Return to the FIPS
Home Page

FIPS PUB 127-2
Supersedes FIPS PUB 127
1990 February 2

Federal Information
Processing Standards Publication 127-2

1993 June 02
Announcing the Standard for

Database Language SQL

)ee Important Change Notice at the end of this document.
(The Foreword, Abstract, and Key Words
can be found at the end of this document.)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949, as amended by the Computer Security Act of 1987, Public Law 100-235.

1. Name of Standard. Database Language SQL (FIPS PUB 127-2).

2. Category of Standard. Software Standard, Database.

3. Explanation. This publication is a revision of FIPS PUB 127-1 and supersedes that document in its entirety. It provides a substantial, upward- compatible enhancement of Database Language SQL. It includes four levels of conformance: Entry SQL, Transitional SQL, Intermediate SQL, and Full SQL. Entry SQL is a minor enhancement over the minimum requirements of FIPS PUB 127-1, Intermediate SQL is a major enhancement over Entry SQL, and Full SQL is a major enhancement over Intermediate SQL. Transitional SQL is a temporary FIPS specification that falls approximately half way between Entry SQL and Intermediate SQL. Conformance to Entry SQL is required in all Federal procurements of SQL products. Conformance to Transitional SQL, Intermediate SQL, or Full SQL are options that may be specified, explicitly, as requirements in a Federal procurement. Section 13 identifies the minimum requirements for conformance to Entry SQL in FIPS PUB 127-2 that differ from the minimum requirements for conformance to FIPS PUB 127-1, and Section 14 defines requirements for the three additional levels of conformance.

This publication announces adoption of American National Standard Database Language SQL, ANSI X3.135-1992, as the Federal Information Processing Standard for Database Language SQL (FIPS SQL).

The exact specification is in Section 10 of this standard.

ANSI SQL is a revision and replacement of two previous American National Standards, ANSI X3.135-1989 and ANSI X3.168-1989. It specifies the syntax and semantics of SQL language facilities for defining and accessing SQL databases. These facilities include:

-)chema definition, to declare the structures, integrity constraints, and access privileges of a database.
-)chema manipulation, to alter a schema definition.
- Data manipulation, to populate a database and access SQL-data.
- Transaction management, to define and manage SQL-transactions.
- Connection management, to establish and manage SQL-connections.
-)ession management, to set the attributes of an SQL-session.
- Dynamic SQL, to provide facilities for dynamic construction and execution of SQL statements.
- Diagnostics management, to communicate constraint violations and warnings to applications.
- Information schema tables, to provide an SQL description of schema definitions.
- Programming language bindings, to declare database procedures that may be called from various programming languages.
- Embedded SQL, to define how SQL statements may be syntactically embedded into one of the following programming languages: Ada, C, COBOL, FORTRAN, MUMPS, Pascal, or PL/I. Embedded SQL was formerly defined in ANSI X3.168-1989.

ANSI SQL is specified in three levels: Entry SQL, Intermediate SQL, and Full SQL. Entry SQL is a minor enhancement of ANSI X3.135-1989 (see Section 13). Intermediate SQL adds provisions for schema manipulation, dynamic SQL, diagnostics management, long identifiers, multiple module support, cascade delete for referential integrity, multiple schemas per authorization identifier, DATE and TIME data types, domains, variable length character strings, support for national character sets, and substantial enhancements for data manipulation. The data manipulation enhancements in Intermediate SQL include: a CASE expression, CAST functions between data types, string operations, natural join, outer join, union join, row value expressions, and subqueries in value expressions, as well as table operations for union, intersection, and complement. Full SQL adds provisions for connection management, session management, pre-defined character translations and form-of-use conversions, a BIT string data type, deferrable integrity constraints, derived tables in the FROM clause, subqueries in CHECK clauses, insensitive cursors, self-referencing data operations, assertions, and temporary tables. A list of optional FIPS SQL features, comprising all of the additional facilities in ANSI Intermediate SQL and Full SQL, is defined in Section 14 of this standard.

The purpose of FIPS SQL is to promote portability and interoperability of database application programs, to facilitate maintenance of database systems among heterogeneous data processing environments, and to allow for the efficient exchange of programmers among different data management projects. The standard is used by implementors as the reference authority in developing a FIPS conforming relational model database management system, with standard programming language interfaces to that database management system. The standard is used by application programmers to help write SQL conforming applications and by other computer professionals who need to know the precise syntactic and semantic rules of Database Language SQL.

4. Approving Authority. Secretary of Commerce.

5. Maintenance Agency. Department of Commerce, National Institute of Standards and Technology (Computer Systems Laboratory)

6. Cross Index

a. American National Standard Database Language SQL, ANSI X3.135-1992 (revision of ANSI X3.135-1989 and replacement of ANSI X3.168-1989).

b. ISO/IEC 9075:1992, Database Language SQL (revision of ISO/IEC 9075:1989).

Note: Except for a different Foreword, Introduction, and Normative references, ANSI X3.135-1992 and ISO/IEC 9075:1992 are identical documents.

7. Related Documents

a. Federal Information Resources Management Regulations (FIRMR) subpart 201.20.303, Standards, and subpart 201.39.1002, Federal Standards, April 1992.

b. Federal ADP and Telecommunication Standards Index, U.S. General Services Administration, Information Resources Management Service, October 1992 (updated periodically).

c. NIST, Validated Products List: Programming Languages, Database Language SQL, Graphics, GOSIP, POSIX, Security; Judy B. Kailey, Editor, NISTIR 5103, issue No. 1, January 1993 (republished quarterly). Available by subscription from the National Technical Information Service (NTIS).

d. FIPS PUB 21-3, Programming Language COBOL, 1990.

e. FIPS PUB 69-1, Programming Language FORTRAN, 1985.

f. FIPS PUB 109, Programming Language Pascal, 1985.

g. FIPS PUB 119, Programming Language Ada, 1985.

h. FIPS PUB 125, Programming Language MUMPS, 1986 (Revision expected in 1993).

i. FIPS PUB 160, Programming Language C, 1991.

j. FIPS PUB 146, Government Open Systems Interconnection Profile (GOSIP). A revision to FIPS PUB 146-1, including Remote Database Access (RDA) specifications, is planned for mid-1993. To be issued in conjunction with IGOSS.

k. IGOSS, Industry/Government Open Systems Specification, publication planned mid-1993. This specification will reference "stable agreements" from the NIST OSI Implementor's Workshop as of December 1992.

l. NIST SP 500-206, Stable Implementation Agreements for Open Systems Interconnection Protocols, Version 6, Edition 1, NIST Workshop for Implementors of Open Systems Interconnection, December 1992.

m. ISO/IEC DIS 9579-1, Information Technology - Open Systems Interconnection - Remote Database Access - Part 1: Generic model, service, and protocol, document ISO/IEC JTC1/SC21 N6375, August 1991.

n. ISO/IEC DIS 9579-2, Information Technology - Open Systems Interconnection - Remote Database Access - Part 2: SQL specialization, document ISO/IEC JTC1/SC21 N6376, August 1991.

o. ISO/IEC 10026, Information Technology - Open Systems Interconnection - Distributed Transaction Processing - Part 1: OSI TP Model, Part 2: OSI TP Service, Part 3: OSI TP Protocol Specification, International Standard, December 1992.

p. SQL Information Bulletin, Number 1, SQLIB-1, Interpretations of ANSI X3.135-1989, available from Global Engineering Documents, April 1991.

q. FIPS PUB 29-2, Interpretation Procedures for FIPS Software, 14 September 1987.

r. ISO 646, Information Processing - ISO 7-bit coded character set for information interchange, 2nd edition, Third Edition, December 1991.

s. ISO 4873, Information Processing - ISO 8-bit code for information interchange - Structure and rules for implementation, Third Edition, 1991. Replaces ANSI X3.134.1, 8-bit ASCII.

t. ISO 8859-1, Information processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1, February 1987. Replaces ANSI X3.134.2.

u. ISO/IEC CD 11404, Information Technology - Programming Languages - Language Independent Data Types (CLID), document JTC1/SC22/WG11 N345, December 1992.

8. Objectives. The FIPS for Database Language SQL permits Federal departments and agencies to

exercise more effective control over the production, management, and use of the Government's information resources. The primary objectives are:

- to encourage more effective utilization and management of database application programmers by ensuring that skills acquired on one project are transportable to other projects, thereby reducing the cost of database programmer retraining.

- to reduce overall software costs by making it easier and less expensive to maintain database definitions and database application programs and to transfer those definitions and programs among different computers and database management systems, including replacement database management systems.

- to promote communication and interoperability among data installations conforming to FIPS SQL and related GOSIP communications standards.

- to reduce the cost of software development by achieving increased database application programmer productivity through the understanding and use of database methods employing standard structures and operations, standard data types, standard constraints, and standard interfaces to programming languages.

- to protect the software assets of the Federal government by ensuring to the maximal feasible extent that Federal database management system standards are technically sound and that subsequent revisions are compatible with the installed base.

Government-wide attainment of the above objectives depends upon the widespread availability and use of comprehensive and precise standard database management system specifications.

9. Applicability

9.1 Database Language SQL is one of the database language standards provided for use by all Federal departments and agencies. These database language standards should be used for all computer database applications and programs that are either developed or acquired for government use. FIPS SQL is particularly well suited for use in database applications that employ the relational data model. The relational data model is appropriate for applications requiring flexibility in the data structures and access paths of the database. The relational data model is desirable where there is a substantial need for ad hoc data manipulation, and data restructuring, in addition to the need for access by static applications under production control.

9.2 FIPS SQL shall be used for relational database applications and programs when one or more of the following situations exist:

- It is anticipated that the life of the database application will be longer than the life of the presently utilized equipment or database management system, if any.

- The database application is under constant review for updating of the specifications, and changes

may result frequently.

- The database application is being designed and developed centrally for a decentralized system that employs computers of different makes and models or database software acquired from a different vendor.

- The database application will or might be run under a database management system other than that for which the database application is initially written.

- The database application is to be understood and maintained by programmers other than the original ones.

- The database application is one part of a distributed application that requires exchange of data or interoperation of the various parts.

- The database application is or is likely to be used by organizations outside the Federal government (e.g., Federal government contractors, State and local governments, and others).

9.3 Nonstandard language features shall be used only when the needed operation or function cannot reasonably be implemented with the standard features alone. A needed language feature not provided by the FIPS database languages should, to the extent possible, be acquired as part of an otherwise FIPS conforming database management system. Although nonstandard language features can be very useful, it should be recognized that their use may make the interchange of programs and future conversion to a revised standard or replacement database management system more difficult and costly.

9.4 Although this standard does not specifically address interactive database access through graphical user interfaces (GUI), the SQL statements specified by this standard are appropriate for such use. In a Client/Server environment, a GUI client may use SQL statements to access SQL conformant server databases.

