# H2-2000-568

**File:** H2-2000-568.doc

2000-11-20 — 20 November, 2000

**Printed:** Monday, 20 November, 2000 at 10:52

Authoritative Version: Adobe Acrobat Portable Document Format (PDF)

# ISO

**International Organization for Standardization**


# ANSI

**American National Standards Institute**

**ANSI TC X3H2**

**Database**

**ISO/IEC JTC 1/SC 32**

**Data Management and Interchange**

**WG 3**

**Database Languages**

**Project:** ANSI: 1234D — ISO: 1.32.3.4

**Title:** SQL/Replication Scope and Requirements Document

**Status:** Informational

**Author:** Herb Sutter (USA)

**Abstract:** Scope and requirements document for Database Language SQL, Part 12: SQL/Replication.

**References:**

1. [Frame99] ISO/IEC 9075-2:1999, International Standard, Database Language SQL, Part 1 — *SQL/Framework*, by Jim Melton (ed.), September 1999.

2. [Found99] ISO/IEC 9075-2:1999, International Standard, Database Language SQL, Part 2 — *SQL/Foundation*, by Jim Melton (ed.), September 1999.

3. [CLI99] ISO/IEC 9075-3:1999, International Standard, Database Language SQL, Part 3 — *SQL/CLI*, by Jim Melton (ed.), September 1999.

4. [PSM99] ISO/IEC 9075-4:1999, International Standard, Database Language SQL, Part 4 — *SQL/PSM*, by Jim Melton (ed.), September 1999.

5. [Bindings99] ISO/IEC 9075-5:1999, International Standard, Database Language SQL, Part 5 — *SQL/Bindings*, by Jim Melton (ed.), September 1999.

6. [99-474] H2-99-474, *Proposal to Develop an NCITS Standard*, by Herb Sutter, November 1999.

7. [99-475] H2-99-475 = WG3:SAF-045, *ISO Project Proposal for Replication*, by Herb Sutter, November 1999.

# Table of Contents

# 1. Purpose and Scope

This document describes the scope and high-level requirements proposed for Phase 1 of ANSI SQL Part 12: SQL/Replication.

## *1.1. Discussion*

A replication standard could follow one of two major approaches.

The first approach enables a user to take databases from different vendors that conform to Part 12, use standard syntax to set up and administer replication, and replicate the databases directly using each database's built-in interoperable replication facilities. Each vendor's database is responsible only for its end of the replication. Multiple replication implementations are used, and are interoperable. Such an approach may be taken at some future time, but has for now been deferred at least until Phase 2 of this standard because of its larger scope.

This paper contemplates the second approach, which is to facilitate the development of heterogeneous replication systems available to users. A user with two databases from two different vendors can replicate between them using a single replication implementation — one of those supplied by the database vendors, or one supplied by a third party. Each vendor's database is responsible only for providing conforming facilities for identifying and applying changes efficiently for the purpose of replication. The selected replication system controls all replication for all databases, and supplies whatever facilities and features it desires (such as facilities for subset definition and management, or for replication scheduling).

To illustrate, consider a database vendor A with a replication product who wants to support heterogeneous replication to and from a database from Vendor B. Vendor A must be able to perform two major operations:

a) Change Apply. A set of changes C has been shipped to the destination database D, and must now be applied so that the changes in C are reflected in D. Without specifying the format or contents of C, which are beyond the scope of Phase 1, C can be thought of as containing some number of inserted, updated, and/or deleted rows. The basic changes can be applied using existing SQL insert, update, and delete operations.

