Efficient View Self-Maintenance

A TECHNICAL REPORT

Nam Huyn
Stanford University
huyn@cs.stanford.edu

Abstract

We revisit the problem of finding a maximal test that guarantees that a view can be maintained using only the view, its definition and the update, regardless of the actual database instance. These maximal tests, we call *Complete Tests for View Self-Maintainability* (abbrev CTSM), are also known in the literature as necessary and sufficient conditions for *Conditionally Autonomously Computable Updates* ([TB88]). This paper focuses on finding an efficient evaluation of self-maintainability (abbrev SM). Our working hypothesis is that at least for conjunctive-query (abbrev CQ) views (i.e. select-project-join views with only equality comparisons), SM evaluation can be separated into a view-definition-time portion where a maximal test is generated solely based on the view definition, and an update-time portion where the test is actually applied to the view and the update. We also conjecture that these tests can be expressed as queries over the view in some traditional query language. The obvious practical significance would be that CTSM's not only can be precomputed but can also be optimized using traditional query optimizers, thus minimizing work that needs to be done at update time.

Unlike previous work, we use an alternative formulation of SM and a different approach for deriving CTSM. This paper reports on some interesting new results for insertion updates and CQ views: 1) the CTSM's are extremely simple queries that look for certain tuples in the view to be maintained; 2) these CTSM's can be generated at view definition time using a very simple algorithm based on the concept of *Minimal Z-Partition*; 3) view self-maintenance can also be expressed as simple update queries over the view itself.

1 Introduction

In this paper, we consider the problem of determining self-maintainability (abbrev SM) of views expressed as conjunctive queries over base relations. That is, given a view definition specified as a conjunctive query Q, a materialized view V defined by Q over database D, and an update μ to the base relations in D, we would like to find a test:

- That only looks at the view definition Q, view V and update μ,
- That determines whether or not $Q(D^\mu)$ [1] only depends on V and μ, regardless of the actual database D, assuming that $V = Q(D)$,
- And that is maximal in the sense that when the test answers negatively, there are database instances D_1 and D_2 that are both consistent with V but such that $Q(D_1^\mu) \neq Q(D_2^\mu)$.

Example 1.1 Consider the view definition:

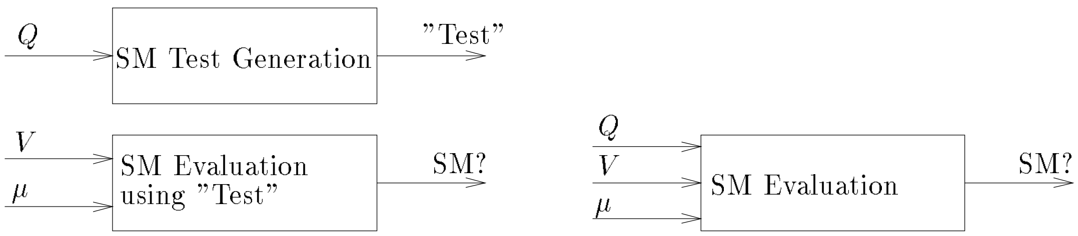
$$Q : v(X,Y) \text{ :- } r(X,Y) \ \& \ t_1(X) \ \& \ t_2(Y)$$

[1] D^μ denotes the result of applying μ to D.

and the insertion of tuple (a, b) into the base relation represented by predicate r. While the test $(a, b) \in V$ guarantees that the view can be self-maintained (since (a, b) is already in r and thus V does not need to change), it is not maximal since the test $(a, -) \in V \wedge (-, b) \in V$ [2] also guarantees self-maintainability of the view. Indeed, if the latter test is satisfied, the database must already contains $t_1(a)$ and $t_2(b)$, and inserting (a, b) into V is sufficient to bring the view up to date. ∎

We call such maximal tests *Complete Tests for Self-Maintainability* (abbrev CTSM). This problem has been studied in [TB88] and more recently in [GB95] for views that are conjunctive queries with arithmetic comparisons (aka select-project-join queries) but with single occurrence of predicates (i.e. no self-joins). Results from [TB88] and [GB95] gave necessary and sufficient conditions for *Conditionally autonomously Computable Updates* (abbrev CAU). We would like to point out that the two notions SM and CAU are equivalent, even though [TB88] and [GB95] only explicitly mentioned that SM follows from CAU.

The approach used in [TB88] and [GB95] has two main disadvantages: efficiency of determining SM evaluation is not well understood, and construction of the test and its execution are intermingled, forcing most of SM evaluation to be done at update time.



Our approach **Previous approach**

Figure 1: Separating SM test generation from SM test evaluation

In this paper, we explore the hypothesis that for conjunctive-query views at least, separation of SM evaluation into a view-definition-time portion and an update-time portion is possible. That is, a complete test can be constructed from the view definition alone. Figure 1 contrasts our approach with previous approach. Furthermore, the complete test can be efficient to execute, such as a database query.

The main result of this paper essentially confirms our hypothesis:

- For CQ views and insertion updates, view definitions have a very simple characterization using the concept of *Minimal Z-Partition* (Section 3.)
- Using this characterization, we derive CTSM's that turn out be simple queries on the view that essentially look for certain tuples. We found a class of CQ views where no SM evaluation is ever needed, simply because no view in this class is self-maintainable (Section 4.)
- View self-maintenance, expressible as simple queries, are also given (Section 4).

[2] $(a, -) \in V$ is a shorthand for the query $(\exists Y)(a, Y) \in V$. Similarly, $(-, b) \in V$ denotes $(\exists X)(X, b) \in V$.