9.5 Although this standard does not specifically address distributed database management systems or distributed database applications, the connection management statements defined in this standard may be used, along with facilities for remote database access (ISO/IEC 9579) and distributed transaction processing (ISO/IEC 10026), to access SQL-data at remote nodes in a distributed system and to present a global view to application programs.

9.6 Although this standard does not specifically address user-defined data types, class hierarchies, inheritance, polymorphism, or other features of object database management systems, such capabilities are upward compatible extensions of this standard and may be specified in a future revision of FIPS SQL (see Section 16.8).

9.7 It is recognized that some programmatic requirements may be more economically and efficiently satisfied through the use of a database management system employing a different data model than those provided by the FIPS database languages or the use of a database management system that functionally conforms to a FIPS database language but does not conform to all other aspects of the FIPS. The use of

any facility should be considered in the context of system life, system cost, data integrity, and the potential for data sharing.

9.8 Some programmatic requirements may be more economically and efficiently satisfied by the use of automatic program generators or by database access through other high-level language information processing systems. However, if the final output of a program generator or high-level language system is language that accesses a relational database, then that language shall conform to the conditions and specifications of SQL.

10. Specifications. FIPS SQL adopts all provisions of ANSI X3.135- 1992, Database Language SQL, with the exceptions listed below:

- a. FIPS SQL requires conformance to Entry SQL. Conformance to Transitional SQL, Intermediate SQL, or Full SQL are options that may be specified explicitly in SQL procurements (see Section 14).

- b. FIPS SQL does not include PL/I language bindings, since PL/I is not a FIPS programming language.

- c. FIPS SQL does not recognize conformance solely by "direct invocation and processing of SQL language" as specified in Subclause 23.2 of ANSI X3.135-1992, because direct invocation does not mandate all of the facilities desired in a FIPS SQL conforming product. Conformance to FIPS SQL requires a Module or Embedded SQL binding style to one or more FIPS programming languages.

- d. FIPS SQL requires that the "SQL Flagger" be implemented in Entry SQL in addition to Intermediate SQL and Full SQL. This is because FIPS SQL has always included a flagger requirement, even from its first specification in 1987. For conformance to Entry SQL, FIPS SQL requires "Entry SQL Flagging" with the "Syntax Only" extent of checking option as defined in Subclause 4.33 of ANSI X3.135-1992. The SQL Flagger is required for each language binding style, including "Interactive Direct SQL" (see Section 16.5).

- e. For conformance to Intermediate SQL or to Full SQL, FIPS SQL requires implementation of the following named character sets: SQL_CHARACTER, ASCII_GRAPHIC, LATIN1, ASCII_FULL, and SQL_TEXT. The form-of-use and default collation requirements for these character sets are defined in Section 16.7 of this standard.

- f. For conformance to Intermediate SQL or to Full SQL, FIPS SQL requires implementation of the FIPS_DOCUMENTATION schema, as specified in Section 15 of this standard.

11. Implementation. Implementation of this standard involves four areas of consideration: the effective date, acquisition of FIPS SQL implementations, interpretation of FIPS SQL, and validation of FIPS SQL implementations.

11.1 Effective Date. This publication is effective six (6) months after the date of publication of a final

document in the Federal Register announcing its approval by the Secretary of Commerce. Prior to that date the requirements of FIPS PUB 127-1 apply to Federal SQL procurements. This delayed effective date is intended to give implementations that conform to FIPS PUB 127-1 time to make the enhancements necessary to enable conformance to Entry SQL (see Section 13). No further transitional period is necessary.

11.2 Acquisition of SQL Implementations. Relational model database management systems acquired for Federal use shall implement FIPS SQL. Conformance to FIPS SQL is required whether SQL implementations are developed internally, acquired as part of an ADP system procurement, acquired by separate procurement, used under an ADP leasing arrangement, or specified for use in contracts for programming services. Recommended terminology for procurement of FIPS SQL is contained in the U.S. General Services Administration publication ***Federal ADP & Telecommunications Standards Index***, Chapter 4 Part 1.

11.3 Interpretation of FIPS SQL. NIST provides for the resolution of questions regarding FIPS SQL specifications and requirements, and issues official interpretations as needed. Procedures for interpretations are specified in FIPS PUB 29-2. All questions about the interpretation of FIPS SQL should be addressed to:
Director
Computer Systems Laboratory
ATTN: Database Language SQL Interpretation
National Institute of Standards and Technology
Gaithersburg, MD 20899
Telephone: (301) 975-2833

11.4 Validation of SQL Implementations. Implementations of FIPS SQL shall be validated in accordance with NIST Computer Systems Laboratory (CSL) validation procedures for FIPS SQL. Recommended procurement terminology for validation of FIPS SQL is contained in the U.S. General Services Administration publication ***Federal ADP & Telecommunications Standards Index***, Chapter 4 Part 2. This GSA publication provides terminology for three validation options: Delayed Validation, Prior Validation Testing, and Prior Validation. The agency shall select the appropriate validation option and shall specify whether a Validation Summary Report or Certificate of Validation is required. The agency shall specify appropriate time frames for validation and correction of nonconformities. The agency is advised to refer to the NIST publication ***Validated Products List*** for information about the validation status of SQL products. This information may be used to specify validation time frames that are not unduly restrictive of competition.

The agency shall specify the criteria used to determine whether a Validation Summary Report (VSR) or Certificate is applicable to the hardware/software environment of the SQL implementation offered. The criteria for applicability of a VSR or Certificate should be appropriate to the size and timing of the procurement. A large procurement may require that the offered version/release of the SQL implementation shall be validated in a specified hardware/software environment and that the validation shall be conducted with specified hardware/software features or parameter settings; e.g. the same parameter settings to be used in a performance benchmark. An agency with a single-license procurement may review the ***Validated Products List*** to determine the applicability of existing VSRs or Certificates

to the agency's hardware/software environment.

Implementations shall be evaluated using the NIST SQL Test Suite, a suite of automated validation tests for SQL implementations. The NIST SQL Test Suite was first released in August 1988 to help users and vendors determine compliance with FIPS SQL. Version 3.0 of the test suite was released in January 1992, to be used for validating conformance to FIPS PUB 127-1 after July 1, 1992. It is expected that Version 4.0 of the test suite will be available in mid-1993, to be used for testing conformance to Entry SQL of FIPS PUB 127-2 after the effective date. The results of validation testing by the SQL Testing Service are published on a quarterly basis in the ***Validated Products List***, available from the National Technical Information Service (NTIS).

Each release of the test suite has provided additional interfaces and test cases to increase the test suite's coverage of the SQL language. Version 3.0 of the NIST SQL Test Suite provides 11 test suite types (interfaces): Embedded (pre-processor) Ada, Embedded C, Embedded COBOL, Embedded FORTRAN, Embedded Pascal, module language Ada, module language C, module language COBOL, module language FORTRAN, module language Pascal, and Interactive Direct SQL. Version 3.0 does not include tests for Embedded MUMPS or module language MUMPS because the MUMPS programming language interface is not defined in FIPS 127-1; such tests may be available in Version 4.0 for testing of FIPS 127-2. There are additional tests in Version 3.0 for the Integrity Enhancement Feature, default database sizing constructs, and the FIPS Flagger requirement of FIPS 127-1.

An SQL Test Suite license includes all of the tests described above, documentation, and automatic notifications of approved changes to the SQL Test Suite for a six month period. A license for SQL Test Suite Version 3.0 is a necessary requirement for an organization that wishes to be tested by the NIST SQL Testing Service between July 1, 1992 and the effective date of FIPS 127-2.

Current information about the NIST SQL Validation Service and validation procedures for FIPS SQL is available from:
National Institute of Standards and Technology
Computer Systems Laboratory
Software Standards Validation Group
Building 225, Room A266
Gaithersburg, Maryland 20899
(301) 975-2490
13.

12. Waivers.

Under certain exceptional circumstances, the heads of Federal departments and agencies may approve waivers to Federal Information Processing Standards (FIPS). The head of such agency may redelegate such authority only to a senior official designated pursuant to section 3506(b) of Title 44, U.S. Code. Waivers shall be granted only when:

- a. Compliance with a standard would adversely affect the accomplishment of the mission of an operator of a Federal computer system, or

- b. Cause a major adverse financial impact on the operator which is not offset by Government-wide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision which explains the basis on which the agency head made the required finding(s). A copy of each such decision, with procurement sensitive or classified portions clearly identified, shall be sent to: National Institute of Standards and Technology; ATTN: FIPS Waiver Decisions, Technology Building, Room B-154; Gaithersburg, MD 20899.

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Governmental Affairs of the Senate and shall be published promptly in the ***Federal Register***.

When the determination on a waiver applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment to such notice.

A copy of the waiver, any supporting documents, the document approving the waiver and any supporting and accompanying documents, with such deletions as the agency is authorized and decides to make under 5 U.S.C. Sec. 552(b), shall be part of the procurement documentation and retained by the agency.

13. New FIPS SQL Requirements. Conformance to Entry SQL requires additional capabilities from an SQL implementation beyond those required for minimal conformance to FIPS PUB 127-1. The following list identifies the additional capabilities required. All terms delimited by angle brackets (i.e. **(**...**)**) refer to syntactic productions specified in ANSI X3.135-1992.

1. Integrity enhancement. The Integrity Enhancement Feature was an option in FIPS PUB 127-1. Many implementations of the existing FIPS SQL support this feature and both ISO and ANSI standardization groups have made it required for all conforming Entry SQL implementations. This feature is now required in FIPS SQL. It includes any explicit or implicit **(**unique constraint definition**)**, **(**referential constraint definition**)**, **(**check constraint definition**)**, or the **(**default clause**)**, each with specified restrictions for Entry SQL.

2. SQLSTATE status codes. This mechanism for returning exceptions to SQL statements augments the SQLCODE status codes originally specified. SQLCODE is now a deprecated feature. SQLSTATE specifies more than seventy-five (75) exception, completion, and warning codes, whereas SQLCODE specifies only three. SQLSTATE and SQLCODE are defined in Clause 22, "Status codes", of ANSI X3.135-1992.

3. Delimited identifiers. In the previous ANSI SQL specification, it was not possible for an application to specify identifiers with spaces or other special symbols. Also, it was not possible

to protect against future assaults on the name space for (identifier) by additions to the (reserved word) list. The new facility for (delimited identifier) allows a user to enclose all identifiers in double-quotation marks, thereby ensuring that the name defined or referenced may contain spaces or other special symbols and will not be impacted by future additions to the (reserved word) list.

