NISTIR 5283

Security Considerations for SQL-based Implementations of STEP

Lawrence E. Bassham
W. Timothy Polk

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Gaithersburg, MD 20899

NIST

NISTIR 5283

Security Considerations for SQL-based Implementations of STEP

Lawrence E. Bassham
W. Timothy Polk

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Gaithersburg, MD 20899

October 1993

Contents

1 Introduction

The Database Language SQL (SQL) is a widely used language for accessing and manipulating relational databases. As such, SQL can be of use in many different operational environments, with correspondingly different needs for security. One specific application of this standard is in Product Data Exchange using STEP[1] (PDES).[PDE93a]

This paper examines the security implications of the versions of the SQL standard as used to implement STEP. STEP does not imply any particular security policy, so a variety of security policies are examined. The paper has been written as a companion document to NIST's general SQL security document, *Security Issues in the Database Language SQL* [PB93], and references that document frequently.

1.1 Audience

This paper is intended for designers of systems and databases that use SQL in implementing STEP. With [PB93], this paper will help to evaluate the security controls of SQL with respect to different security policies. This evaluation should help determine the types of controls and mechanisms that will be required from the SQL processor(s), as well as other components of the system (such as operating system (OS) controls, network protocols, and identification and authentication mechanisms).

1.2 The Standards

SQL has been adopted by both national and international standards organizations. It is the nature of standards to specify some features as required and other features as prohibited. Additional features may also be included in implementations, provided they do not conflict with elements specified in the standard. This allows for product differentiation. Certain security-relevant features are required in an SQL-compliant database management system (DBMS). Other security features are not specified by SQL, but may appear in particular products. The exact functionality of an SQL-compliant DBMS varies according to which version of the SQL standard is selected and what unspecified features are included. In addition, many products are not fully compliant, so further variation is possible.

[1]Standard for the Exchange of Product Model Data, an emerging international standard. The goal is a complete, unambiguous, computer-readable definition of the physical and functional characteristics of a product throughout its life cycle.[NIP91]

The basic SQL definition is ANSI X3.135-1989, *Database Language - SQL with Integrity Enhancement* [ANS89a], and will be referred to as SQL'89. The functionality of SQL'89 includes schema definition, data manipulation, and transaction management. SQL'89 and ANSI X3.168-1989, *Database Language - Embedded SQL* [ANS89b], form the basis for the Federal Information Processing Standard (FIPS) 127-1 [FIP90].

ANSI X3.135-1992 [ANS92] describes an enhanced SQL, known as SQL'92. The enhancements include schema manipulation, dynamic creation and execution of SQL statements, and network environment features for connection and session management [CO92][DD92]. The FIPS 127-2 [FIP93] is based upon X3.135-1992.

Finally, a third version of SQL, referred to in this paper as SQL3, is currently under development in ANSI and is expected to be approved in 1994 or 1995. SQL3 enhancements will include the ability to define, create and manipulate user-defined data types in addition to tables.

ISO/IEC Draft International Standards 9579-1 [ISO90a] and 9579-2 [ISO90b] define the Remote Database Access (RDA) protocol. RDA provides a method for interconnecting database management systems. ISO/IEC 9579-1 describes the generic model; ISO/IEC 9579-2 presents the SQL specialization information.

STEP is defined by the multi-part international standard ISO 10303. ISO 10303 is in development within ISO/TC 184/SC4. STEP includes generic parts for data specification[PDE92] (EXPRESS Language), file structure[PDE93b], and conformance testing[PDE93c]. STEP also includes Integrated Resources (IRs), which describe data exchange specific applications. Application Protocols (APs) define data exchange for specific applications and are defined in terms of the IRs and generic parts. The initial release of STEP contained twelve parts; a number of additional parts are under development. The twelve parts include an Overview, the three generic parts, six IRs and two APs.

Note that PDES is not a standard. PDES stands for Product Data Exchange Using STEP. PDES "refers to the United States contributing effort to this standardization process"[FCF93].

1.3 Using This Document

This paper will be most useful to people who are familiar with SQL or relational database management systems. The reader is also assumed to have a basic knowledge of computer security or to have previously read [PB93]. Finally, the document also assumes some familiarity with STEP, although it is not required.