2 Preliminaries

2.1 Notation, terminology and assumptions

Throughout the rest of this paper, the definition of a conjunctive-query view is represented as follows:

$$Q : v(\bar{X}', \bar{U}', \bar{Z}') :- r(\bar{X}, \bar{U}) \ \& \ S(\bar{U}, \bar{Z}) \qquad (1)$$

where $\bar{U}$, $\bar{X}$ and $\bar{Z}$ denotes sets of variables, $\bar{X}'$, $\bar{U}'$ and $\bar{Z}'$ denote subsets of $\bar{X}$, $\bar{U}$ and $\bar{Z}$ respectively, r is the predicate for the updated relation, and S denotes a conjunction of subgoals.

$\bar{U}$ represents the *join* variables, i.e., the variables shared between the subqueries $r(\bar{X}, \bar{U})$ and $S(\bar{U}, \bar{Z})$. From the point of view of S, we also call $\bar{U}$ the *distinguished* variables (while we call $\bar{Z}$ the *nondistinguished* variables).

It is important not to confuse our definition of distinguished variables with that commonly used to designate those variables that are used in the head (i.e. those variables that are not "projected out"). We call the latter variables *exposed* and use ' to denote them. Thus, $\bar{X}'$ denotes the variables in $\bar{X}$ that are exposed. Variables that are not exposed are called *hidden*.

We sometimes call the variables in $\bar{X}$ the X-variables, $\bar{U}$ the U-variables, and $\bar{Z}$ the Z-variables.

Finally, we assume all predicates in the body have single occurrences, that is, no self-joins are allowed.

Example 2.1 The view definition $v(X, Z, T) :- r'(Y, X, 3, Z) \ \& \ s'(Z, T, Y, Z, 5)$, where r' and s' are base relation predicates, is represented in our notation as

$$v(X, Z, T) :- r(X, Y, Z) \ \& \ s(Y, Z, T)$$

where the original subgoals are normalized (i.e., constant symbols are removed, multiple occurences of variables consolidated, variables reordered) using predicates r and s. In (1)'s notation, $\bar{U}$ represents the join variables $\{Y, Z\}$, $\bar{U}' = \{Z\}$, $\bar{X} = \bar{X}' = \{X\}$, $\bar{Z} = \bar{Z}' = \{T\}$. ∎

We will use D, D_1 and D_2 to denote database instances, and D^μ, D_1^μ and D_2^μ the respective instances that result from applying update μ.

2.2 Definition of self-maintainability

Given a view definition Q, a view V defined by Q over some database D and an update μ on D, we say that view V is self-maintainable (SM) if the new view that results from update μ is independent of the underlying database. The following more formally defines the SM notion.

Definition 2.1 (Self-Maintainability): View V is self-maintainable if $Q(D^\mu)$ is the same for every database instances D such that $Q(D) = V$. ∎

Proposition 2.1 *SM and CAU are equivalent.* ∎

Proof: [TB88] showed that CAU implies SM. Now, we show that SM implies CAU as well. SM says that $Q(D^\mu)$ is independent of D, provided that D is consistent with V. So to compute $Q(D^\mu)$, we can choose the "canonical" database $D_c = Q^{-1}(V)$ obtained as follows: each tuple in V binds the variables in the head; these bindings are extended to the hidden variables in the body by binding them to new constants. One can easily show that $Q(D_c)$ is identical to V. Computing $Q(D^\mu)$ is reduced to computing $Q(D_c^\mu)$. We just defined a function that takes V as input and computes the new view that is consistent with the updated database. Thus the update is also conditionally autonomously computable. ∎

Equivalence between SM and CAU justifies our use of SM throughout the rest of this paper, which we find more natural for our purpose.

2.3 Approach to finding CTSM's

The approach we take can be summarized as follows:

- Find a syntactic characterization of Q for the purpose of deriving CTSM's.
- Based on a specific characterization of Q, find a test condition that typically looks for the existence of certain tuples in V.
- Verify that the condition is sufficient for SM by showing that for any database D such that $Q(D) = V$, $Q(D^{\mu})$ does not depend on D.
- Verify that the condition is necessary for SM by finding an appropriate counterexample consisting of database instances D_1 and D_2. Typically, D_1 is some canonical minimal database that is consistent with V. D_2 is typically obtained by introducing some perturbation (to be found) to D_1 that is sufficiently small to maintain consistency with V but sufficiently large to assure that $Q(D_2^{\mu})$ is different from $Q(D_1^{\mu})$.

2.4 View self-maintainability vs. self-maintenance

When a view is self-maintainable, how do we determine the actual updates to the view that will make it consistent with the updated base relations?

By definition of SM, $Q(D^{\mu})$ does not depend on D as long as $Q(D) = V$. In the worst case, we can pick D arbitrarily (e.g. the canonical database consistent with V), apply the update μ to D to obtain D^{μ} and run the query Q over D^{μ}. However, we can do much better: the required updates to view V can typically be derived from sufficiency proofs of the SM condition.

3 Minimal Z-Partition

Example 3.1 Consider the view definition:

$$v(X,Y) :- r(X,Y) \; \& \; t(X,Y)$$

and the definition from Example 1.1. Consider the insertion of $r(a,b)$ and the problem of maintaining some view V defined by either the query above or the query in Example 1.1. One can easily verify that while the condition $(a,b) \in V$ is a complete test for self-maintainability (CTSM) in the first case, it is no longer a necessary condition for SM in the second case. We mentioned in Example 1.1 that a CTSM in the second case is indeed given by the condition $(a,-) \in V \wedge (-,b) \in V$. ∎

Example 3.1 suggests that in general, the CTSM for views defined by

$$v(\bar{X}', \bar{U}', \bar{Z}') :- r(\bar{X}, \bar{U}) \; \& \; S(\bar{U}, \bar{Z})$$

is not independent of the actual shape of S. But how do we syntactically characterize $S(\bar{U}, \bar{Z})$ for the purpose of finding a CTSM? In the rest of this section, we develop the tool for characterizing S that will be used in later sections.