4. Renaming columns. In the SQL language it is possible to reference (sort key) columns in a (cursor specification) by position number instead of by column name. This is because it was not possible in the previous standard to name derived columns resulting from (value expression). Reference by position number is now a deprecated feature; as an alternative, it is required to be able to name, or rename, any (derived column) with an (as clause).

5. Commas in parameter lists. In the previous ANSI SQL specification, items in a (parameter declaration list) were separated by spaces or other token separators. It is now possible to also separate such items by commas. This makes parameter lists compatible in style with all other lists in the SQL language.

6. SQL Errata. Several errors in the previous ANSI SQL specification have been corrected via announcement in the ***SQL Information Bulletin***, SQLIB-1, including: addition of a leading colon for (parameter name), corrections to the specification of WITH CHECK OPTION in a (view definition), clarification of the argument mode for Pascal parameters, and clarification of statement termination requirements for Embedded SQL in Pascal programs. Other errors in ANSI X3.135-1989 have also been corrected in ANSI X3.135-1992, including: additional overflow exceptions on data assignment, exceptions generated during evaluation of a (numeric value expression), correction of the data type for the SQLCODE variable in an (embedded SQL Ada program), removal of an ambiguity in the definition of WITH CHECK OPTION in nested view definitions, addition of a rule to prevent defining two (unique constraint definition) with identical unique column lists in the same table, additional requirements to enforce SELECT privileges, restrictions on circular view definitions, correction to the syntax of a (host label identifier) in an (embedded exception declaration), corrections for data type declarations in (parameter declaration) for COBOL and PL/I, correction to the specification of (COBOL integer type), and correction to the specification of PL/I support for SQLCODE. Most of these corrections are listed in Annex E, "Incompatibilities", and Annex F, "Maintenance and interpretation of SQL" of ANSI X3.135-1992. All such corrections that pertain to Entry SQL are now required elements of FIPS SQL.

Note: The term "deprecated", as used in ANSI SQL, means that a feature so labeled may not be supported in some future version of the standard, but it is still a fully supported and required feature of the existing standard.

14. Optional FIPS SQL features. FIPS SQL requires implementation of Entry SQL (see Section 10). Optionally, an SQL procurement may require conformance to Intermediate SQL, Full SQL, or Transitional SQL. Since implementations conforming to Intermediate SQL may not be available immediately, FIPS SQL specifies a temporary FIPS "Transitional SQL" approximately half way between Entry SQL and Intermediate SQL. FIPS Transitional SQL is intended to provide a common, near-term goal for SQL implementations that already have a number of features beyond Entry SQL.

Federal procurements may wish to specify Transitional SQL as a requirement during the interim period before Intermediate SQL implementations are widely available. It is expected that a future version of the NIST SQL Test Suite will provide conformance testing for both Entry SQL and Transitional SQL before tests are completed for testing Intermediate SQL or Full SQL. There is no requirement for the SQL Flagger to flag Transitional SQL features separately from Entry SQL Flagging or Intermediate SQL Flagging.

The following subsections partition FIPS SQL features into four subgroups: 1) required in Transitional SQL, 2) required in Intermediate SQL, 3) required in Full SQL, and 4) suitable in combination with Remote Database Access (RDA). The list of features in each subsection determines the required features for conformance to that level of FIPS SQL. In most cases, an SQL procurement will specify a mandatory base level of conformance along with a list of desirable features. Desirable features may be used to support the evaluation of a product offered in response to a procurement.

In the following subsections, all Clause and Subclause references, and all syntactic terms delimited by angle brackets (i.e. **(**...**)**) are from ANSI X3.135-1992.

14.1 Transitional SQL. The following FIPS SQL features should start becoming widely available in SQL conforming products in the near future. A conservative SQL procurement in an open systems environment could list these items as mandatory requirements with some degree of confidence that a competitive procurement would follow, at least by the end of the initial effective year of this standard.

1. Dynamic SQL. All provisions of Clause 17, "Dynamic SQL", with restrictions identified in the Leveling Rules for Intermediate SQL; removal of all Entry SQL Leveling Rules in Clause 17; removal of the Entry SQL restriction requiring that a **(**module contents**)** not be a **(**dynamic declare cursor**)**, as specified in Leveling Rule 2b of Subclause 12.1, "**(**module**)**"; removal of Leveling Rule 2a of Subclause 6.2 that prohibits reference to a dynamic parameter in a **(**general value specification**)**. A **(**preparable statement**)** shall include all **(**preparable SQL data statement**)** that are otherwise supported for Entry SQL, as well as any other **(**preparable statement**)** for which non-preparable support is claimed by that implementation.

2. Basic information schema. Requires existence of an accessible INFORMATION_SCHEMA consisting of the following views defined in Subclause 21.2, "Information Schema": TABLES, VIEWS, and COLUMNS, all with any restrictions identified in the Leveling Rules for Intermediate SQL.

3. Basic schema manipulation. Support for the following schema definition and schema manipulation statements as **(**SQL statement**)** in an explicit or implicit **(**procedure**)**: Subclauses 11.3 through 11.9, "**(**table definition**)**"; Subclause 11.18, "**(**drop table statement**)**"; Subclause 11.19, "**(**view definition**)**"; Subclause 11.20, "**(**drop view statement**)**"; Subclause 11.10, "**(**alter table statement**)**", containing Subclause 11.11, "**(**add column definition**)**"; Subclause 11.10, "**(**alter table statement**)**", containing Subclause 11.15, "**(**drop column definition**)**"; Subclause 11.36, "**(**grant statement**)**"; and Subclause 11.37, "**(**revoke statement**)**"; all with any other restrictions identified in the Leveling Rules for Entry SQL, and all with any enhancements derived from other features claimed to be supported by the implementation. Removal of Leveling Rules 2a in

Subclauses 11.11, 11.15, 11.18, 11.20, and 11.37.

4. Joined table. All provisions for NATURAL JOIN, INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, (join condition), and (named columns join) from Subclause 7.5, "(joined table)", with restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Entry SQL Leveling Rule 2a of Subclause 6.3, "(table reference)", that prohibits (joined table) in a table reference. Removal of Entry SQL Leveling Rule 2c of Subclause 7.10, "(query expression)", that prohibits (joined table) in a (query expression). This feature does not include support for CROSS JOIN, UNION JOIN, or FULL OUTER JOIN.

5. DATETIME data types. Support for DATE, TIME, TIMESTAMP, and INTERVAL data types as defined in Subclause 6.1, "(data type)", with the exception of support for time zones and time zone management; support for (datetime literal) and (interval literal) as defined in Subclause 5.3, (literal); support for the following datetime operations: (datetime field) in an (extract expression) as defined in Subclause 6.6, (numeric value function), (datetime value function) defined in Subclause 6.8, (datetime value expression) defined in Subclause 6.14, (interval value expression) defined in Subclause 6.15, and (overlaps predicate) defined in Subclause 8.11 (Note: With support for (row value constructor) here); support for datetime comparison defined in Subclause 8.2, (comparison predicate); support for (interval qualifier) as specified in Subclause 10.1; support for (datetime value function) in a (default clause) as specified in Subclause 11.5, (default clause); all with restrictions identified in the Leveling Rules for Intermediate SQL. The Syntax Rules require, by default from Syntax Rule 26 of Subclause 6.1, (data type), a (timestamp precision) greater than or equal to 6 decimal digits. This feature does not include support for time zones, including: (time zone interval) (time zone field) (time zone) (time zone specifier) and (set local time zone statement) (see feature #41). Removal of the Entry SQL restriction on datetime data types as specified in Leveling Rule 2b of Subclause 6.1, (data type), and on datetime functions as specified in Leveling Rule 2a of Subclause 6.8, (datetime value function). Removal of the Entry SQL restriction on datetime literals as specified in Leveling Rule 2b of Subclause 5.3, on datetime value expressions as specified in Leveling Rules 2a and 2b of Subclause 6.11, Leveling Rule 2a of Subclause 6.14, and Leveling Rule 2a of Subclause 6.15, on the overlaps predicate as specified in Leveling Rules 2a of Subclause 8.1 and Subclause 8.11, and for specifying the field precision of an INTERVAL data type or literal as specified in Leveling Rule 2a of Subclause 10.1, (interval qualifier), all with the exception of retaining the restrictions on time zone support.

6. VARCHAR data type. Support for CHARACTER VARYING, and its syntactic shorthands VARCHAR and CHAR VARYING, as defined in Subclause 6.1, (data type); support for the following character operations: (length expression) defined in Subclause 6.6, (numeric value function), (character substring function) defined in Subclause 6.7, (string value function), (concatenation> and (character value expression> defined in Subclause 6.13, (string value expression); support for comparison of fixed and variable length character strings as defined in Subclause 8.2, (comparison predicate); support for CHARACTER VARYING in any (embedded SQL host program) supported by the implementation; all with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of the Entry SQL requirement that at least one (character representation) be present in a (character string literal) as specified in Leveling Rule 2c

of Subclause 5.3, (literal). Removal of Leveling Rule 2a of Subclause 6.1, (data type). Removal of the Entry SQL requirement that a (numeric value function) not be a (length expression) as specified in Leveling Rule 2a of Subclause 6.6, (numeric value function). Removal of the Entry SQL restriction against using the SUBSTRING function, as specified in Leveling Rule 2a of Subclause 6.7, ()tring value function). Removal of Leveling Rule 2a of Subclause 6.13, (string value expression), that prohibits concatenation in Entry SQL. Removal of Leveling Rule 2a of Subclause 12.3, (procedure), that prohibits specification of CHARACTER VARYING in an Entry SQL (procedure). Removal of Leveling Rule 2a of Subclause 19.4, (embedded SQL C program), and Leveling Rule 2a of Subclause 19.9, (embedded SQL PL/I program).

7. TRIM function. Removal of the Entry SQL restriction against specifying a (trim function) as an alternative in a (character value function) as specified in Leveling Rule 2b of Subclause 6.7, (string value function).

8. UNION in views. Removal of the Entry SQL restriction against specification of UNION in a (view definition) as specified by the Entry SQL Leveling Rule 2a of Subclause 11.19, (view definition). Support (query expression> in a (view definition) provided that the (query expression) abides by the other restrictions for Entry SQL, as specified in Leveling Rule 2 of Subclause 7.10, (query expression).

9. Implicit numeric casting. Removal of all Entry SQL restrictions for operations involving the assignment of approximate numeric values to exact numeric types, including: Leveling Rule 2c of Subclause 13.3, (fetch statement); Leveling Rule 2a of Subclause 13.5, (select statement: single row); Leveling Rule 2b of Subclause 13.8, (insert statement); Leveling Rule 2a of Subclause 13.9, (update statement: positioned); Leveling Rule 2a of Subclause 13.10, (update statement: searched).