   One example of what is desirable in addition to existing SQL is a facility to suppress normal triggers for changes made during replication, because trigger effects that have already taken place on the originating database should typically not be applied again on every destination database.

b) Change Capture. A source database S must be replicated, and changes must be identified so that they can be marshaled and distributed in whatever fashion is desired by the replication process. However, Vendor A has a more challenging time gathering these changes, and most existing implementations use a variant of one of three major alternative strategies:

   i) Use the standard SQL facilities. Facilities like triggers do help with change capture but weren't designed/optimized specifically for replication. Methods in this family are relatively portable, but typically do have some drawbacks such as slowing down user transactions.

   ii) Write custom code to read proprietary logs or change notifications. Custom code can be written to read Vendor B's proprietary transaction log or change notification API (if one exists, or directly read a reverse-engineered log format if no API is published). This can require considerable effort, and the resulting code is specialized for Vendor B's log and so isn't directly reusable for Vendor C's database. This kind of code also tends to be fragile, and has to be maintained as Vendor B's log API changes across releases.

   iii) Require custom middleware. Some products are provided as libraries or toolkits, or require the application using the replicated database to perform its changes through a proprietary SQL/CLI driver or similar module. Changes are captured as the middleware observes them being made by the application. This approach is generally suitable only for new application development, and even then often interferes with the integration of third-party software tools that are not aware of the change

capture module and cannot be rewritten to interface with it. It typically cannot detect changes made by database administrators who execute SQL statements and procedures, or by Java, HTML, or other tools that may not be able to use the middleware.

iv) Duplicate the database. Some early third-party products detected changes in a replicated table by storing an exact copy of the table in a second location and periodically scanning to detect differences. This approach was highly portable across SQL implementations, but was prohibitively space- and time-intensive for any but very small databases.

Instead, Phase 1 of the standard will narrowly focus on providing a standard facility to identify (capture) changes to replicated data, allowing all replication products to avoid having to use triggers or read proprietary facilities. The facility as contemplated may take the form of a data stream format, or an API, or some other format, or one or more of the above; these details are to be decided during standardization.

This restricted scope will enable the standards architects to reduce the initial extensions to SQL and to produce a standard faster, and enable the vendors to produce conforming implementations sooner.

## *1.2. Included Scope*

The goal of Phase 1 is to specify facilities to simplify the implementation of heterogeneous database replication. The primary emphasis will be on facilities to efficiently implement change capture and change apply more portably across SQL implementations.

### 1.2.1. Guiding Principles

The following are intended to be fundamental guiding principles that influence all features and other details of the proposed work.

a) Preservation of Homogeneous Implementation Latitude. Many vendors have existing replication products that are optimized to replicate their own native database formats efficiently using proprietary extensions and tools. The features in this standard shall not preclude such efficiencies and tuning for a vendor's own database, even when that database is being replicated heterogeneously with a database supplied by another vendor by making use of the facilities described herein.

b) Compatibility With Current Implementation Models. Existing replication products implement a variety of replication models and architectures, some similar and some dissimilar. The proposed standard should, to the extent possible, provide sufficient information to be able to act as a "front end" that a product can use and map into its own change capture representations.

c) Asynchronous Replication Only. Specifically, this means replication that is not performed within the context of the original user transaction, whereas in synchronous replication (e.g., 2-Phase Commit) the original user transaction does not fully commit until the change is also committed at some or all other participating databases.

d) SQL To SQL. This standard is concerned with replication from and to SQL implementations only.

e) Vendor-Neutral. The facilities described in this standard should be sufficient for a third party vendor to implement a replication system without knowledge of the source and destination SQL implementations. Conversely, a SQL implementation that conforms to this standard by providing the specified facilities should be able to operate with any replicator designed to capture and apply changes using the facilities specified in this standard.

f) Phase 1 Architectural Scope Limited To Change Capture & Change Apply. See Figure 1. The "replication agent" virtual component is not further defined in the scope of this standard and may be implemented as a single component, multiple distributed components, or in any other convenient fashion chosen by the implementor.