Definition 3.1 (Minimal Z-Partition): Let $S(\bar{U}, \bar{Z})$ be a conjunction of subgoals with distinct predicates where certain variables are designated as "distinguished" ($\bar{U}$ in our notation) and the remaining variables as "nondistinguished" ($\bar{Z}$ in our notation). A Z-partition for $S(\bar{U}, \bar{Z})$ is a partition of the subgoals into groups such that no two groups share the same Z-variable. A minimal Z-partition is a Z-partition such that further partitioning is not possible without introducing groups sharing the same nondistinguished variables. ∎

Example 3.2 Consider the conjunction

$$s_1(U,V)\ \&\ s_2(V,Z)\ \&\ s_3(W,Z)\ \&\ s_4(T)$$

where the distinguished variables are U, V and W. The subgoals can be partitioned into three groups: $\{s_1(U,V)\}$, $\{s_2(V,Z), s_3(W,Z)\}$, $\{s_4(T)\}$. This Z-partition is minimal and is illustrated in Figure 2 in connection hypergraph form. In Figure 2, the distinguished variables are in boldface, and each group in the partition is represented by hyperedges with the same shading. ▮

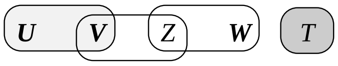


Figure 2: The three groups in the minimal Z-partition in Example 3.2

Example 3.3 Consider another conjunction

$$s_1(U,Z)\ \&\ s_2(V,Z)\ \&\ s_3(Z,T)\ \&\ s_4(W,T)$$

where the distinguished variables are U,V and W. This time, the minimal Z-partition consists of only one group that includes all the subgoals, and is depicted in Figure 3. ▮

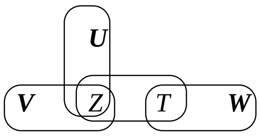


Figure 3: Only one group in the minimal Z-partition in Example 3.3

Properties of minimal Z-partitions

- A minimal Z-partition always exists and is unique.
- Any group having no nondistinguished variables is a singleton that consists of some subgoal that uses no nondistinguished variables.
- Any group having some nondistinguished variables consists of subgoals that are all "interconnected" by nondistinguished variables. That is, suppose we cannot remove a subgoal from the group without also removing any other subgoal that can join with it via some nondistinguished variable. Then removing any subgoal would force us to remove all the subgoals from the group.

Algorithm for computing the minimal Z-partition

There is a simple one pass algorithm that computes the minimal Z-partition. Scan the given list of subgoals and consider each subgoal in turn. If the subgoal has no nondistinguished variable, assign it to a new group. If the subgoal has some nondistinguished variable, look for an existing group that shares some nondistinguished variable with the subgoal. If none can be found, assign the subgoal to a new group. Otherwise, merge all such groups and assign the subgoal to the result.

4 Complete Tests for SM

This section presents solutions for finding CTSM for CQ views defined by (1). We only consider insertions of a single tuple into a base relation. We first present our result for the special case of (1) where $\bar{X}' = \bar{X} = \emptyset$, $\bar{U}' = \bar{U}$ and $\bar{Z}' = \bar{Z}$. This special case is important only because the technique used is applicable to the general case. We then present results for the general case.

4.1 Important special case

Consider the following view definition:

$$Q : v(\bar{U}, \bar{Z}) \coloneq r(\bar{U}) \ \& \ S(\bar{U}, \bar{Z}) \tag{2}$$

Let the minimal Z-partition of $S(\bar{U}, \bar{Z})$ consist of groups $g_1, \ldots, g_n$. Let $\bar{U}_i$ (resp. $\bar{Z}_i$) denote the set of distinguished (resp. nondistinguished) variables used in group g_i. A necessary and sufficient condition for self-maintainability is given in the following theorem.

Theorem 4.1 *A CTSM for inserting $r(\bar{a})$ is given by the following condition:*

$$\bigwedge_{i=1}^{n} (\exists \bar{U} \bar{Z})[(\bar{U}, \bar{Z}) \in V \wedge \bar{U}_i = \bar{a}_i]^3 \tag{3}$$

To maintain view V (when the view is self-maintainable), insert tuples $(\bar{a}, \bar{z})$ for all $\bar{z}$ in the cross-product $\bar{z}_1 \times \ldots \times \bar{z}_n$ where $\bar{z}_i$ is obtained from the query

$$\{\bar{Z}_i \mid (\bar{U}, \bar{Z}) \in V \wedge \bar{U}_i = \bar{a}_i\} \tag{4}$$

∎

Proof: The full proof is given in the Appendix. ∎

Example 4.1 Consider the view definition

$$v(U, V, W, Z, T) \coloneq r(U, V, W) \ \& \ s_1(U, V) \ \& \ s_2(V, Z) \ \& \ s_3(W, Z) \ \& \ s_4(T).$$

The minimal Z-partition consists of three groups: group $\{s_1(U, V)\}$ using distinguished variables UV, group $\{s_2(V, Z), s_3(W, Z)\}$ using distinguished variables VW and group $\{s_4(T)\}$ using no distinguished variables. A complete test of self-maintainability for inserting $r(a, b, c)$ is given by

$$(a, b, -, -, -) \in V \wedge (-, b, c, -, -) \in V \wedge (-, -, -, -, -) \in V$$

Since the last conjunct (meaning that V is nonempty) is subsumed by the other conjuncts, the condition can be simplified to:

$$(a, b, -, -, -) \in V \wedge (-, b, c, -, -) \in V$$

To maintain V, add all tuples (a, b, c, z, t) such that $(-, b, c, z, -) \in V$ and $(-, -, -, -, t) \in V$. ∎

Simplification

The complete test (3) for SM can often be simplified by eliminating any conjunct that is subsumed by another conjunct: when two groups g_i and g_j are such that $\bar{U}_i \subseteq \bar{U}_j$, the conjunct that corresponds to g_i can be eliminated without affecting the logical meaning of the test.