10. Implicit character casting. Removal of all Entry SQL restrictions for operations involving the assignment of character string values to character string types, including: Leveling Rule 2c of Subclause 13.8, (insert statement); Leveling Rule 2b of Subclause 13.9, (update statement: positioned); Leveling Rule 2b of Subclause 13.10, (update statement: searched).

11. Transaction isolation. All provisions, except DIAGNOSTICS SIZE, of Subclause 14.1, (set transaction statement), in particular: READ ONLY, READ WRITE, and ISOLATION LEVEL, and support for the (set transaction statement> as an (SQL statement) in an explicit or implicit (procedure) with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Leveling Rule 2a of Subclause 14.1. Removal of the Entry SQL limitation on updatable tables, as specified by Leveling Rule 2a of Subclause 7.9, (query specification). Removal of the Entry SQL restriction against inclusion of an (updatability clause) in a (declare cursor) as specified by Leveling Rule 2b of Subclause 13.1, (declare cursor).

12. Get diagnostics. Support for the (get diagnostics statement) specified in Subclause 18.1, as an (SQL statement) in an explicit or implicit (procedure) with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Leveling Rule 2a of Subclause 18.1. If feature #11, "Transaction isolation" is supported, then support for specification of (diagnostics size) in

the (set transaction statement).

13. Grouped operations. Removal of all Entry SQL restrictions for operations involving grouped views and other grouping operations, including: Leveling Rule 2a of Subclause 7.3, (table expression); Leveling Rule 2a of Subclause 7.4, (from clause); Leveling Rule 2c of Subclause 7.9, (query specification); Leveling Rule 2a of Subclause 7.11, (calar subquery) (row subquery) and (table subquery), and Leveling Rule 2b of Subclause 13.5, (select statement: single row).

14. Qualified * in select list. Removal of the Entry SQL restriction for specifying (qualifier>.* in a (select sublist) as specified in Leveling Rule 2b of Subclause 7.9, (query specification).

15. Lowercase identifiers. Removal of the Entry SQL restriction requiring that identifiers not contain any lowercase letters, as specified in Leveling Rule 2b of Subclause 5.2, (token> and (separator); with restrictions identified in the Leveling Rules for Intermediate SQL.

16. PRIMARY KEY enhancement. Removal of the Entry SQL restriction requiring that NOT NULL always be declared with any UNIQUE or PRIMARY KEY, as specified in Leveling Rule 2a of Subclause 11.7, (unique constraint definition).

17. Multiple schemas per user. Support for separation of (schema name) and (authorization identifier) in a (schema definition). The (schema definition) itself need be processed only as required for Entry SQL (i.e. it need not be supported as an SQL statement in an explicit or implicit (procedure)), and a (schema element) need only be as required by Entry SQL. See below feature #31, "Schema definition statement", for more restrictive syntactic requirements. If feature #2, "Basic information schema", is supported, then implementation of Subclause 21.2.4, "SCHEMATA view", in the INFORMATION_SCHEMA. If the Module language binding style is supported, as specified in Subclause 12.1, (module), and Subclause 23.2, "Claims of conformance", then removal of Leveling Rule 2c of Subclause 12.1, (module), that prohibits reference to a (schema name) in a (module) definition.

18. Multiple module support. Removal of the Entry SQL restriction that a (module) be associated with an SQL-agent during its execution, and that an SQL-agent be associated with at most one (module) as specified in Leveling Rule 2a of Subclause 12.1, (module). With removal of this restriction, it will be possible to compile (module) or (embedded SQL host program) separately and rely on the implementation to link them together properly at execution time. To ensure universal portability, applications should adhere to the following self-imposed restrictions: 1) avoid linking modules having cursors with the same (cursor name) 2) avoid linking modules that prepare statements with the same <)SQL statement name) 3) avoid linking modules that allocate descriptors with the same (descriptor name) 4) the scope of an (embedded exception declaration) is a single compilation unit, 5) an (embedded variable name) can be referenced only in the same compilation unit in which it is declared.

19. Referential delete actions. Remove Leveling Rule 2a of Subclause 11.8, (referential constraint definition) thereby providing support for a (referential triggered action) that contains a (delete rule) with restrictions identified in the Leveling Rules for Intermediate SQL. Remove

Leveling Rule 2b of Subclause 11.4, (column definition), thereby allowing an ON DELETE trigger.

20. CAST functions. All provisions of Subclause 6.10, (cast specification), with restrictions identified in the Leveling Rules for Intermediate SQL. The resulting data type of the (cast operand) and of the (cast target) of a (cast specification) shall include all data types required for Entry SQL, as well as any other SQL standard data type whose support is claimed by the implementation. In particular, if feature #5, "Datetime data type", is supported, then (cast specification) for casting DATE, TIME, TIMESTAMP, and INTERVAL to and from character strings. Removal of the Entry SQL restrictions against use of CAST, as specified in Leveling Rule 2a of Subclause 6.10, (cast specification), and Leveling Rule 2d of Subclause 6.11, (value expression).

21. INSERT expressions. Removal of the Entry SQL restriction against specifying a (value expression> in an (insert statement) as specified in the second part of Leveling Rule 2a of Subclause 13.8, (insert statement).

22. Explicit defaults. Removal of the Entry SQL restriction against use of DEFAULT VALUES in an (insert statement) as specified in Leveling Rule 2d of Subclause 13.8, (insert statement). Removal of the Entry SQL restriction against use of DEFAULT in a (row value constructor) as specified in Leveling Rule 2a of Subclause 7.1, (row value constructor). Removal of the Entry SQL restriction against use of datetime and certain USER defaults, as specified in Leveling Rule 2a of Subclause 11.5, (default clause). Removal of the Entry SQL restriction against use of DEFAULT in an (update source) of an (update statement: positioned) or an (update statement: searched) as specified in Leveling Rule 2c of Subclause 13.9, (update statement: positioned).

23. Privilege tables. Support for feature #2, "Basic information schema", defined above, plus inclusion of the following views as defined in Subclause 21.2, "Information Schema": TABLE_PRIVILEGES,COLUMN_PRIVILEGES, and USAGE_PRIVILEGES, all with any restrictions identified in the Leveling Rules for Intermediate SQL.

24. Keyword relaxations. Removal of the Entry SQL restriction against using AS before a (correlation name) as specified in Leveling Rule 2b of Subclause 6.3, (table reference). Removal of the Entry SQL restriction against using the optional keyword TABLE in a GRANT statement, as specified in Leveling Rule 2a of Subclause 11.36, (grant statement). Removal of the Entry SQL restriction for specifying FROM in a FETCH statement, as specified in Leveling Rule 2b of Subclause 13.3, (fetch statement). Removal of the Entry SQL requirement to include the keyword WORK in COMMIT and ROLLBACK statements, as specified in Leveling Rules 2a of Subclause 14.3, (commit statement) and Subclause 14.4, (rollback statement).

14.2 Intermediate SQL. The following FIPS SQL features are required for conformance to Intermediate SQL. They are the focus of current implementation attention and should start becoming widely available in the second or third effective year of this standard.

25. Domain definition. Support for Subclause 11.21, (domain definition), and Subclause 11.27, (drop domain statement); support for VALUE in (general value specification> as specified in

Syntax Rule 5 of Subclause 6.2, (value specification); reference to (domain name> in a (column definition> as specified in Subclause 11.4, (column definition); reference to (domain name> in GRANT and REVOKE statements as specified in Subclause 11.36, (grant statement), and Subclause 11.37, (revoke statement); if feature #2, "Basic information schema", is supported, then implementation of Subclause 21.2.5, "DOMAINS view", in the INFORMATION_SCHEMA; if feature #3, "Basic schema manipulation", is supported, then support for (domain definition) and (drop domain statement> as (SQL statement) in an explicit or implicit (procedure>; if feature #20, "CAST functions", is supported, then support for (domain name) as a (cast target); if feature #33, "Constraint tables", is supported then implementation of Subclause 21.2.6, "DOMAIN_CONSTRAINTS view"; all with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Leveling Rules 2a of Subclause 11.1, (schema definition), and Subclause 11.21, (domain definition), that prohibit domain definitions in Entry SQL; removal of Leveling Rule 2b of Subclause 6.2 that prohibits use of VALUE in a domain check constraint; removal of Leveling Rule 2a of Subclause 11.27, (drop domain statement), that prohibits dropping a domain from a schema;removal of Leveling Rule 2b of Subclause 11.36, (grant statement), that prohibits privilege definition on domains; removal of Leveling Rule 2a of Subclause 11.4, (column definition), that prohibits column data type references to a domain.

26. CASE expression. All provisions of Subclause 6.9, (case expression), with restrictions identified in the Leveling Rules for Intermediate SQL. Removal of all Entry SQL restrictions against use of CASE expressions, including Leveling Rule 2a of Subclause 6.9, (case expression), and Leveling Rule 2c of Subclause 6.11, (value expression).

27. Compound character literals. Removal of the Entry SQL restriction against specifying multiple repetitions of (character representation) strings in a (character string literal) as specified in Leveling Rule 2d of Subclause 5.3, (literal). This feature allows very long character strings to be subdivided into components prior to concatenation into a result string, thereby avoiding problems with line length or line termination in some programming languages.

28. LIKE enhancements. Removal of all Entry SQL restrictions in the LIKE predicate, including: Leveling Rules 2a, 2b, and 2c of Subclause 8.5, (like predicate), thereby allowing the (match value) to be a general (character value expression) instead of just a (column reference) and allowing the (pattern> and (escape character) to be general (character value expression) instead of just simple (value specification).

29. UNIQUE predicate. Support for the UNIQUE predicate, as specified in Subclause 8.9, (unique predicate); with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of the Entry SQL restrictions against use of the (unique predicate) as specified by Leveling Rule 2b of Subclause 8.1 and Leveling Rule 2a of Subclause 8.9.

30. Table operations. Removal of Entry SQL restrictions in Subclause 7.10, (query expression), pertaining to the UNION, EXCEPT and INTERSECT operations, and the CORRESPONDING option, including Leveling Rules 2a, 2b, 2d, and 2e; removal of the Entry SQL restriction requiring that a (query expression) in an (insert statement) not include any table operations, as

specified in part 1 of Leveling Rule 2a of Subclause 13.8, (insert statement); removal of the Entry SQL restriction against using a (derived column list) in a (table reference) as specified in Leveling Rule 2c of Subclause 6.3, (table reference); removal of the Entry SQL restriction requiring that a (query expression> in a ()ubquery) not include any table operations, as specified in Leveling Rule 2b of Subclause 7.11, (subquery); removal of the Entry SQL restriction requiring that a (query expression> in a (view definition> not include any table operations, as specified in Leveling Rule 2a of Subclause 11.19, (view definition); all with any other restrictions identified in the Leveling Rules for Intermediate SQL.