SQL is a general purpose tool that could be used to implement STEP in various ways. Section 2, *SQL/STEP Architecture*, presents a basic architecture for an SQL-based implementation of STEP. This section examines security considerations that are of increased importance for this architecture.

Section 3 identifies controls and mechanisms that can be used to address the concerns described in Section 2. These controls are organized according to an augmented Trusted Computer System Evaluation Criteria (TCSEC) model presented in [PB93]. This model reflects the Trusted Database Interpretation (TDI) [TDI91] and National Computer Security Center (NCSC) Technical Report 79-91, *Integrity In Automated Information Systems* [TR791]. The Interpreted Requirements presented in the TDI provide the basis for discussions of labels and audit in database. Additions to the model from TR 79-91 include the definitions of data integrity and systems integrity.

STEP itself does not imply any particular implementation form or security policy. Certain policies are more easily implemented with the SQL/STEP architecture than others. In Section 4, three broad classes of security policies are briefly described, accompanied by a comparison of particular requirements with the facilities of SQL. The policy classes are: a role based policy, a mandatory access control policy for confidentiality, and a mandatory access control policy for integrity.

1.4 Security Considerations

The basic security requirements are the preservation of confidentiality and integrity while maintaining availability. There are a number of specific threats within these categories that merit special consideration with respect to database security.

Inference and aggregation are usually considered threats to mandatory access control policies, but they are threats in any environment. There are also a number of DBMS specific security issues, such as referential integrity and polyinstantiation. Classic OS problems such as deadlock and transaction completion must also be considered.

The following definitions will be used in this document:

- inference: derivation of new information from known information. The inference problem is that of users deducing unauthorized information from the legitimate information they acquire.[Thu92]

- aggregation: The result of assembling or combining distinct units of data when handling sensitive information. Aggregation of data at one sensitivity level may result in the total data being designated at a higher sensitivity level.[Rob91]

- polyinstantiation - A database design technique utilizing simultaneous existence of multiple data objects with the same name, differentiated by their access class, to hide the existence of high data[2] from low users[3] .[Cam90] Polyinstantiation allows a relation to contain multiple rows with the same primary key; the multiple instances are distinguished by their security levels.[SFD92]

- referential integrity - foreign keys must reference existing primary keys.[Cam90] It is important to maintain the integrity between the referencing values (foreign key values) and the referenced key values (primary key values).[DJ92]

- entity integrity - A tuple in a relation cannot have a null value for any of the primary key attributes.[DJ92]

[2]Data that should only be available to users with a high security level.
[3]Users with a low security level

2 SQL/STEP Architecture

SQL is not a part of STEP, but is a tool that can be used in the implementation of STEP. This section begins with a brief description of SQL's functionality. Secondly, two models of an SQL/STEP implementation are presented. Finally, the security problems associated with each component of the models are highlighted.

2.1 SQL Functionality

SQL defines standard facilities for relational database management systems. It describes facilities to perform four specific functions:

- define the structure of the database and the type of database elements by creating schema definitions;
- retrieve data from a database with a standard query interface;
- modify the contents of a database by adding, modifying or deleting tables or components; and
- the ability to define and manage SQL transactions.

Each of these components has associated security threats. Schema definition is related to the problems of inference and aggregation. Data retrieval tasks must conform to confidentiality policies. Data modification must conform to integrity policy.

2.2 SQL/STEP Implementation

SQL is a general purpose tool that could be applied to STEP in many ways. To perform the security analysis, it is necessary to assume some architecture for an SQL/STEP implementation. Figure 1 depicts one model, which can be designed with any SQL standard. Figure 2 depicts a second model, which requires SQL'92 or SQL3 along with RDA.

The first model shows an application interfacing with an SQL processor, which interfaces with a physical database on a local system. This model can be implemented with any version of SQL.