[3] Notation: $\bar{a}_i$ denotes the restriction of $\bar{a}$ over the $\bar{U}_i$ components.

4.2 General case where all join variables are exposed

This case directly generalizes the special case (2), in which the X-variables are introduced to r and not all X-variables and Z-variables are exposed. That is, consider the view definition:

$$Q : v(\bar{X}', \bar{U}, \bar{Z}') \coloneq r(\bar{X}, \bar{U}) \;\&\; S(\bar{U}, \bar{Z}) \tag{5}$$

Theorem 4.2 *A CTSM for inserting $r(\bar{b}, \bar{a})$ is given by the following condition:*

$$\bigwedge_{i=1}^{n} (\exists \bar{X}'\bar{U}\bar{Z}')[(\bar{X}', \bar{U}, \bar{Z}') \in V \wedge \bar{U}_i = \bar{a}_i] \tag{6}$$

To maintain view V (when the view is self-maintainable), insert tuples $(\bar{b}', \bar{a}, \bar{z}')$ for all $\bar{z}'$ in the cross-product $\bar{z}'_1 \times \ldots \times \bar{z}'_n$ where $\bar{z}'_i$ is obtained from the query

$$\{\bar{Z}'_i \mid (\bar{X}', \bar{U}, \bar{Z}') \in V \wedge \bar{U}_i = \bar{a}_i\}^4 \tag{7}$$

∎

Proof: The proof is not included due to space limitation. We have a proof very similar to the one for the special case of Section 4.1, where the database instance D_1 in the counterexample is constructed from V by padding the hidden variables with new constants, for each tuple from V. ∎

Example 4.2 Consider the view definition

$$v(U, V, W, Z) \coloneq r(U, V, W, X) \;\&\; s_1(U, V) \;\&\; s_2(V, Z) \;\&\; s_3(W, Z) \;\&\; s_4(T).$$

A CTSM for inserting $r(a, b, c, d)$ is given by

$$(a, b, -, -) \in V \wedge (-, b, c, -) \in V$$

To maintain V, add all tuples (a, b, c, z) such that $(-, b, c, z) \in V$. ∎

4.3 General case where some join variables are hidden

When some of the join variables (i.e. U-variables) are hidden, the view becomes "less self-maintainable" in some sense. Intuitively, any CTSM is expected to be stricter than when all join variables are exposed. Consider the view definition:

$$Q : v(\bar{X}', \bar{U}', \bar{Z}') \coloneq r(\bar{X}, \bar{U}) \;\&\; S(\bar{U}, \bar{Z}) \tag{8}$$

where $\bar{U}'$ is a proper subset of $\bar{U}$. There are two subcases we need to consider: the case where every group g_i either has no exposed $\bar{Z}_i$ or has no hidden $\bar{U}_i$, and the opposite case.

4.3.1 For all i, $\bar{Z}'_i$ empty or $\bar{U}_i \subseteq \bar{U}'$

Theorem 4.3 *A CTSM for inserting $r(\bar{b}, \bar{a})$ is given by the following condition:*

$$(\exists \bar{X}'\bar{U}'\bar{Z}')[(\bar{X}', \bar{U}', \bar{Z}') \in V \wedge \bar{X}' = \bar{b}' \wedge \bar{U}' = \bar{a}'] \tag{9}$$

To maintain view V (when the view is self-maintainable), no tuples need to be inserted into V. ∎

Proof: The proof is not included due to space limitation. We have a proof that uses the same technique as for the other cases. The sufficiency proof involves showing that $Q(D^{\mu}) = Q(D) = V$. In the counterexample used, $Q(D_1^{\mu}) = Q(D_1)$. ∎

[4]Notation: $\bar{Z}'_i$ denotes $\bar{Z}' \cap \bar{Z}_i$, that is, the exposed variables in $\bar{Z}_i$.

4.3.2 For some i, $\bar{Z}_i'$ nonempty and $\bar{U}_i \not\subseteq \bar{U}'$

This is the worst case in the sense that the view is totally not self-maintainable, as stated in the following theorem.

Theorem 4.4 *For insertion of $r(\bar{b}, \bar{a})$, the view is not SM.* ∎

Example 4.3 A view defined by

$$v(X, V, W, Z) :- r(U, V, W, X) \ \& \ s_1(U, V) \ \& \ s_2(V, Z) \ \& \ s_3(W, Z) \ \& \ s_4(T)$$

may be self-maintainable when tuples are added to relation r. But, when the join-variable V is dropped from the head, the view is no longer self-maintainable. ∎

5 Conclusion

The table in Figure 4 summarizes the main results of this paper. Interestingly, when all join-variables are exposed in the view, the CTSM does not depend on $\bar{b}$, the X-components of the inserted tuple. When enough of the join-variables are hidden, the view ceases to be able to self-maintain.

Characterization of view definition	SM complete test for inserting $r(\bar{b}, \bar{a})$
$\bar{U}' = \bar{U}$	$\bigwedge_{i=1}^{n}(\exists \bar{X}'\bar{U}\bar{Z}')[(\bar{X}', \bar{U}, \bar{Z}') \in V \wedge \bar{U}_i = \bar{a}_i]$
$\bar{U}' \subset \bar{U}$, $\bigwedge_{i=1}^{n}(\bar{Z}_i' = \emptyset \vee \bar{U}_i \subseteq \bar{U}')$	$(\exists \bar{X}'\bar{U}'\bar{Z}')[(\bar{X}', \bar{U}', \bar{Z}') \in V \wedge \bar{X}' = \bar{b}' \wedge \bar{U}' = \bar{a}']$
$\bar{U}' \subset \bar{U}$, $\bigvee_{i=1}^{n}(\bar{Z}_i' \neq \emptyset \wedge \bar{U}_i \not\subseteq \bar{U}')$	$FALSE$

Figure 4: Self-Maintainability Tests of CQ Views

The results also demonstrate that testing for self-maintainability not only can be practically implemented, but can also be efficiently implemented: the CTSM's that are generated at view-definition-time can be optimized and suggest ways to index the materialized view that can be exploited to speed up update-time testing and maintenance works.