31. Schema definition statement. Support for (schema definition) Subclause 11.1, as an (SQL statement) in an explicit or implicit(procedure) by removal of Leveling Rule 2a of Subclause 12.5, (SQL procedure statement), with any restrictions identified in the Leveling Rules for Intermediate SQL. A (schema element) shall be an element required byEntry SQL, or an element defined in another feature whose support is claimed by the implementation. Support for feature #17, "Multiple schemas per user", defined above and removal of the Entry SQL restriction that prohibits definition of a (schema name) in a (schema definition) as specified in Leveling Rule 2b of Subclause 11.1. A (schema definition) may contain any circular references that are permitted for Intermediate SQL; in particular, (referential constraint definition) in two different (table definition) that reference columns in the other table.

32. User authorization. All provisions of Subclause 16.4, (set session authorization identifier statement), with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Leveling Rule 2a of Subclause 16.4. Support for CURRENT_USER, SESSION_USER, and SYSTEM_USER in (general value specification) by removal of Leveling Rule 2c of Subclause 6.2, (value specification).

33. Constraint tables. Support for feature #2, "Basic information schema", defined above, plus inclusion of the following views as defined in Subclause 21.2, "Information Schema": TABLE_CONSTRAINTS,REFERENTIAL_CONSTRAINTS, and CHECK_CONSTRAINTS, all with any restrictions identified in the Leveling Rules for Intermediate SQL.

34. Usage tables. Support for feature #2, "Basic information schema", defined above, plus inclusion of the following views as defined in Subclause 21.2, "Information Schema": KEY_COLUMN_USAGE, VIEW_TABLE_USAGE, VIEW_COLUMN_USAGE, CONSTRAINT_TABLE_USAGE, CONSTRAINT_COLUMN_USAGE, and COLUMN_DOMAIN_USAGE, all with any restrictions identified in the Leveling Rules for Intermediate SQL.

35. Intermediate information schema. All provisions of Subclause 21.2, "Information schema", with restrictions identified in the Leveling Rules for Intermediate SQL. This feature includes feature #2, "Basic information schema", feature #23, "Privilege tables", feature #33, "Constraint tables", and feature #34, "Usage tables", defined above, the SCHEMATA view from feature #17, the DOMAINS and DOMAIN_CONSTRAINTS views from feature #25, and the CHARACTER_SETS view from feature #45, as well as all remaining tables required for Intermediate SQL, including: Subclause 21.2.2,

"INFORMATION_SCHEMA_CATALOG_NAME base table", Subclause 21.2.17, "ASSERTIONS view", and Subclause 21.2.26, "SQL_LANGUAGES view". Removal of all Leveling Rules 2a in Subclause 21.2, "Information Schema". In Entry SQL or Intermediate SQL, some of these tables may have empty or trivial contents, but an implementation supporting this feature is required to process statements that properly reference any of these tables. These tables are all self-describing in the sense that they appear in the Definition Schema as PUBLIC tables and thus are visible in the INFORMATION_SCHEMA for every user.

36. Subprogram support. Removal of the Entry SQL restriction that prohibits full use of local variable name scoping in subprograms, as specified in Leveling Rule 2b of Subclause 19.1, (embedded SQL host program). With removal of this restriction, compilation units may include subprograms with SQL statements that reference local variables defined in the subprogram even if some other identically named variable, having different scope, happens to be defined in an SQL (host variable definition) in the main program or in a different subprogram of the same compilation unit. This feature applies only to the Embedded SQL language interface specified in Clause 19, "Embedded SQL"; it does not apply to any Module or Direct Invocation interfaces.

37. Intermediate SQL Flagging. Support for both "Entry SQL Flagging" and "Intermediate SQL Flagging" with the "Syntax Only" extent of checking option as specified for conforming Intermediate SQL implementations in Subclause 23.4, "Flagger requirements", of ANSI X3.135-1992. This facility would allow an application to distinguish between vendor extensions beyond Entry SQL that are supported in Intermediate SQL and those that are beyond Intermediate SQL or are non-standard.

38. Schema manipulation. Support for feature #3, "Basic schema manipulation". In addition, support for the following schema definition and schema manipulation statements as (SQL statement) in an explicit or implicit (procedure): Subclause 11.2, (drop schema statement); Subclauses 11.10 through 11.17, (alter table statement); all with any restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Leveling Rules 2a in Subclause 11.2 and Subclauses 11.10 through 11.17. If Direct invocation and processing of SQL language is supported, as specified in Subclause 23.2, "Claims of conformance", then removal of Leveling Rule 2a of Subclause 20.1, (direct SQL statement).

39. Long identifiers. Support for (regular identifier) or (delimiter identifier body) that have lengths of up to 128 characters; that is, remove the restrictions of Leveling Rule 2a of Subclause 5.2, (token> and (separator).

40. Full outer join. Support for feature #4, "Joined table". In addition, support for FULL OUTER JOIN by removal of Leveling Rule 2a from Subclause 7.5, (joined table).

41. Time zone specification. Support for feature #5, "DATETIME data types". In addition, full implementation of time zones and time zone management, including: support for (time zone interval) specified in Subclause 5.3, (literal), (time zone field) specified in Subclause 6.6, (numeric value function), (time zone) and (time zone specifier) specified in Subclause 6.14, (datetime value expression), and (set local time zone statement) specified in Subclause 16.5, (set local time zone

statement). Removal of the Entry SQL restriction on (timezone field) in an (extract expression) as specified in Leveling Rule 2b of Subclause 6.6, (numeric value function). Removal of Leveling Rule 2a in Subclause 16.5, (set local time zone statement).

42. National character. Support for NATIONAL CHARACTER, and its syntactic shorthands NATIONAL CHAR and NCHAR, as defined in Subclause 6.1, (data type); support for (national character string literal) as defined in Subclause 5.3, (literal); removal of the Entry SQL restriction requiring that a (delimiter token) shall not be a (national character string literal) as specified in Leveling Rule 2a of Subclause 5.3, (literal). Removal of (national character string type) from Leveling Rule 2c of Subclause 6.1, (data type). If feature #6, "VARCHAR data type" is supported, then support for NATIONAL CHARACTER VARYING, and its syntactic shorthands NATIONAL CHAR VARYING and NCHAR VARYING, as specified in Subclause 6.1, (data type).

43. Scrolled cursors. Support for SCROLL in a cursor declaration and for (fetch orientation) in a FETCH statement, by removal of the Entry SQL restrictions against declaration and use of scrolled cursors as specified in Leveling Rule 2a of Subclause 13.1, (declare cursor), and Leveling Rule 2a of Subclause 13.3, (fetch statement).

44. Intermediate set function. Removal of Entry SQL restrictions against certain set function operations, including: removal of Leveling Rule 2a concerning COUNT ALL, Leveling Rule 2b concerning (column reference), Leveling Rule 2c concerning (value expression) and Leveling Rule 2d concerning column references, all of Subclause 6.5, (set function specification); removal of Leveling Rule 2a concerning reference to a column generally containing a set function, as specified in Subclause 7.6, (where clause).

45. Character set definition. Support for Subclause 11.28, (character set definition), and Subclause 11.29, (drop character set statement), as (SQL statement) in an explicit or implicit (procedure) Support for granting and revoking USAGE privileges on any defined character sets, as specified in Subclause 10.3, (privileges), Subclause 11.36, (grant statement), and Subclause 11.37, (revoke statement). If feature #2, "Basic information schema", is supported, then implementation of Subclause 21.2.18, "CHARACTER_SETS view" in the INFORMATION_SCHEMA. Removal of Leveling Rules 2a in Subclause 11.28 and Subclause 11.29. Removal of the Entry SQL restriction against using a (character set specification) in Embedded SQL, as specified in the Leveling Rules of each (embedded SQL host program) Support for all other references to (character set specification) including removal of Leveling Rule 2a of Subclause 10.4, (character set specification), Leveling Rules 2c and 2d of Subclause 11.1, (schema definition), and Leveling Rule 2a of Subclause 12.2, (module name clause).

46. Named character sets. Support for the named character sets: SQL_CHARACTER, ASCII_GRAPHIC, LATIN1, ASCII_FULL, and SQL_TEXT. If feature #2, "Basic information schema" is supported, then implementation of the CHARACTER_SETS view in the INFORMATION_SCHEMA. Support for SQL_TEXT is required by ANSI SQL Syntax Rule 3 of Subclause 10.4, (character set specification); the other character sets are defined in Section 16.7 of this standard. Removal of Leveling Rule 2e of Subclause 5.3, (literal). Removal of Leveling

Rule 2b of Subclause 5.4, "Names and identifiers". Removal of CHARACTER SET from Leveling Rule 2c of Subclause 6.1, (data type). Removal of the Entry SQL restriction against referencing these character sets in Embedded SQL, as specified by Leveling Rule 2a of Subclause 19.1, (embedded SQL host program), and by the Leveling Rule of each (embedded SQL host program) that prohibits use of a (character set specification) Support for all other references to these named character sets in any (character set specification) in any SQL statement supported in Entry SQL, and in any other SQL statement whose support is claimed by the implementation.

47. Scalar subquery values. Support for the use of a (scalar subquery) in any (value expression) by removal of Leveling Rule 2e of Subclause 6.11, (value expression).

48. Expanded null predicate. Support for referencing values other than a (column reference> in a (null predicate) by removal of Leveling Rule 2a of Subclause 8.6, (null predicate). This includes support for testing any Entry SQL (value expression) or any supported Intermediate SQL (value expression) to see if it is null.

49. Constraint management. Support for user-defined constraint names, including use of an explicit (constraint name definition> in any table constraint, by removal of: Leveling Rule 2a of Subclause 10.6, (constraint name definition), Leveling Rule 2c of Subclause 11.4, (column definition), and Leveling Rule 2a of Subclause 11.6, (table constraint definition). Support for feature #33, "Constraint tables". If feature #25, "Domain definition", is supported, then removal of all restrictions against explicit (constraint name definition) in Subclause 11.21, (domain definition). If feature #12, "Get diagnostics", is supported, then support for reference to user-defined, explicit (constraint name) in the (get diagnostics statement).

50. Documentation schema. Implementation of the documentation schema tables, SQL_FEATURES and SQL_SIZING, as specified in Section 15 of this standard.

14.3 Full SQL. The following FIPS SQL features are required for conformance to Full SQL. They will start becoming available in some products during the first through third effective years of FIPS 127-2. Some care should be taken before specifying Full SQL as a mandatory requirement in an SQL procurement because it is not yet certain if these features will be completely or competitively available in all operating environments.