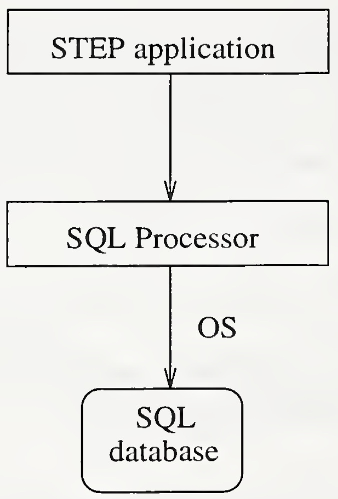

Figure 1: STEP/SQL Model 1

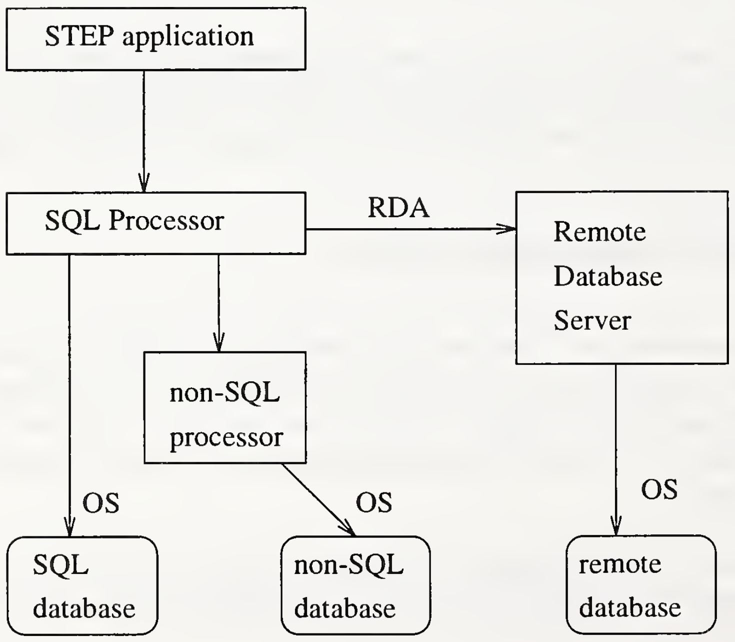

Figure 2: STEP/SQL Model 2

The second model expands the first to include remote databases, as well as the integration of non-SQL databases. SQL'92 and SQL3 have the required functionality for this model.[4] This is a simplification of the model presented in [GS92].

2.3 Security Responsibilities: The SQL Component

There are valid security considerations for each of the four areas of SQL functionality.

- Database Schema: The database schema must be well designed to ensure that aggregation and inference are not threats.
- Transaction Management: The SQL processor must prevent denial of service due to deadlock. The SQL processor must provide appropriate transaction management features: incomplete transactions can result in loss of external consistency, i.e., the tables and elements are not "synchronized."
- Modification: The SQL processor is responsible for maintaining access control for SQL level objects. The SQL processor is responsible for enforcing external consistency issues, such as type checking and ranges.
- Retrieval: The SQL processor is responsible for maintaining access control for SQL level objects.

2.4 Security Responsibilities: Non-SQL Components

Many aspects of security in a STEP environment will take place outside the SQL processor. This section outlines these responsibilities in the following areas: application interface, SQL interface to the physical database, SQL interface to non-SQL DBMS's, and interface to remote databases.

2.4.1 Application Interface

The interface between the SQL processor and STEP applications may utilize the embedded SQL language or the STEP Data Access Interface (SDAI) interface. The application must supply accurate information regarding the identity of the user to the SQL processor. This places two requirements on the system: appropriate selection

[4]An implementation of SQL'89 with a proprietary client-server model may also resemble this model. It is not possible to evaluate the security of such an implementation without details of its architecture.

and management of identification and authentication (I&A) controls and control of this critical attribute's propagation.

If the I&A control is weak or poorly managed, there is little assurance of accuracy for this attribute. Consider passwords where the account name and password are identical (a.k.a., "joe accounts"). If a STEP application accesses SQL for such an account, there is an increased probability that the actual user is not the authorized account user. Identity-based controls become useless.

Some systems include programs or features that allow users to modify their identity. The UNIX operating system, for example, includes the file attributes *suid*, which re-sets the user id, and *sgid*, which re-sets the group id. Termination of the program is supposed to cause the old user and group ids to resume. However, mis-use of these features may allow a user to continue to masquerade as the other user, executing STEP/SQL programs with unauthorized privileges.

2.4.2 SQL Interface to Physical Database

The OS provides the basic services that enable the SQL processor to store, retrieve, and modify data on the system. The OS is responsible for guaranteeing the integrity of the data and preventing denial of service.

The OS must also prevent data from being accessed outside the SQL processor. Such action can result in loss of integrity (improper modification) or confidentiality (by circumventing SQL's internal access controls).