We briefly mentioned that multivalued dependencies are an alternative technique to characterize view definitions, which lends itself easily to analyses involving dependencies on base relations.

Work is under way to find CTSM's for CQ views that allow self-joins in their definition. Consider for instance the following view definition

$$v(X, Y, Z) :- r(X, Y) \ \& \ t(X, Z) \ \& \ t(Y, Z).$$

A CTSM for inserting $r(a, b)$ is

$$(a, b, -) \in V \vee (b, a, -) \in V \vee [(a, a, -) \in V \wedge (b, b, -) \in V \wedge (No\ c)[(a, a, c) \in V \wedge (b, b, c) \in V]]$$

Thus the presence of self-joins introduces extra complexity in the CTSM, since components of the $S(\bar{U}, \bar{Z})$ may "commute" among themselves. We are currently investigating the use of generalized dependencies to capture this added constraint on S.

In future work, we plan to extend our techniques to analyzing views whose definition involves use of negation. Similar techniques have already been successfully used in our work on finding complete tests for constraint maintenance under limited data access ([Huyn96]), a different problem but related to the view maintenance problem.

References

[GB95] Gupta A. and Blakeley J. A.: Using Partial Information to Update Materialized Views. *To appear in Information Systems*, 1996?.

[Huyn96] Huyn N.: Testing CQC$^{\neg}$ constraints under limited data access. *Unpublished Technical Report*, available as `URL http:// www-db.stanford.edu/pub/papers/cqcnclt-tr.ps`, 1996.

[TB88] Tompa F. W. and Blakeley J. A.: Maintaining Materialized Views Without Accessing Base Data. *Information Systems,* 13(4), pp. 393–406, 1988.

A Appendix

A.1 Proof of Theorem 4.1

View definition (2) can be rewritten as:

$$v(\bar{U}, \bar{Z}) :\!-\, r(\bar{U}) \;\&\; \bigwedge_{i=1}^{n} S_i(\bar{U}_i, \bar{Z}_i)$$

where each S_i represents the conjunction of subgoals from group g_i as defined by the minimal Z-partition of $S(\bar{U}, \bar{Z})$. Another way to look at the abstract structure of $S(\bar{U}, \bar{Z})$ is to use multivalued dependencies, as depicted in Figure 5.

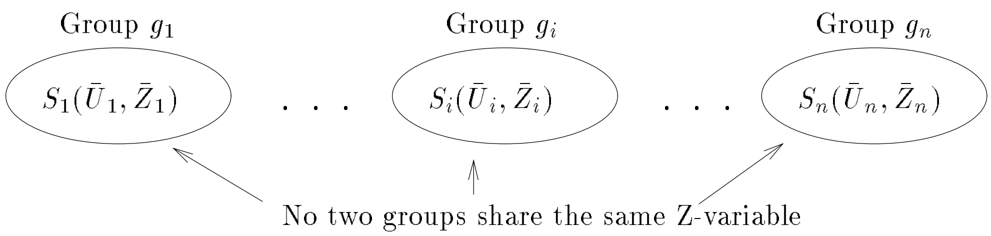


Figure 5: In the minimal Z-partition, $S(\bar{U}, \bar{Z})$ satisfies MVD $\bar{U} \twoheadrightarrow \bar{Z}_i$ for all i.

Sufficiency

To show that condition (3) is sufficient for SM, assume it is satisfied. Let D be a database instance consistent with V. We need to show that $Q(D^{\mu})$ does not depend on D.

$$\begin{aligned} Q(D^{\mu}) &= Q(D \cup \{r(\bar{a})\}) = Q(D) \cup \{(\bar{a}, \bar{z}) \mid S(\bar{a}, \bar{z}) \in D\} = V \cup \{(\bar{a}, \bar{z}) \mid \bigwedge_{i=1}^{n} S_i(\bar{a}_i, \bar{z}_i) \in D\} \\ &= V \cup \{(\bar{a})\} \times \{(\bar{z}_1) \mid S_1(\bar{a}_1, \bar{z}_1) \in D\} \times \ldots \times \{(\bar{z}_n) \mid S_n(\bar{a}_n, \bar{z}_n) \in D\} \end{aligned}$$

For any i, there is a tuple $(\bar{u}, \bar{z})$ in V such that $\bar{u}_i = \bar{a}_i$ (i.e. $\bar{u}$ and $\bar{a}$ agree over $\bar{U}_i$) and D contains $r(\bar{u})$, $S_1(\bar{u}_1, \bar{z}_1)$, $\ldots$, $S_n(\bar{u}_n, \bar{z}_n)$. Thus any $S_i(\bar{a}_i, \bar{z}'_i)$ would join with these tuples to generate

$v(\bar{u}, \bar{z}')$ where $\bar{z}'$ is obtained by replacing the $\bar{Z}_i$ components of $\bar{z}$ with $\bar{z}'_i$. Conversely, any tuple $(\bar{u}, \bar{z})$ in V such that $\bar{u}_i = \bar{a}_i$ implies the existence of some $S_i(\bar{a}_i, \bar{z}'_i)$ where $\bar{z}'_i = \bar{z}_i$. Therefore $\{\bar{Z}_i \mid (\bar{a}_i, \bar{Z}_i) \in S_i\} = \{\bar{Z}_i \mid (\bar{U}, \bar{Z}) \in V \wedge \bar{U}_i = \bar{a}_i\}$. Now we can rewrite $Q(D^\mu)$ as:

$$Q(D^\mu) = V \cup \{(\bar{a})\} \times \bigotimes_{i=1}^{n} \{(\bar{z}_i) \mid (\bar{u}, \bar{z}) \in V \wedge \bar{u}_i = \bar{a}_i\}^5$$

Therefore, not only we showed $Q(D^\mu)$ is independent of D, but we also derived the view maintainance expression (4).