51. BIT data type. Support for BIT and BIT VARYING data types, as defined in Subclause 6.1, (data type); support for (bit string literal) and (hex string literal) as defined in Subclause 5.3, (literal); support for the following bit string operations: (length expression) defined in Subclause 6.6, (numeric value function), (bit substring function) defined in Subclause 6.7, (string value function), (concatenation) and (bit value expression) defined in Subclause 6.13, (string value expression); support for comparison of fixed and variable length bit strings as defined in Subclause 8.2, (comparison predicate); support for BIT VARYING in any (embedded SQL host program) supported by the implementation. If feature #20, "CAST functions", is supported, then (cast specification) for casting bit strings to and from character strings. Removal of Leveling Rule 1b of Subclause 5.3, (literal). Removal of Leveling Rule 1b of Subclause 6.1, (data type). Removal of the Intermediate SQL requirement that a (numeric value function) not be a (length expression) as specified in Leveling Rule 1b of Subclause 6.6, (numeric value function). Removal

of the Intermediate SQL restriction against using a (bit value function) as specified in Leveling Rule 1d of Subclause 6.7, (string value function). Removal of Leveling Rule 1b of Subclause 6.13, (string value expression), that prohibits use of a (bit value expression) in Intermediate SQL. Removal of Leveling Rule 1a of Subclause 12.3, (procedure), that prohibits specification of BIT or BIT VARYING in an Intermediate SQL (procedure) Removal of Leveling Rule 1a of Subclause 19.3, (embedded SQL Ada program), Leveling Rule 1a of Subclause 19.4, (embedded SQL C program), Leveling Rule 1a of Subclause 19.5, (embedded SQL COBOL program), Leveling Rule 1a of Subclause 19.6, (embedded SQL Fortran program), Leveling Rule 1a of Subclause 19.8, (embedded SQL Pascal program), and Leveling Rule 1a of Subclause 19.9, (embedded SQL PL/I program).

52. Assertion constraints. Support for (assertion definition) and (drop assertion statement) as (SQL statement) in an explicit or implicit (procedure) as specified in Subclauses 11.34 and 11.35, respectively. If feature #2, "Basic information schema" is supported, then implementation of Subclause 21.2.17, "ASSERTIONS view", in the INFORMATION_SCHEMA. Removal of Leveling Rules 1a in Subclause 11.1, Subclause 11.34, and Subclause 11.35.

53. Temporary tables. Support for GLOBAL TEMPORARY or LOCAL TEMPORARY table definitions and ON COMMIT DELETE ROWS or ON COMMIT PRESERVE ROWS options, as specified in Subclause 11.3, (table definition); all provisions of Subclause 13.11, (temporary table declaration); all with any other restrictions identified in the Leveling Rules for Intermediate SQL. Removal of Leveling Rules 1a from Subclause 11.3, Subclause 12.1, and Subclause13.11.

54. Full dynamic SQL. All provisions of Clause 17, "Dynamic SQL", including: host variables for descriptor names, statement names, and cursor names; DEALLOCATE PREPARE; DESCRIBE INPUT; dynamic single row select; and preparable positioned dynamic update/delete. Removal of the Intermediate SQL restriction against reference to system-supplied names, as specified in Leveling Rule 1a of Subclause 5.4, "Names and identifiers". Removal of Leveling Rules 1a of Subclauses 17.2, 17.3, 17.4, and 17.9 that restrict (occurrences> and (descriptor name) to be (literal). Support for Subclause 17.7, (deallocate prepared statement), Subclause 17.8, (describe statement), Subclause 17.13, (allocate cursor statement), Subclause 17.19, (preparable dynamic delete statement: positioned), and Subclause 17.20, (preparable dynamic update statement: positioned), by removal Intermediate SQL Leveling Rule 1a from each of these clauses. Support for a (result using clause) in an EXECUTE statement, by removal of Leveling Rule 1a of Subclause 17.10, (execute statement).

55. Full DATETIME. Support for feature #5, "DATETIME data type", and feature #41, "Time zone specification". In addition, removal of the Intermediate SQL restrictions against specification of precision in TIME and TIMESTAMP data types, as specified in Leveling Rule 1a of Subclause 6.1, (data type), Leveling Rule 1a of Subclause 5.3, (literal), and Leveling Rule 1a of Subclause 6.8, (datetime value function).

56. Full value expressions. Support for references to (value expression) in all places where they are restricted in Entry SQL or Intermediate SQL, including: removal of the Intermediate SQL restriction against using a (value expression) in a (general set function) with DISTINCT, as

specified in Leveling Rule 1a of Subclause 6.5, (set function specification); removal of the Intermediate SQL restriction against using a (value expression) in an (in predicate) as specified in Leveling Rule 1a of Subclause 8.4, (in predicate).

57. Truth value tests. Support for (truth value) tests of TRUE, FALSE, or UNKNOWN, or their negations, applied to a (boolean primary) as specified in Subclause 8.12, (search condition). Removal of Leveling Rule 1a of Subclause 8.12.

58. Full character functions. Removal of the Intermediate SQL restriction against use of a POSITION expression, as specified in Leveling Rule 1a of Subclause 6.6, (numeric value function). Removal of the Intermediate SQL restrictions against use of UPPER and LOWER functions, as specified in Leveling Rule 1a of Subclause 6.7, (string value function).

59. Derived tables in FROM. Removal of the Intermediate SQL Leveling Rule 1a of Subclause 6.3, (table reference), that prohibits a (table reference) from being a (derived table). The effect is that a (derived table) possibly with a (derived column list) may be specified in a FROM clause.

60. Trailing underscore. Removal of the Intermediate SQL restriction against using an (underscore) as the last character of an identifier, as specified in Leveling Rule 1a of Subclause 5.2, (token) and (separator).

61. Indicator data types. Removal of the Intermediate SQL restrictions on the data types of indicator parameters and variables, as specified in Leveling Rule 1a of Subclause 6.2, (value specification) and (target specification).

62. Referential name order. Removal of the Intermediate SQL restriction on the order of column names in a referential constraint definition, as specified in Leveling Rule 1c of Subclause 11.8, (referential constraint definition).

63. Full SQL Flagging. Support for "Entry SQL Flagging", "Intermediate SQL Flagging", and "Full SQL Flagging", each with the "Syntax Only" extent of checking option as defined in Subclause 4.33 of ANSI X3.135-1992. This facility would allow an application to distinguish among vendor extensions beyond Entry SQL that are supported in Intermediate SQL, those beyond Intermediate SQL that are supported in Full SQL, and those that are non- standard. This feature does not include support for the "Catalog Lookup" option of the SQL Flagger (see feature #81).

64. Row and table constructors. All provisions of (row value constructor) and (table value constructor) as specified in Subclause 7.1, (row value constructor), and Subclause 7.2, (table value constructor), thereby providing support for multiple column row and table constructors. Removal of all Intermediate SQL Leveling Rules in these two clauses, allowing full use of (row value constructor) and (table value constructor) in Clause 8, "Predicates", and in Subclause 13.8, (insert statement).

65. Catalog name qualifiers. Removal of the Intermediate SQL restriction against reference to

catalog names, as specified in part of Leveling Rule 1b of Subclause 5.4, "Names and identifiers".

66. Simple tables. Support for (simple table) and removal of the Intermediate SQL restriction against simple or explicit table references in a (query expression) as specified in Leveling Rules 1a and 1b of Subclause 7.10, (query expression).

67. Subqueries in CHECK. Removal of the Intermediate SQL Leveling Rule 1a of Subclause 11.9, (check constraint definition), that prohibits specification of a (subquery) in a CHECK constraint, but retaining any other restrictions on (subquery) as required for Intermediate SQL. Support for granting and revoking REFERENCES privileges on all tables, including views, as specified in Subclause 10.3, (privileges), so that the (subquery) can reference base tables and views external to the table containing the check constraint. Removal of Intermediate SQL Leveling Rule 1b of Subclause 11.9, (check constraint definition), that allows implicit assumption of REFERENCES privileges on tables in Intermediate SQL.

68. Union and Cross join. Support for feature #4, "Joined table". In addition, support for UNION JOIN and CROSS JOIN by removal of Leveling Rule 1a and Leveling Rule 1b of Subclause 7.5, (joined table).

69. Collation and translation. Support for Subclause 11.30, (collation definition), Subclause 11.31, (drop collation statement), Subclause 11.32, (translation definition), and Subclause 11.33, (drop translation statement), as (SQL statement) in an explicit or implicit (procedure) Removal of Leveling Rules 1a in Subclauses 11.30 through 11.33. Support for granting and revoking USAGE privileges on any defined collations or translations, as specified in Subclause 10.3, (privileges), by removal of Leveling Rule 1a in Subclause 11.36, (grant statement). If feature #2, "Basic information schema" is supported, then implementation of Subclause 21.2.19, "COLLATIONS view", and Subclause 21.2.20, "TRANSLATIONS view", in the INFORMATION_SCHEMA. Removal of the Intermediate SQL restriction against reference to collation, translation, or conversion names, as specified in Leveling Rule 1b of Subclause 5.4, "Names and identifiers", in Leveling Rule 1a of Subclause 6.13, (string value expression), and in Leveling Rules 1b and 1c of Subclause 11.1, (schema definition). Removal of the Intermediate SQL restriction against use of the (collate clause) as specified in Leveling Rule 1a of Subclause 10.5, (collate clause), Leveling Rule 1a of Subclause 11.4, (column definition), Leveling Rule 1a of Subclause 7.7, (group by clause), Leveling Rule 1a of Subclause 11.21, (domain definition), and Leveling Rule 1a of Subclause 11.28, (character set definition). Removal of the Intermediate SQL restrictions against use of character translations or form-of-use conversions, as specified in Leveling Rules 1b and 1c of Subclause 6.7, (string value function).

70. Referential update actions. Support for a (referential triggered action) that contains an (update rule) as defined in Subclause 11.8, (referential constraint definition). Removal of Leveling Rule 1b of Subclause 11.8, (referential constraint definition), that prohibits specification of an (update rule) in a (referential triggered action) thereby allowing an ON UPDATE trigger.

71. ALTER domain. Support for (alter domain statement) (set domain default clause) (drop domain default clause) (add domain constraint definition) and (drop domain constraint definition)

all specified in Subclauses 11.22 through 11.26, as (SQL statement) in an explicit or implicit (procedure) Remove all Intermediate SQL Leveling Rules in Subclauses 11.22 through 11.26.

72. Deferrable constraints. Support for the (set constraints mode statement) specified in Subclause 14.2, as an (SQL statement> in an explicit or implicit (procedure) Removal of Leveling Rule 1a of Subclause 14.2. Removal of Intermediate SQL Leveling Rule 1a of Subclause 10.6, (constraint name definition) and (constraint attributes), that prohibits user specification of DEFERRABLE constraints and (constraint check time) as DEFERRED or IMMEDIATE.