2.4.3 SQL Interface to Non-SQL DBMS

This is unspecified in SQL'89. SQL'92 introduces the concepts of the SQL server and the SQL processor. By matching an SQL server with a non-SQL processor, SQL queries may be performed on non-conforming databases, provided the non-SQL processor is willing to provide an SQL view of its services and data.[GS92]

The interface between processor and server must be protected. Other processes on the system could eavesdrop, insert incorrect information, or perhaps even delete information. Clearly, these actions would result in loss of integrity or confidentiality.

2.4.4 Interface to Remote Databases

The RDA protocol is designed as an interface between a local SQL server and a remote SQL processor. The use of RDA on an open network may expose the system to many threats, including eavesdropping, packet replay, and host spoofing. These threats can be minimized by employing encryption techniques and strong authentication measures. RDA has built-in facilities which allow for the exchange of authentication data.

In short, the security achieved will be dependent upon the implementation. Stating that RDA is in use does not reveal anything about the security of the system.

3 Applicable Security Controls

SQL processors are required to have certain features in the areas of security policy and assurance that will readily apply to the security needs of implementations of STEP. There are additional security-relevant features in these areas that may also be present; however, they are not required for SQL compliance.

The SQL standard does not include specific requirements for accountability, but many relevant features could be included in compliant products. These include the ability to use advanced authentication mechanisms (such as smart cards) or audit trail generation.

3.1 Security Problems for implementing STEP

There are a number of security problems that are likely to be accentuated with STEP. The basic intent is to reliably exchange product data. In most cases, that exchange is intended to be limited to known partners. This implies a number of problems regarding authentication of host and user, inference, and aggregation. The system is inherently distributed. As a result, auditing is also a difficult problem. Finally, assurance is difficult in such an environment. SQL is well suited to address some of these problems. Others are outside the realm of SQL entirely. The following subsections will examine these areas and suggest how they might be addressed to suit a STEP environment.

3.1.1 Access Control

SQL processors are required to enforce discretionary access controls. The controls have a granularity of tables or views and include a variety of privileges: *INSERT*; *DELETE*; *SELECT*; *UPDATE*; and *REFERENCES*. Privileges may be allowed to propagate, or they may be static.[5]

Further controls may be imposed upon access to columns within a table by granting access privileges to views rather than the table itself. If the view and table are directly linked, modifications to the view will also modify the table.

SQL does not specify mandatory access controls. However, they can be supported in a variety of ways (see [PB93]). With a minor modification of the SQL semantics, polyinstantiation can be supported in the following way:

[5]The schema designer has the option of permitting users to share their privileges.

> All primary keys implicitly include the security label. All searches, etc. implicitly include a clause selecting the row with the highest security level that is dominated by the user's security level.

Note that this works nicely for levels of security, but is complex if categories are supported.

3.1.2 Inference and Aggregation

Inference and aggregation are not addressed in the SQL specifications, but can be addressed through add-on tools. These security problems are generally the result of security flaws in the design of the schema of the database. The SQL specifications do not address security flaws in schema design.

There is a flaw in the SQL'89 specification that can allow a user with DELETE or UPDATE privileges to interrogate a database and determine the values of rows in a table even if they do not have SELECT privilege. There is a simple work-around; users with DELETE or UPDATE privileges can be limited to accessing a view with appropriate fields. This flaw is eliminated in SQL'92.

3.1.3 Object Reuse

Object reuse is defined in [Rob91] as:

> The reassignment to some subject of a medium (e.g., page frame, disk sector, magnetic tape) that contained one or more objects. To be securely reassigned, such media must contain no residual data from the previously contained object(s).

Object reuse can pose a major disclosure threat. There are no requirements for object reuse in the SQL specifications. In addition, object reuse cannot be addressed through the SQL processor alone. Object reuse must also be addressed through the OS features.

Object reuse can be addressed in SQL by specifying default values for all columns that are not part of the primary key. This will ensure that no data is recovered accidentaly by creation of new records being placed upon old records. It may be possible to obtain that information in other ways, though.

The OS must ensure that processes outside of the SQL processor cannot obtain storage areas used by SQL without erasure of all data. This requires that the OS enforce separation of data areas and erase data by overwriting when an object is relinquished by a process.