Necessity

To show that condition (3) is necessary for SM, assume it is not satisfied. We need to construct two database instances D_1 and D_2 that are both consistent with V but such that $Q(D_1^\mu) \neq Q(D_2^\mu)$.

For D_1, we use the "canonical" database instance consistent with V, constructed the following way: each tuple in V binds the variables $\bar{U}$ and $\bar{Z}$ in the head of (2); substituting these bindings into the body makes each subgoal into an atom, a ground atom in this case. The canonical instance consists of such ground atoms generated by all tuples in V.

To construct D_2, we add to D_1 a set Δ of new tuples (i.e., that is not already in D_1) as follows. Condition (3) can be written as $\bigwedge_{i=1}^{n} cond_i$. Since the condition is not satisfied, there is some $cond_i$ that is false. For each i such that $cond_i$ is false, new tuples are included into Δ according to which of the following categories group g_i belongs:

A If the group has no nondistinguished variable (i.e. $\bar{Z}_i = \emptyset$), it consists of a single subgoal, say $p(\bar{U}_i)$. It is not difficult to see that by construction of the canonical instance, D_1 could not possibly contain $p(\bar{a}_i)$. Therefore we include $p(\bar{a}_i)$ in Δ.

B If the group has some nondistinguished variable (i.e. $\bar{Z}_i \neq \emptyset$), we bind all nondistinguished variables in the group to new constants (say bind $\bar{Z}_i$ to $\bar{z}_i^{new}$). $S_i(\bar{a}_i, \bar{z}_i^{new})$ is a set of ground atoms each of which contains some new constant and thus cannot be in D_1. We therefore include $S_i(\bar{a}_i, \bar{z}_i^{new})$ in Δ.

This construction of Δ is illustrated in Figure 6.

	$g_1 \ldots$	g_i $cond_i$ false, Cat. A	g_i $cond_i$ false, Cat. B	g_i $cond_i$ true	$\ldots g_n$
D_1		$S_i(\bar{a}_i)$ absent		Some $S_i(\bar{a}_i, \bar{z}_i)$ already present	
Δ		Add $S_i(\bar{a}_i)$	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$	No tuples added	

Figure 6: Construction of the counterexample

Now that we have specified D_2, we need to verify that it is indeed consistent with V. Since $D_1 \subseteq D_2$ and Q is monotonic, we only need to make sure that Q cannot generate any new tuple when Δ is added to D_1. Any new tuple Q generates must use some tuple $t \in \Delta$ which falls into either Category A or Category B:

[5] $\bigotimes_{i=1}^{n} R_i$ denotes the cross-product $R_1 \times R_2 \times \ldots \times R_n$.

- For Category A, t includes $\bar{a}_i$ as components and since $cond_i$ is false, relation r in D_1 (or D_2) has no tuple that agrees with $\bar{a}$ over $\bar{U}_i$. Therefore, t cannot join with any tuple from r, and using t, Q cannot generate any new tuple.
- For Category B, using t from $S_i(\bar{a}_i, \bar{z}_i^{new})$ forces us to use all tuples from $S_i(\bar{a}_i, \bar{z}_i^{new})$. S_i generates exactly the tuple $(\bar{a}_i, \bar{z}_i^{new})$ which cannot join with any tuple from r since $cond_i$ is false. So again, Q cannot generate any new tuple if t is used.

Finally, to verify that $Q(D_1^\mu) \neq Q(D_2^\mu)$, we need to find a tuple in $Q(D_2^\mu)$ that is not in $Q(D_1^\mu)$. Consider the tuple t' that joins the following facts from $Q(D_2^\mu)$:

- $r(\bar{a})$,
- All the new facts from Δ (there is at least one such new fact),
- For each group g_i such that $cond_i$ is satisfied, we know that $(\exists Z_i)S_i(\bar{a}_i, \bar{Z}_i)$ is satisfied in the canonical instance D_1. We arbitrarily choose some value $\bar{z}_i$ that satisfies $S_i(\bar{a}_i, \bar{z}_i)$. So we use all the facts in $S_i(\bar{a}_i, \bar{z}_i)$. These facts are old since they are all in D_1.

Tuple t' could not possibly be in $Q(D_1^\mu)$ since it is derived from at least a new fact from Δ:

- If the new fact falls into Category A (say $p(\bar{a}_i)$), the only way t' can be in $Q(D_1^\mu)$ is that $p(\bar{a}_i) \in D_1$, which we already know is not possible.
- If the new fact falls into Category B, one of its components must be a new constant. So t' must contain some new constant and thus cannot be in $Q(D_1^\mu)$.

A.2 Proof of Theorem 4.3

Sufficiency

To show that condition (9) is sufficient for SM, assume it is satisfied. Let D be a database instance consistent with V. We need to show that $Q(D^\mu)$ does not depend on D.

$$Q(D^\mu) = Q(D \cup \{r(\bar{b}, \bar{a})\}) = Q(D) \cup \{(\bar{b}', \bar{a}', \bar{z}') \mid S(\bar{a}, \bar{z}) \in D\} = V \cup \{(\bar{b}', \bar{a}')\} \times \{(\bar{z}') \mid \bigwedge_{i=1}^{n} S_i(\bar{a}_i, \bar{z}_i) \in D\}$$

Figure 7 summarizes what can be inferred about S_i from condition (9), depending on which category group g_i belongs.