73. INSERT column privileges. Removal of the Intermediate SQL Leveling Rule 1a of Subclause 10.3, (privileges), that prohibits the granting of INSERT privileges on individual columns.

74. Referential MATCH types. Support for MATCH FULL and MATCH PARTIAL, as defined in Subclause 11.8, (referential constraint definition), by removal of Intermediate SQL Leveling Rule 1a. Support for the MATCH predicate, as specified in Subclause 8.10, (match predicate), by removal of Intermediate SQL Leveling Rules 1a of Subclause 8.1, (predicate), and Subclause 8.10, (match predicate).

75. View CHECK enhancements. Support for CASCADED and LOCAL options in the WITH CHECK OPTION clause of a (view definition) Removal of the Intermediate SQL restrictions against explicit declaration of these options, as specified in Leveling Rule 1a of Subclause 11.19, (view definition).

76. Session management. Support for "Session management", as specified in Subclause 16.1, (set catalog statement) Subclause 16.2, (set schema statement), and Subclause 16.3, (set names statement). Removal of all Intermediate SQL Leveling Rules in Subclauses 16.1, 16.2, and 16.3.

77. Connection management. Support for all provisions of Clause 15, "Connection management", including: CONNECT, SET CONNECTION, and DISCONNECT. Those (simple value specification) that are valid (connection target) and (user name) are implementation-defined, so long as Syntax Rule 1 of Subclause 15.1, (connect statement), is satisfied. The communication protocols used to implement the connection management statements are implementation-defined. Removal of all Intermediate SQL Leveling Rules in Subclauses 15.1, 15.2, and 15.3. Removal of the Intermediate SQL restriction against reference to (connection name) as specified in Leveling Rule 1b of Subclause 5.4, "Names and identifiers".

78. Self-referencing operations. Removal of the Intermediate SQL restrictions against self-referencing DELETE, INSERT, and UPDATE statements, as specified in Leveling Rules 1a of Subclause 13.7, (delete statement: searched), Subclause 13.8, (insert statement), and Subclause 13.10, (update statement: searched).

79. Insensitive cursors. Support for the INSENSITIVE option on a cursor declaration, by removal of the Intermediate SQL Leveling Rule 1a of Subclause 13.1, (declare cursor). If feature #54, "Full dynamic SQL", is supported, then removal of Leveling Rule 1a in Subclause 17.12,

(dynamic declare cursor).

80. Full set function. Support for feature #44, "Intermediate set function". In addition, removal of the Intermediate SQL restrictions against use of DISTINCT in a (general set function) as specified in Leveling Rule 1a of Subclause 6.5, (set function specification), and removal of the Intermediate SQL restriction against multiple use of DISTINCT in a (query specification) as specified in Leveling Rule 1a of Subclause 7.9, (query specification).

81. Catalog flagging. Support for both the "Syntax Only" and "Catalog Lookup" extent of checking options of the SQL Flagger feature, as defined in Subclause 4.34 of ANSI X3.135-1992. This facility would allow the SQL Flagger to catch a syntactic extension that violates a Syntax Rule dependent upon information stored in the INFORMATION_SCHEMA.

82. Local table references. Support for qualified local table references of the form MODULE.T, by removal of the Intermediate SQL restriction against reference to a (qualified local table name) as specified in Leveling Rule 1b of Subclause 5.4, "Names and identifiers".

83. Full cursor update. Support for the updatability of SCROLL or ORDER BY cursors, by removal of Leveling Rule 1b of Subclause 13.1, (declare cursor) and Leveling Rule 1a of Subclause 13.9, (update statement: positioned). If feature #54, "Full dynamic SQL", is supported, then removal of Leveling Rule 1b in Subclause 17.12, (dynamic declare cursor).

14.4 Integration with RDA. The following FIPS SQL optional features require conformance to ISO/IEC 9579, the International Standard for Remote Database Access (RDA), with the implementation agreements specified in FIPS 146 (GOSIP) from the NIST OSI Implementors Workshop. SQL implementations that also conform to the RDA portion of GOSIP should start becoming available near the end of the first or at the beginning of the second effective year of FIPS 127-2.

84. RDA/SQL-Client. Conformance to the Remote Database Access (RDA) component of FIPS PUB 146, "Government Open Systems Interconnection Profile (GOSIP)", by satisfying all of the requirements for the "Immediate Execution" profile as an "RDA Client" and abiding by the NIST OSI Implementor's Workshop implementation agreements. The Immediate Execution profile requires the following RDA functional units: Dialogue Initialization, Dialogue Termination, Transaction Management (Basic Application Context), Resource Handling, and Immediate Execution DBL. Conformance to FIPS PUB 127-2, "FIPS SQL", at the Entry SQL level or above, and support for FIPS feature #1, "Dynamic SQL", and FIPS feature #2, "Basic information schema". Support for the FIPS feature #77, "Connection management", defined above, with the following additional requirements: the CONNECT statement triggers Dialogue Initialization (R-Initialize Service) and Resource Handling (R-Open Service); the DISCONNECT statement triggers Resource Handling (R-Close Service) and Dialogue Termination (R-Terminate Service); the SET CONNECTION statement re-establishes active Resource Handling and, if necessary may trigger Dialogue Termination (R-Close Service) to make a current SQL- connection dormant; Transaction Management supports SQL-COMMIT and SQL-ROLLBACK, and satisfies the timing and semantics of an SQL-Transaction; other SQL-Statements, in the host language binding style supported by the SQL-implementation, are mapped to Immediate Execution DBL

(R-ExecuteDBL Service) protocols.

85. RDA/SQL-Server. Conformance to Remote Database Access (RDA) component of FIPS PUB 146, "Government Open Systems Interconnection Profile (GOSIP)", by satisfying all of the requirements for the "Immediate Execution" profile as an "RDA Server" and abiding by the NIST OSI Implementor's Workshop implementation agreements. The Immediate Execution profile includes: Dialogue Initialization, Dialogue Termination, Transaction Management (Basic Application Context), Resource Handling, and Immediate Execution DBL. Conformance to FIPS PUB 127-2, "FIPS SQL", at the Entry SQL level or above, and support for FIPS feature #1, "Dynamic SQL", and FIPS feature #2, "Basic information schema".

86. RDA Stored Execution. Conformance to feature #84, "RDA/SQL-Client", or feature #85, "RDA/SQL-Server", defined above and, in addition, support for the RDA Stored Execution Functional Unit as specified in ISO/IEC 9579- 2 (RDA SQL Specialization), and with implementor agreements specified by the NIST OSI Implementor's Workshop.

87. RDA Cancel. Conformance to feature #84, "RDA/SQL-Client", or feature #85, "RDA/SQL-Server", defined above and, in addition, support for the RDA Cancel Functional Unit as specified in ISO/IEC 9579-2 (RDA SQL Specialization), and with implementor agreements specified by the NIST OSI Implementor's Workshop.

88. RDA Status. Conformance to feature #84, "RDA/SQL-Client", or feature #85, "RDA/SQL-Server", defined above and, in addition, support for the RDA Status Functional Unit as specified in ISO/IEC 9579-2 (RDA SQL Specialization), and with implementor agreements specified by the NIST OSI Implementor's Workshop.

89. RDA TP Application Context. Conformance to feature #84, "RDA/SQL- Client", or feature #85, "RDA/SQL-Server", defined above and, in addition, support for the RDA SQL TP Application Context as specified in ISO/IEC 9579-2 (RDA SQL Specialization), and dependent upon ISO/IEC DIS 10026 (Distributed Transaction Processing), and with Distributed Transaction Processing implementor agreements specified by the NIST OSIImplementor's Workshop.

15. FIPS documentation schema. For conformance to Intermediate SQL or to Full SQL, FIPS SQL requires that the implementation provide a special schema, the FIPS_DOCUMENTATION schema, as a system-owned schema in every catalog supported by that implementation (see Section 10.f). The FIPS_DOCUMENTATION schema has, effectively, the following schema definition:

```
CREATE SCHEMA FIPS_DOCUMENTATION
    AUTHORIZATION "_SYSTEM"
    DEFAULT CHARACTER SET SQL_CHARACTER

CREATE TABLE SQL_FEATURES
    (
    FEATURE_ID   SMALLINT   PRIMARY KEY   CHECK
     (FEATURE_ID > 0),
```

```
        FEATURE_NAME   CHARACTER (50)   NOT NULL,
        CLASSIFICATION   CHARACTER (12)   NOT NULL  CHECK
         (CLASSIFICATION IN
        ('TRANSITIONAL', 'INTERMEDIATE', 'FULL', 'RDA')),
        IS_SUPPORTED  CHARACTER (3) NOT NULL CHECK
(IS_SUPPORTED IN ('YES',
        'NO')),
        IS_VERIFIED   CHARACTER (3)   NOT NULL   CHECK
         (IS_VERIFIED IN ('YES',
        'NO')),
        FEATURE_COMMENTS   VARCHAR (500)   CHARACTER SET
         SQL_TEXT,
             CHECK (IS_SUPPORTED='YES' OR IS_VERIFIED='NO')
        )

    CREATE TABLE SQL_SIZING
        (
        SIZING_ID   SMALLINT   PRIMARY KEY   CHECK (SIZING_ID >
         0),
        DESCRIPTION   CHARACTER (50)   NOT NULL,
        ENTRY_VALUE   INTEGER,
        INTERMEDIATE_VALUE   INTEGER,
        VALUE_SUPPORTED   INTEGER,
        SIZING_COMMENTS   VARCHAR (500)   CHARACTER SET
         SQL_TEXT
        )

    GRANT SELECT, REFERENCES ON SQL_FEATURES TO PUBLIC
         WITH GRANT OPTION
    GRANT SELECT, REFERENCES ON SQL_SIZING TO PUBLIC WITH
         GRANT OPTION
```

15.1 SQL_Features table. The SQL_FEATURES table shall consist of exactly one row for each FIPS SQL feature defined in Section 14 of this standard. The FEATURE_ID and FEATURE_NAME columns identify the feature by the integer and name assigned to it in Section 14. The CLASSIFICATION column identifies the FIPS conformance level in which the feature first becomes required; all features in Subsection 14.1 are classified as TRANSITIONAL, all features in Subsection 14.2 are classified as INTERMEDIATE, all features in Subsection 14.3 are classified as FULL, and all features in Subsection 14.4 are classified as RDA. The IS_SUPPORTED column is 'YES' if an implementation fully supports that feature when data in the identified catalog is accessed through that implementation, and is 'NO' if the implementation does not fully support the feature when accessing that catalog. If full support for the feature has been verified by testing with the NIST SQL Test Suite, then the IS_VERIFIED column is 'YES'; otherwise, the IS_VERIFIED column is 'NO'. If the IS_VERIFIED column is 'YES', then the vendor of the implementation shall have passed, either by self testing or by witnessed testing, all tests in the then current version of the NIST SQL Test Suite that apply to that feature. The FEATURE_COMMENTS column is intended for any vendor comments pertinent to the identified FIPS SQL feature.