3.2 Assurance

The SQL specification requires SQL processors to maintain serializability of transactions. Other transaction management features are also included, such as ROLLBACK and COMMIT statements. These statements help ensure integrity of the database.[6]

3.3 Accountability

SQL does not specify any accountability features. Identification and authentication is implicitly required for the access control features, but implementation is entirely unspecified. This means that any authentication requirements could be supported. For instance, an SQL-compliant processor might re-authenticate for each transaction using a smart card. It might also obtain the identity from the OS and assume that it was correct for the duration of the session.

SQL does not specify any auditing requirements. Products will vary widely in their ability to generate audit records and the granularity of the objects at which auditing occurs.

[6]SQL'92 allows the user to relax the serializability requirements when in read-only mode. This may affect the integrity of the transaction's *result*, but will not affect the integrity of the database.

4 Security Policies

An SQL-compliant DBMS can be used to implement STEP with adequate support for most security policies. However, not all SQL-compliant DBMS's will be appropriate for every security policy. The appropriate security policy must be determined before the system acquisition phase begins. It may be possible to augment OS mechanisms with add-on tools, but if the SQL processor is inappropriate it may need to be replaced entirely.

As noted previously, STEP does not imply any specific security policy. In this section, we assume a number of different security policies and try to demonstrate the ability of SQL to enforce these policies. The security policies examined are: a role based policy, a mandatory access control policy for confidentiality, and mandatory access control policy for integrity.

4.1 Role Based Security Policy

In a role based security policy, privileges are assigned according to the person's role(s), or job functions, in an organization. As a simple example, consider a system with manufacturing information for computer peripherals, specifically one containing design specifications for an optical mouse. This system would have the following users: a database administrator, a functional design engineer, an ergonomic design engineer, and systems staff. Different jobs require different access to the DBMS. The database administrator needs to design the schema, assign privileges, and review audit trails. The functional design engineer needs access to specifications regarding the mechanical functions of the mouse. The ergonomic design engineer needs access to physical design requirements. The systems staff make backups and create applications.

SQL'89 specifies powerful discretionary access control (DAC) features. These controls provide access to DBMS tables and views, restricting access to specified actions. The actions themselves are more constrained than typically provided by the OS. Use of views can help control access to columns within tables.

SQL'92 makes minor enhancements, such as removing the "bad semantics" associated with the DELETE and UPDATE commands (see Sec 3.1.2). This allowed users to gather data by inference. For additional details, see [PB93, pg. 22]. SQL'92 also specifies the syntax and semantics of the REVOKE statement.

SQL-3 may improve upon the administration of access rights to database objects through the use of roles [PB93, Sec 3.1.3]. This will greatly simplify the database administrator's job.

These controls should be augmented by appropriate authentication mechanisms, auditing, and inference detection tools. Authentication mechanisms may be provided by the STEP implementiation. If the STEP implementiation requires frequent re-authentication (e.g., for each transaction), the SQL processor must support this. Audit trails may be protected by the OS, but they must be generated by the SQL processor[7]. Inference detection tools are an add-on tool for schema design.

Any SQL-compliant processor can be applied to enforcement of a role-based security policy. The basic DAC features are sufficient, if not ideal. SQL'92 has significant enhancements for this task, but SQL3 may be most appropriate if the final specification includes role-based access controls. For any version of SQL, role-based security requires that the OS provide adequate authentication mechanisms and protect SQL data and audit trails from modification outside the SQL processor. An SQL-compliant processor will augment these features with strong DAC features appropriate for role-based policies. The SQL process should include one additional feature: audit trail generation by the SQL processor. It will also be necessary to perform procedures to detect opportunities for inference; this is best performed with automated inference detection tools.

4.2 Mandatory Access Control Policy for Confidentiality

The majority of mandatory access control (MAC) models specify a security policy for the enforcement of a level and category-based mandatory security policy. In such a policy, each object has two security attributes: a security level (such as SECRET or SENSITIVE) and an optional set of categories. Categories specify "need-to-know" areas, with the level specifying minimum clearance requirements. These security attributes are often called the *label.*

The primary concerns in such a policy are enforcing access rights and maintaining appropriate labels as new objects are created. The policy is often described as "read down, write up". A user may read an object with a lower level; an object will be written with the highest level of any objects involved. For example, a user with SECRET clearance can read SENSITIVE objects. If that user uses a SECRET object and a SENSITIVE object as input, the output of the process will be labeled as SECRET.