Since there is no group in Category B_1 (using the hypothesis that $\bar{Z}_i'$ empty or $\bar{U}_i \subseteq \bar{U}'$ for all i), the only groups that can contribute any value to the exposed Z-variables belong to Category B_2 and Category AB. $Q(D^\mu)$ can be rewritten as:

$$V \cup \{(\bar{b}', \bar{a}')\} \times \bigotimes_{Cat.B_2} \{(\bar{z}_i') \mid S_i(\bar{a}_i, \bar{z}_i) \in D\} \times \bigotimes_{Cat.AB} \{(\bar{z}_i') \mid S_i(\bar{z}_i) \in D\} \times \{() \mid \bigwedge_{OtherCat.} S_i(\bar{a}_i, \bar{z}_i) \in D\}$$

The last factor in the cross-product, denoted below as $cond_D$, represents the effect the remaining groups have on $Q(D^\mu)$. While it does not contribute any value to the exposed Z-variables, it is boolean condition that can still affects the result for the entire cross-product. Even though the boolean condition may or may not be satisfied, depending on the actual instance D, it should not matter since we will show that the entire cross-product is in V and hence that $Q(D^\mu) = V$ regardless of D.

In fact, since condition (3) holds by hypothesis, the following holds for any group g_i in either Category B_2 or AB:

$$\{(\bar{z}_i') \mid S_i(\bar{a}_i, \bar{z}_i) \in D\} = \{(\bar{z}_i') \mid (\bar{b}', \bar{a}', -\bar{z}_i'-) \in V\}$$

We can finally rewrite $Q(D^\mu)$ as:

$$V \cup \{(\bar{b}', \bar{a}')\} \times \{(\bar{z}') \mid (\bar{b}', \bar{a}', \bar{z}') \in V\} \times \{() \mid cond_D\}$$

Since the cross-product is clearly a subset of V, we conclude that $Q(D^\mu) = V$. Thus, $Q(D^\mu)$ is independent of D simply because *no update is needed* to bring it up to date.

Necessity

To show that condition (9) is necessary for SM, assume it is not satisfied. That is, V has no tuples of the form $(\bar{b}', \bar{a}', -)$. We need to construct two database instances D_1 and D_2 that are both consistent with V but such that D_1^μ and D_2^μ derive different views.

We can assume $V \neq \emptyset$, since otherwise a trivial counterexample can be constructed.

Let D_1 be the canonical database instance constructed from V the usual way. Since $\bar{U}'$ is a proper subset of $\bar{U}$, D_1 has no $r(-, \bar{a})$ and no $S(\bar{a}, -)$. As a consequence, the newly inserted $r(\bar{b}, \bar{a})$ cannot join with S, and therefore $Q(D_1^\mu) = Q(D_1)$.

To construct D_2, a set Δ of new tuples is added to D_1. Δ is specified in Figure 8, where the group categories are defined as in Figure 7. Note that Category B_1 is omitted since there are no groups in this category.

To show that D_2 is consistent with V, we show that no tuple in $Q(D_2)$ can be derived using some tuple $t \in \Delta$. The following considers all possible cases tuple t can be in:

- Category A_1. D_1 has no $r(-, -\bar{a}_i-)$ since $\bar{U}_i$ is not completely exposed. Therefore t cannot join with any tuple from r.
- Category A_2. D_1 has no $r(-, -\bar{a}_i-)$ since V has no $(-, -\bar{a}_i-, -)$. Therefore t cannot join with any tuple from r.
- Category B_2. D_1 has no $r(-, -\bar{a}_i-)$ since V has no $(-, -\bar{a}_i-, -)$. Using any tuple t from $S_i(\bar{a}_i, \bar{z}_i^{new})$ forces all tuples from $S_i(\bar{a}_i, \bar{z}_i^{new})$ to be used. So S_i generates exactly the tuple $(\bar{a}_i, \bar{z}_i^{new})$ which cannot join with any tuple from r.
- Category B_3. D_1 has no $r(-, -\bar{a}_i-)$ since $\bar{U}_i$ is not completely exposed. Using any tuple t from $S_i(\bar{a}_i, \bar{z}_i^{new})$ forces all tuples from $S_i(\bar{a}_i, \bar{z}_i^{new})$ to be used. So S_i generates exactly the tuple $(\bar{a}_i, \bar{z}_i^{new})$ which cannot join with any tuple from r.
- Category B_4: same arguments as for Category B_2.

Finally, to show that D_1^μ and D_2^μ derive different views, we will find a tuple $t' \in Q(D_2^\mu)$ that is not in V. Consider the tuple $t' = (\bar{b}', \bar{a}', \bar{z}')$ derived by joining the following facts from $Q(D_2^\mu)$:

- $r(\bar{b}, \bar{a})$,
- All the new facts from Δ,
- For each group g_i that contributes no tuples to Δ, we know that $(\exists Z_i) S_i(\bar{a}_i, \bar{Z}_i)$ is satisfied in the canonical instance D_1. We arbitrarily choose some value $\bar{z}_i$ that satisfies $S_i(\bar{a}_i, \bar{z}_i)$. So we use all the facts in $S_i(\bar{a}_i, \bar{z}_i)$. These facts are not new since they are all in D_1.

Now, t' cannot possibly be in V since by hypothesis, V has no tuples of the form $(\bar{b}', \bar{a}', -)$.

A.3 Proof of Theorem 4.4

We need to show we can always find a counterexample such that D_1 and D_2 are both consistent with V but D_1^μ and D_2^μ derive different views.

We can assume $V \neq \emptyset$, since otherwise a trivial counterexample can be constructed.

Let D_1 be the canonical database instance constructed from V the usual way. Since $\bar{U}'$ is a proper subset of $\bar{U}$, D_1 has no $r(-, \bar{a})$ and no $S(\bar{a}, -)$. As a consequence, the newly inserted $r(\bar{b}, \bar{a})$ cannot join with S, and therefore $Q(D_1^\mu) = Q(D_1)$.

To construct D_2, a set Δ of new tuples is added to D_1. Δ is specified in Figure 9.

