. With appropriate cooperation among users and an appropriate sequence of operations, every tuple entered into the data base can later be deleted. In fact, the initial state of the data base (all relations having no tuples) can be reached by some sequence of operations from any data base state that conforms to the constraints. Also, from any state conforming to the constraints, some sequence of operations leads to a state in which each of the operations is applicable. Such sequences are indicated in Figure 4. Note that ASSOCIATE and ASSIGN require respectively two and three sequences of operations to guarantee their applicability. From the properties mentioned above, we see that the set of data base states is closed under the operations, and all data base states that obey the constraints can be reached by an appropriate sequence of operations.

As well as the completeness indicated above, the fifteen operations defined in section 3 are also compatible in that no sequence of operations can lead from a data base configuration that satisfies all constraints to one that does not. A detailed argument to this effect is tedious since each constraint must be treated individually. Here, we exemplify the general approach by discussing a few of the constraints.

Consider first the constraint that no project without a leader can have employees associated with it. This constraint is preserved because ASSOCIATE, the only operation that creates an association between an employee and a project, requires that the project have a leader, and SUSPEND, the only operation that converts a project with a leader to a leader-less one, requires that all associations of employees to that project be broken. Since the project leader can be changed without severing the ties to all employees, the REPLACE operation is provided specifically for that purpose.

Next, consider the constraint that an employee can be assigned to a task only if he possesses all skills required for the task. The ASSIGN operator only assigns a task to an employee if, at that time, he has all the skills required for the task. The REQUIRE operation reviews all assignments to the task for which an additional skill has been required, and ends assignments in which