SQL can be used when a MAC policy is required, but not every SQL processor is suited to the task. SQL has no built in feature for handling labels, however nothing in the standard prevents an implementation from using and main-

[7]SQL does not require audit trail generation. This must be added as a value-added enhancement to the SQL processor.

taining this information. First, an appropriate MAC implementation of SQL for the target OS is required. If the target OS is a MAC system, the Trusted Computing Base (TCB) subset architecture and Trusted Subject architecture can be used. If the target OS is not a MAC system, the integrity lock architecture is required[8]. Add-on tools for detection and elimination of inference and aggregation are required. Authentication and auditing requirements are identical to those for role-based systems.

4.3 Mandatory Access Control Policy for Integrity

The Clark-Wilson framework[CW89], the most complete model for integrity, specifies an access control policy which depends upon the user's identity, the program to be executed, and the set of data files that will be accessed. The critical programs are called *trusted processes*, or TPs, and the data files are known as *constrained data items*, or CDIs. Few systems support such complex access control decisions.

The complexity of the access control decision closes the confidentiality loophole associated with the DAC policies. With DAC policies, user A may disclose information intended solely for user B. User B can then copy that information and make it available to anyone. In the Clark-Wilson framework it is possible to permit users to access an object, without allowing them to copy it. The TP determines the access privileges of any output files (implicitly or explicitly), so disclosure can be tightly controlled.

These access controls would interact with the SQL discretionary controls. Determining the net effect of this combination would not be straightforward, but there is nothing to preclude such a combination.

Clark-Wilson also requires a variety of data integrity checks on both objects and the aggregate data set. (These are known as internal and external consistency.) The most important of these is the Integrity Verification Procedure (IVP). The IVP is a critical feature of the Clark-Wilson framework, ensuring the consistency of CDIs as a whole.

All SQL-compliant processors include mechanisms and controls to maintain integrity. The most significant of these controls are integrity constraints and transaction management features. The integrity constraints enforce simple integrity rules (such as data type or value of a column). Transaction management ensures that the database moves from one consistent state to another.

SQL'92 includes significant enhancements in such mechanisms, such as domains, triggers, and check constraints. Domains can be used to restrict data values; triggers can be useful in auditing and warning mechanisms. Most significantly, the Clark-Wilson

[8]These architectures are discussed more in [PB93].

Integrity Verification Procedure (IVP) can be implemented for a table by using the enhanced check constraint. The IVP is a critical feature of the Clark-Wilson framework, ensuring the consistency of CDIs as a whole.

An SQL-compliant processor can be used to great advantage when implementing the Clark-Wilson framework on a system. Doing so requires an OS that includes: direct support for Clark-Wilson type access controls; appropriate authentication mechanisms; and appropriate auditing mechanisms. An SQL-compliant processor will augment these features with: integrity constraints for internal *and* external consistency; and transaction management features. Several additional features are required; the SQL processor must generate audit records and handle re-authentication if desired.

4.4 Summary

Any SQL-compliant system will include adequate controls to enforce role-based security policies. SQL'92 and SQL3 include additional features that improve support (in comparison to SQL'89). In particular, the current draft of SQL3 includes substantial enhancements for specification of roles. This will greatly simplify the database administrator's job.

Since the basic controls enforce role-based policies, there are minimal demands placed upon the OS. The system must restrict access to the database and audit trail outside the DBMS. The system may also contribute identification and authentication mechanisms, although these could be embedded in the DBMS. If the system's architecture employs fault tolerance, it may contribute in the area of assurance as well.

The features required by the SQL specification are *not* sufficient to enforce mandatory policies. Whether oriented towards integrity or confidentiality, mandatory policies are not addressed by the basic SQL access control mechanisms. These mechanisms may be enforced by the OS, or by enhanced implementations of SQL.

Appropriate access control mechanisms are not sufficient for enforcing mandatory security policies. These policies place additional requirements on the DBMS and OS beyond those specified for role-based policies.

If the security policy specifies mandatory access for confidentiality, tools are required to address inference and aggregation. These tools are not specified by the SQL specifications.