To show that D_2 is consistent with V, we show that no tuple in $Q(D_2)$ can be derived using some tuple $t \in \Delta$. The following considers all possible cases tuple t can be in:

- Category A$_1$. D_1 has no $r(-, -\bar{a}_i-)$ since $\bar{U}_i$ is not completely exposed. Therefore t cannot join with any tuple from r.
- Category A$_2$. D_1 has no $r(-, -\bar{a}_i-)$ since V has no $(-, -\bar{a}_i-, -)$. Therefore t cannot join with any tuple from r.
- Category B$_1$. D_1 has no $r(-, -\bar{a}_i-)$ since $\bar{U}_i$ is not completely exposed. Using any tuple t from $S_i(\bar{a}_i, \bar{z}_i^{new})$ forces all tuples from $S_i(\bar{a}_i, \bar{z}_i^{new})$ to be used. So S_i generates exactly the tuple $(\bar{a}_i, \bar{z}_i^{new})$ which cannot join with any tuple from r.
- Category B$_2$. D_1 has no $r(-, -\bar{a}_i-)$ since V has no $(-, -\bar{a}_i-, -)$. Using any tuple t from $S_i(\bar{a}_i, \bar{z}_i^{new})$ forces all tuples from $S_i(\bar{a}_i, \bar{z}_i^{new})$ to be used. So S_i generates exactly the tuple $(\bar{a}_i, \bar{z}_i^{new})$ which cannot join with any tuple from r.
- Category B$_3$: same arguments as for Category B$_1$.
- Category B$_4$: same arguments as for Category B$_2$.

Finally, to show that D_1^μ and D_2^μ derive different views, we will find a tuple $t' \in Q(D_2^\mu)$ that is not in V. Consider the tuple $t' = (\bar{b}', \bar{a}', \bar{z}')$ derived by joining the following facts from $Q(D_2^\mu)$:

- $r(\bar{b}, \bar{a})$,
- All the new facts from Δ (there is a least one such fact, since there is at least a group from Category B$_1$),
- For each group g_i that contributes no tuples to Δ, we know that $(\exists Z_i) S_i(\bar{a}_i, \bar{Z}_i)$ is satisfied in the canonical instance D_1. We arbitrarily choose some value $\bar{z}_i$ that satisfies $S_i(\bar{a}_i, \bar{z}_i)$. So we use all the facts in $S_i(\bar{a}_i, \bar{z}_i)$. These facts are not new since they are all in D_1.

Now, t' cannot possibly be in V since it is derived from some fact in Category B_1, and hence must have components with new values (recall that a group in Category B_1 has some Z-variables exposed).

Category of group g_i	Syntactic characterization	$S_i(\bar{a}_i, -) \in D$?
A$_1$	$\bar{Z}_i = \emptyset, \bar{U}_i \neq \emptyset, \bar{U}_i \not\subseteq \bar{U}'$	?
A$_2$	$\bar{Z}_i = \emptyset, \bar{U}_i \neq \emptyset, \bar{U}_i \subseteq \bar{U}'$	Yes
AB	$\bar{U}_i = \emptyset$	Yes
B$_1$	$\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset,\ \bar{Z}'_i \neq \emptyset,\ \bar{U}_i \not\subseteq \bar{U}'$	?
B$_2$	$\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset,\ \bar{Z}'_i \neq \emptyset,\ \bar{U}_i \subseteq \bar{U}'$	Yes
B$_3$	$\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset,\ \bar{Z}'_i = \emptyset,\ \bar{U}_i \not\subseteq \bar{U}'$	?
B$_4$	$\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset\ \bar{Z}'_i = \emptyset,\ \bar{U}_i \subseteq \bar{U}'$	Yes

Figure 7: Group categories

	Cat. A$_1$	Cat. A$_2$	Cat. AB
D_1	$S_i(\bar{a}_i)$ absent		Some $S_i(-)$ already present
Δ	Add $S_i(\bar{a}_i)$	Add $S_i(\bar{a}_i)$ if no $V(-, -\bar{a}_i-, -)$	No tuples added

	Cat B$_2$	Cat B$_3$	Cat B$_4$
D_1			
Δ	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$ if no $V(-, -\bar{a}_i-, -)$	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$ if no $V(-, -\bar{a}_i-, -)$

Figure 8: Construction of the counterexample

	Cat. A$_1$: $\bar{Z}_i = \emptyset, \bar{U}_i \neq \emptyset$, $\bar{U}_i \not\subseteq \bar{U}'$	Cat. A$_2$: $\bar{Z}_i = \emptyset, \bar{U}_i \neq \emptyset$, $\bar{U}_i \subseteq \bar{U}'$	Cat. AB: $\bar{U}_i = \emptyset$
D_1	$S_i(\bar{a}_i)$ absent		Some $S_i(-)$ already present
Δ	Add $S_i(\bar{a}_i)$	Add $S_i(\bar{a}_i)$ if no $V(-, -\bar{a}_i-, -)$	No tuples added

	Cat B$_1$: $\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset$ $\bar{Z}'_i \neq \emptyset,\ \bar{U}_i \not\subseteq \bar{U}'$	Cat B$_2$: $\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset$ $\bar{Z}'_i \neq \emptyset,\ \bar{U}_i \subseteq \bar{U}'$	Cat B$_3$: $\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset$ $\bar{Z}'_i = \emptyset,\ \bar{U}_i \not\subseteq \bar{U}'$	Cat B$_4$: $\bar{Z}_i \neq \emptyset, \bar{U}_i \neq \emptyset$ $\bar{Z}'_i = \emptyset,\ \bar{U}_i \subseteq \bar{U}'$
D_1				
Δ	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$ if no $V(-, -\bar{a}_i-, -)$	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$	Add $S_i(\bar{a}_i, \bar{z}_i^{new})$ if no $V(-, -\bar{a}_i-, -)$

Figure 9: Construction of the counterexample