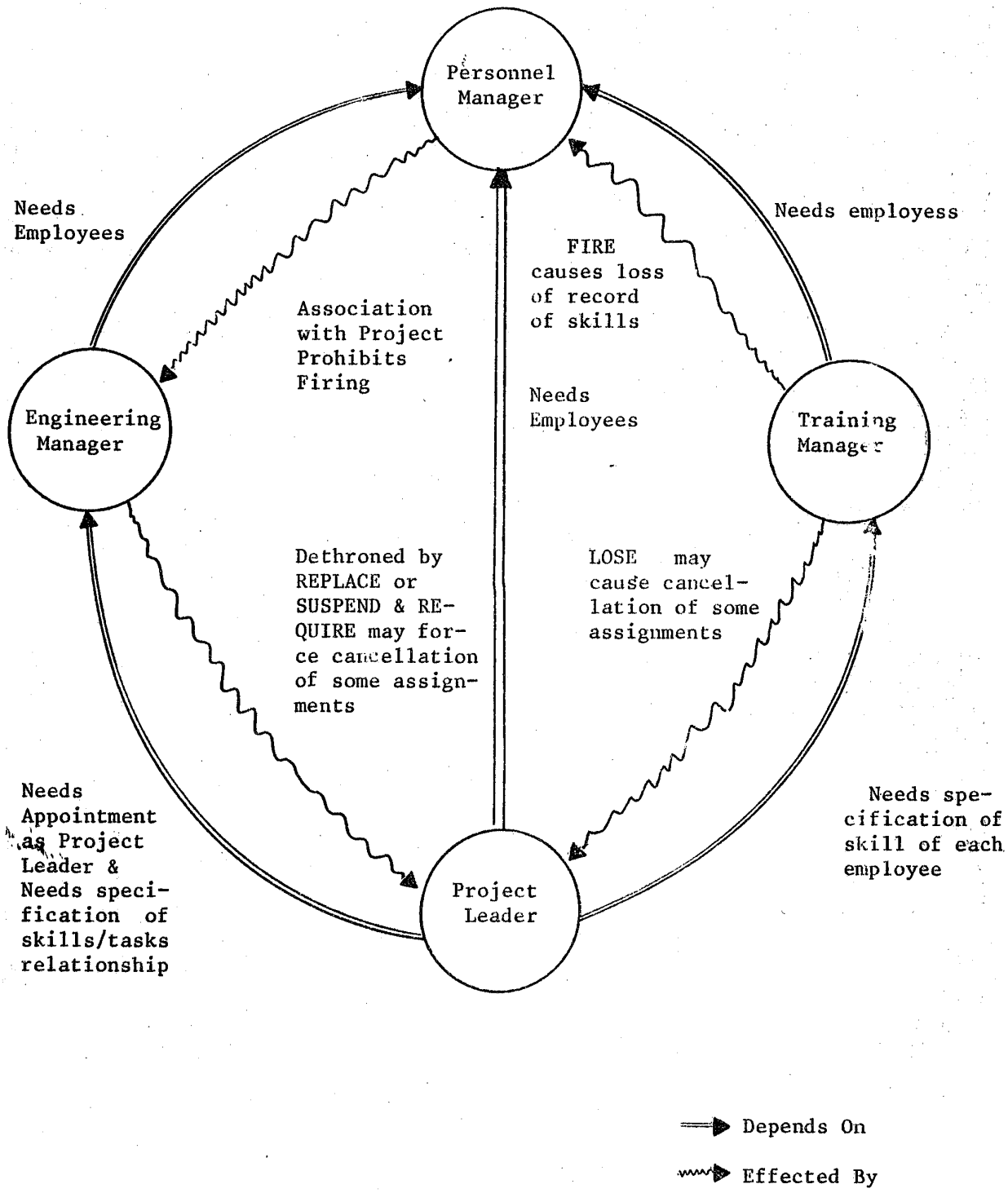


Figure 3. Interactions and Interdependencies Among the Users.

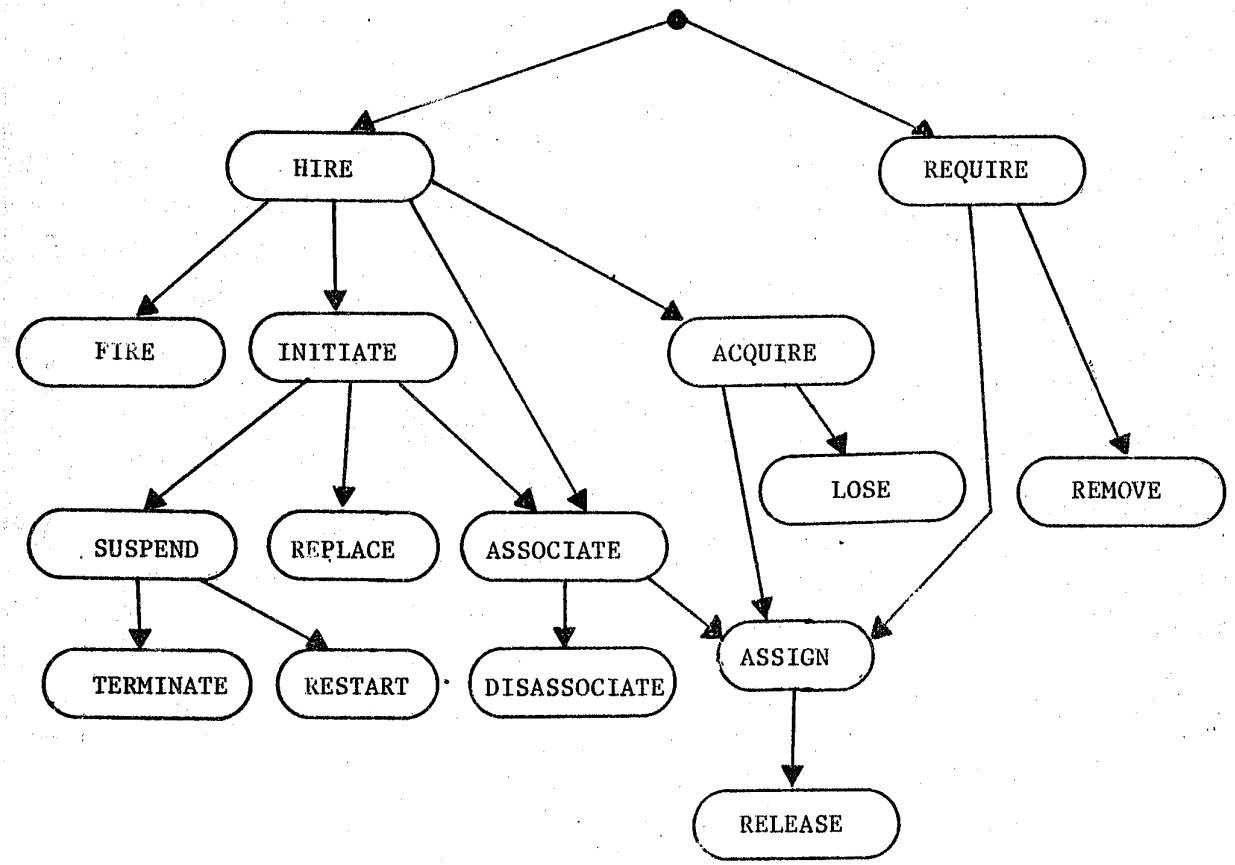


Figure 4. Sequences of operations that can start from any allowable data base state

the employee does not possess the newly required skill.