For integrity policies, additional burdens are placed upon the applications themselves to perform range and type checks upon the data values and external consistency checks on the tables of data. These types of tests are supported directly by features specified within SQL. These mechanisms were significantly enhanced in SQL'92 (compared to SQL'89).

SQL is most suited to enforcement of role-based policies. It includes significant features to support mandatory policies for integrity, but omits appropriate access control. SQL does not address mandatory access control for confidentiality, but does not preclude integration of appropriate controls.

References

[ANS89a] Database language - SQL with integrity inhancements. American National Standard X3.135, American National Standards Institute, 1989.

[ANS89b] Database language - embedded SQL. American National Standard X3.168, American National Standards Institute, 1989.

[ANS92] Database language SQL. American National Standard X3.135-1992, American National Standards Institute, 1992.

[Cam90] John Campbell. A brief tutorial on trusted database management systems (executive summary). In *13th National Computer Security Conference Proceedings*, 1990.

[CO92] S.J. Cannan and G.A.M. Otten. *SQL - The Standard Handbook.* McGraw-Hill Book Co., Berkshire SL6 2QL England, October 1992.

[CW89] David Clark and David Wilson. A comparison of commercial and military computer security policies. In *Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS).* NIST Special Publication 500-160, 1989.

[DD92] C.J. Date and Hugh Darwen. *A Guide to the SQL Standard.* Addison-Wesley Publishing, Reading, MA 01867 USA, October 1992.

[DJ92] Vinti M. Doshi and Sushil Jajodia. Enforcing entity and referential integrity in multilevel secure databases. In *15th National Computer Security Conference Proceedings*, 1992.

[FCF93] Allison Barnard Feeney, Stephen Nowland Clark, and James E. Fowler. Requirements for an application protocol development environment. NISTIR 5197, National Institute of Standards and Technology, May 1993.

[FIP90] Database language SQL. Federal Information Processing Standard 127-1, National Institute of Standards and Technology, 1990.

[FIP93] Database language SQL. Federal Information Processing Standard 127-2, National Institute of Standards and Technology, June 1993.

[GS92] Leonard Gallagher and Joan Sullivan. Database language SQL: Integrator of CALS data repositories. NISTIR 4902, National Institute of Standards and Technology, September 1992.

[ISO90a] Remote database access - part 1: Generic model. ISO/JTC1/SC21 N4282, Information Processing Systems - Open Systems Interconnect, 1990.

[ISO90b] Remote database access - part 2: SQL specialization. ISO/JTC1/SC21 N4281, Information Processing Systems - Open Systems Interconnect, 1990.

[NIP91] National initiative for product data exchange. Implementation plan, Department of Commerce/Department of Defense, November 1991.

[PB93] W. Timothy Polk and Lawrence E. Bassham III. Security issues in the database language SQL. Special Publication 800-8, National Institute of Standards and Technology, August 1993.

[PDE92] Industrial automation systems and integration – product data representation and exchange – part 11: Description methods: The express languagereference manual. Draft International Standard ISO/IEC 10303-11, ISO/IEC, 1992.

[PDE93a] Industrial automation systems and integration – product data representation and exchange – part 1: Overview and fundamental principles. Draft International Standard ISO/IEC 10303-1, ISO/IEC, 1993.

[PDE93b] Industrial automation systems and integration – product data representation and exchange – part 21: Clear text encoding of the exchange structure. Draft International Standard ISO/IEC 10303-21, ISO/IEC, 1993.

[PDE93c] Industrial automation systems and integration – product data representation and exchange – part 31: Framework for conformance testing. Draft International Standard ISO/IEC 10303-31, ISO/IEC, 1993.

[Rob91] Edward Roback. Glossary of computer security terminology. NISTIR 4659, National Institute of Standards and Technology, September 1991.

[SFD92] Linda M. Schlipper, Jarrellann Filsinger, and Vinti M. Doshi. A multilevel secure database management system benchmark. In *15th National Computer Security Conference Proceedings*, 1992.

[TDI91] Trusted database management system interpretation. NCSC-TG 021, National Computer Security Center, April 1991.

[Thu92] Bhavani Thuraisingham. Knowledge-based inference control in a multilevel secure database management system. In *15th National Computer Security Conference Proceedings*, 1992.

[TR791] Integrity in automated information systems. C Technical Report 79-91, National Computer Security Center, September 1991.