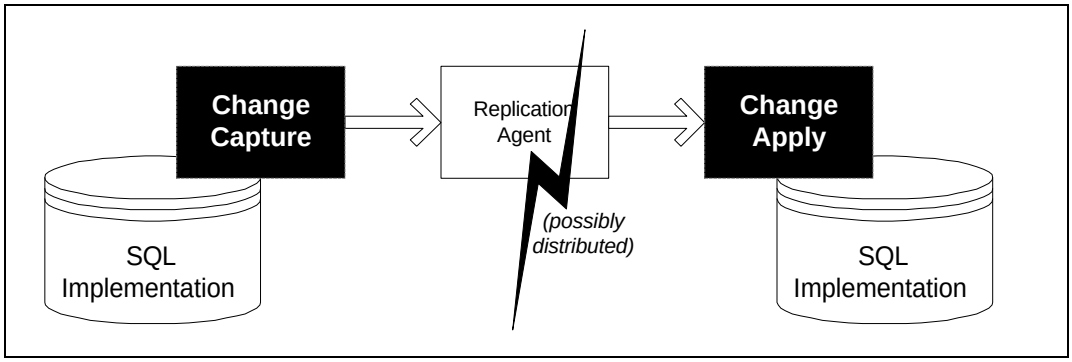


**Figure 1: Architectural scope of SQL/Replication limited to change capture and change apply**

### 1.2.2. Requirements and Features

In order to support the above intended scope, the standard shall include facilities that will include, but are not necessarily limited to, requirements and features such as the following:

a) Change Capture. The change capture facility should include the ability to efficiently identify changes to replicated data using standard facilities. This should preferably require the change capture implementation to maintain minimal state about the agents connected to it, in order to avoid the drawbacks of requiring the implementation to maintain state information about what operations succeeded and failed, and similar operational details. Replication agent quality of service issues, such as timely and robust change delivery to other databases, are unspecified and left to replication agents.

   For example, a possible specification might allow for the possibility that replication agents may need to requery previously-queried information (perhaps because of loss of the replication agent's buffers or for other reasons), and to provide a way for the agent to notify the change capture implementation when it may release information up to a specified window so that the implementation may better manage its internal storage. Such a strategy would require the implementation to know only which agents are using it, and rely on each agent to remember its own detailed state.

   The change information presented by the change capture facility may include net changes only (e.g., if the same row has changed multiple times, show only the final image), individual changes (e.g., if the same row has changed multiple times, show each change individually), and/or transaction commit points (e.g., "INSERT Cust 123, INSERT Cust 124, COMMIT, …" to represent that the two inserts were performed in a single user transaction). Whether the information presented will follow a single standardized mode (e.g., "provide all individual changes and commit points," and the replication agent ignores the information not relevant to it), or multiple modes to be negotiated by the replication agent (e.g., "all changes and commit points mode" vs. "net changes only and no commit points mode"), is to be determined.

b) Change Apply. The standard should augment existing DML facilities to provide desirable replication-oriented facilities. Possible examples of special facilities available to the replication agent when applying changes include the suppression of normal triggers, suspension of referential rules, and/or extending or overriding audit information.

c) Multiple Products. More than one replication product's change/apply processors should be able to acquire/apply data change information from/to the same database.

## *1.3. Excluded Scope*

The following areas are explicitly excluded and out of scope of the proposed work:

a) Synchronous Replication. Specifically, this means replication that is performed within the context of the original user transaction, such that the original user transaction does not fully commit until the change is

also committed at some or all other participating databases. N-phase commit and similar approaches are already well-understood and mature and do not require standardization. Further, they do not solve the problems intended to be addressed by this work.

b) Non-SQL Data. Flat-file and other non-SQL data formats are not addressed by this standard. Some vendors may choose to implement their own vendor-specific extensions to the Part 12 facilities to support non-SQL data.

c) DDL. Replication of DDL actions, such as CREATE TABLE, are not addressed by this standard.

d) Other Replication Facilities. See Figure 1: Phase 1 does not address network topology, scheduling, change transport and propagation, or other aspects of the replication agent using the change capture and change apply facilities provided in this standard.

e) Distributed queries. Facilities for allowing one client query to be performed across multiple databases are already provided by other standards, including SQL/MED.

f) DML Statement Level Replication. The replication of SQL DML statements is deferred to Phase 2. Phase 1 concentrates on replicating the effects of executed statements, namely the final images of the inserted, updated, and deleted rows.

***– End of paper –***