Similarly, the LOSE operation ends assignments for which, due to the lost skill, an employee is no longer fit.

Finally consider the constraint that project leaders may end assignments, yet only the Engineering Manager is authorized to break the association of an employee to a project. Due to the choice of base relations, both relationships are represented in the relation ASN. Thus, it is critical that RELEASE not delete the last tuple relating an employee to a project. Instead, the task

The argument for compatibility presented above is informal. With formal definitions of the operations and the constraints, it might be possible to develop a technique of mechanically verifying compatibility and completeness. The diagram representations of the operations suggest that graph grammar proof methods [6,7] might provide a basis for such mechanical verification. A different approach to assuring the semantic integrity of data bases is taken by Brodie [10]. He defines a schema specification language that is intended to facilitate the verification of a general form of constraints.

5. CONCLUSIONS

We have described, using an extended example, how declared constraints on states of a data base can be automatically preserved. The responsibilities of various users are considered and the logical structure of the relevant information is specified, then the set of update operations and the corresponding data base views are developed. With appropriately defined views, many constraints can be either automatically enforced, or at least automatically checked.

By starting with a textual description of an enterprise and its operation, we risk ambiguity and uncertainty in identifying the constraints. Such ambiguities are generally discovered during the process of defining the operations, so some clarifications of the description may be required at that time. Such iterative clarification of constraints is typically required in the development of any actual data base system.

In the textual description of the enterprise in section 2, an example of an ambiguity is the interpretation of the requirement that employees be assigned only to tasks for which they possess all the required skills. This could mean either that the skills must be possessed at the time the employee is assigned to the task, or that, at all times during the assignment of an employee to a task he must currently possess all skills currently required for the task. In section 3, we took the latter (stricter) interpretation, and the REQUIRE and LOSE operations were given the potential for revoking assignments. Thus, the Engineering Manager and Training Manager can indirectly cause assignments of employees to tasks to be ended. If we take instead the less restrictive interpretation that skills are checked only when assignments are made, then only SUSPEND, RELEASE and DISASSOCIATE are capable of directly or indirectly ending the assignments of employees to tasks. Only the REQUIRE and LOSE operations would be changed under this interpretation. For each, the requirement to check previous assignments would be removed.

In some cases, constraints are preserved by defining operations with secondary effects. A concurrency mechanism must insure that users never observe the data base at times when a transaction (operation plus secondary actions) is started but not completed. While queries do not complicate the evaluation of completeness and compatibility, they do interfere in that they compete for data base resources, and they are delayed while data base updates are in progress.

REFERENCES

1. Astrahan, M.M., et. al., System R: Relational approach to database management, TODS 1,2 (June, 1976), 97-137.

2. Stonebraker, M., et. al., The Design and implementation of INGRES, TODS 1,3 (September 1976), 189-222.

3. Date, C.J., An Introduction to Database Systems, second edition, Addison-Wesley, London, 1977, p. 400.

4. Paolini, P. and G. Pelagatti. Formal definition of mappings in a data base, Proc. ACM-SIGMOD, 1977, 40-46.

5. Furtado, A.L., and K.C. Sevcik, Data base update through views, technical report, Depto. de Informatica, Pontificia Universidade Católica do Rio de Janeiro, 1977 (submitted for publication).

6. Furtado, A.L., and L. Kerschberg, An Algebra of quotient relations, Proc. ACM-SIGMOD, 1977, 1-8.

7. Montanari, U.G., Separable graphs, planar graphs, and web grammars, Information and Control 16 (1970), 243-67.

8. Gotlieb, C.C., and A.L. Furtado, Data Schemata based on directed graphs, to appear in International Journal of Computer and Information Sciences, 1978.

9. Liskov, B. and, S. Zilles, Programming with abstract data types, Proc. of SIGPLAN Symposium on Very High Level Languages, March, 1974.

10. Brodie, M.L., A Formal approach to the specification and verification of semantic integrity in data bases, Ph.D. thesis, in preparation, Dept. of Computer Science, University of Toronto.