15.2 SQL_Sizing table. The SQL_SIZING table shall consist of exactly one row for each FIPS SQL database construct defined in Section 16.6 of this standard. The SIZING_ID and DESCRIPTION columns identify the database construct by the integer and description assigned to it in Section 16.6.

The ENTRY_VALUE column is equal to the default Entry SQL value defined for that construct by FIPS SQL in Section 16.6, with "*" converted to a Null value. The INTERMEDIATE_VALUE column is equal to the default Intermediate SQL value defined for that construct by FIPS SQL in Section 16.6, with "*" converted to a Null value. The VALUE_SUPPORTED column indicates a value for the construct that is supported by an implementation when data in the identified catalog is accessed through that implementation; if this value is Null, then there is no explicit restriction on the size of that construct. A user must be able to depend upon these values when executing SQL-statements against data in the catalog. If a given catalog spans multiple SQL-Server implementations, then the VALUE_SUPPORTED shall be valid in all of them. It is important to recognize that FIPS sizing defaults are not requirements for conformance to FIPS SQL; instead, they identify a default value that is assumed to be specified if a Federal SQL procurement is silent on that topic. For this reason, the VALUE_SUPPORTED may sometimes be less than the FIPS default for the ENTRY_VALUE or the INTERMEDIATE_VALUE, even for a FIPS SQL conforming implementation. The SIZING_COMMENTS column is intended for any vendor comments pertinent to the identified FIPS SQL database construct.

16. Special Procurement Considerations. FIPS SQL includes various alternatives for interfacing to programming languages, allows the additional specification of optional FIPS SQL features, and does not specify any minimum requirements for the size or number of occurrences of database constructs. Any invocation of this standard in a procurement should indicate the programming languages to which it interfaces, whether Modules, Embedded SQL, or both are required for each language, which base level of FIPS SQL conformance is a mandatory requirement, which optional features are desirable, and what the sizing and occurrence requirements are. Any use of this standard in a broader database management system (DBMS) procurement should be accompanied with functional requirements for other DBMS components and facilities.

16.1 Procurement wording. References to this standard in a procurement should be accompanied with appropriate solicitation wording. Information on Acquisition wording is in Section 11.2 of this standard, and information on Validation wording is in Section 11.4.

16.2 Programming language interfaces. References to this standard in a procurement should indicate which FIPS programming languages (e.g. Ada, C, COBOL, FORTRAN, MUMPS, or Pascal) are to be supported for language interface. Failure to make this indication means that support for any one of these languages satisfies the FIPS SQL requirement.

16.3 Style of language interface. References to this standard in a procurement should indicate, for each programming language identified above, whether the language interface is to support Modules, Embedded SQL, or both. Failure to make this indication means that support for any one interface style satisfies the FIPS SQL requirement.

16.4 Optional Features. References to this standard in a procurement should indicate which FIPS SQL conformance level is a mandatory requirement. Valid base level conformance alternatives are: Entry SQL, Transitional SQL, Intermediate SQL, or Full SQL. In addition, procurements may specify desirable features beyond that level (see Section 14). Implementations that support the identified desirable features may be rated higher in the procurement evaluation process. Failure to specify a

mandatory base level of conformance means that only Entry SQL is required.

Under certain circumstances, one or more FIPS SQL features above the base level of conformance may be specified as mandatory requirements in a Federal procurement. Usually such features will only be specified as desirable. Procurements that seek effective competition from a number of different vendors should be very careful in distinguishing between mandatory and desirable features.

Depending upon agency requirements and vendor cooperation, the NIST SQL Test Suite may identify and test various SQL "profiles" for specific purposes. For example, an "interoperability profile" might include some features from Transitional SQL as well as Connection management (feature #77) from Full SQL and Client/Server capabilities (features #84 and #85) from RDA, or a "read only profile" might include only selected data manipulation statements without any schema definition or schema manipulation. See the most recent version of the NIST SQL Test Suite for possible specification of such profiles.

As always, all syntactic extensions beyond Entry SQL, Intermediate SQL, or Full SQL shall be appropriately flagged by the SQL Flagger.

16.5 Interactive Direct SQL. References to this standard in a procurement should indicate whether or not "Interactive Direct SQL" is required. If it is required, then in order to satisfy the requirement, an implementation shall provide interactive access to the database, using any **)** as specified in Clause 20 of ANSI X3.135-1992, and subject to any leveling or FIPS SQL feature requirements specified in Section 16.4 above. Failure to indicate an explicit requirement for "Interactive Direct SQL" in a procurement means that this interface alternative is not required.

Additional FIPS requirements for Interactive Direct SQL are as follows: if a statement raises an exception condition, then the system shall display a message indicating that the statement failed, giving a textual description of the failure; if a statement raises a completion condition that is a "warning" or "no data", then the system shall display a message indicating that the statement completed, giving a textual description of the "warning" or "no data"; an implementation shall provide some implementation-defined symbol for displaying null values and, for character string values, this symbol must be distinguishable from a value of all **)**.

16.6 Sizing for database constructs. References to this standard in a procurement should indicate minimum requirements for the precision, size, or number of occurrences of database constructs. Failure to make this indication means that the Entry Values detailed below are by default the minimum requirements for Entry SQL or Transitional SQL procurements and the Intermediate Values detailed below are by default the minimum requirements for Intermediate SQL or Full SQL procurements.

Sizing Id	Description	Entry Value	Interm. Value
1.	Length of an identifier	18	128
2	CHARACTER max length	240	1000
3.	CHARACTER VARYING max length	254	1000
4.	BIT max length in bits	*	8000

5. BIT VARYING max length in bits	*	8000
6. NATIONAL CHARACTER max length	*	500
7. NATIONAL CHAR VARYING max length	*	500
8. NUMERIC decimal precision	15	15
9. DECIMAL decimal precision	15	15
10. INTEGER decimal precision	9	*
11. INTEGER binary precision	*	31
12. SMALLINT decimal precision	4	*
13. SMALLINT binary precision	*	15
14. FLOAT binary mantissa precision	20	47
15. FLOAT binary exponent precision	*	9
16. REAL binary mantissa precision	20	23
17. REAL binary exponent precision	*	7
18. DOUBLE PRECISION binary mantissa precision	30	47
19. DOUBLE PRECISION binary exponent precision	*	9
20. TIME decimal fractional second precision	*	0
21. TIMESTAMP decimal fractional second precision	*	6
22. INTERVAL decimal fractional second precision	*	6
23. INTERVAL decimal leading field precision	*	7
24. Columns in a table	100	250
25. Values in an INSERT statement	100	250
26. Set clauses in UPDATE statement	20	250
27. Length of a row (see Note 1)	2000	8000
28. Columns in UNIQUE constraint	6	15
29. Length of UNIQUE columns (Note 1)	120	750
30. Columns in GROUP BY column list	6	15
31. Length of GROUP BY column list (Note 1)	120	750
32. Sort items in ORDER BY clause	6	15
33. Length of ORDER BY column list (Note 1)	120	750
34. Referencing columns in FOREIGN KEY	6	15
35. Length of FOREIGN KEY column list (Note 1)	120	750
36. Table references in an SQL statement (Note 3)	15	50
37. Cursors simultaneously open	10	100
38. WHEN clauses in a CASE expression	*	50
39. Columns in a named columns JOIN	*	15
40. Length of JOIN column list (Note 1)	*	750
41. Items in a SELECT list	100	250
42. Length of SQL (Note 2)	*	30000
43. Length of (Note 2)	*	4000
44. Length of (Note 2)	*	4000
45. Occurrences in an ALLOCATE DESCRIPTOR	*	100
46. Default occurrences in ALLOCATE DESCRIPTOR	*	100

Note 1: The length of a collection of columns is conservatively estimated to be no larger than the sum of: twice the number of columns,OCTET_LENGTH of each character or bit column (see Subclause 6.6, **(**numeric value function**)**, of X3.135-1992), decimal precision plus 1 of each exact numeric column, binary precision divided by 4 plus 1 of each approximate numeric column, 10 for each DATE column, 8 for each TIME column, 14 for each TIME WITH TIME ZONE column, 19 for each TIMESTAMP column, 25 for each TIMESTAMP WITH TIME ZONE column, and 20 for each INTERVAL column. In addition, if any DATE, TIME, TIMESTAMP, or INTERVAL column has a non-zero fractional seconds precision, then add that precision plus 1 to the length of the column.

Note 2: The length of an SQL statement is defined to be the result of applying the

OCTET_LENGTH function (see Subclause 6.6, (numeric value function), of X3.135-1992) to the SQL statement with the SQL statement considered to be an instance of a CHARACTER VARYING data type.

Note 3: The number of table references in an SQL statement is the sum of: the number of views and base tables named in the statement, the number of underlying views and tables (see Subclause 4.9, "Tables", of X3.135-1992) for each derived table or cursor, and the number of) (either given in the SQL statement or contained in some view named in the SQL statement) not directly associated with a named table or view.

Some applications may have requirements for CHARACTER VARYING or BIT VARYING data types with lengths much longer than the Entry SQL or Intermediate SQL values specified above. This is particularly true for applications that need to manage large Audio, Graphics, Text, or Video objects. Some applications have requirements for Audio, Text, or Graphics objects in excess of 2-3 million bytes, or Video objects in excess of multiple gigabytes. Implementations that provide such data types often impose severe restrictions in how these very large objects can be referenced in SQL definitions and statements. For example, a very long CHARACTER VARYING data type may not be allowed to participate in a PRIMARY KEY, a UNIQUE constraint, a REFERENTIAL constraint, a) a GROUP BY or HAVING clause, or an ORDER BY in a cursor definition. Applications that stay within the limits specified above should not encounter any unexpected restrictions in how these constructs can be used or referenced in SQL language.

Some implementations address user requirements for very large objects, with a minimum number of restrictions, by allowing arbitrarily large maximum length declarations for CHARACTER VARYING or BIT VARYING, with an internal representation using some sort of indirect addressing mechanism. In this way they can keep the physical length of the row in which the object is represented less than the physical page size of the operating system environment, often necessary for lock management, while at the same time meet user requirements for storing, retrieving, and managing large objects. SQL procurements that anticipate requirements for very long CHARACTER VARYING or BIT VARYING data types should be very explicit in procurement specifications about additional requirements for how these large data types interface to external processors or how they need to be processed by SQL language.

16.7 Character set support. In ANSI Entry SQL, the set of character values for the character data types and the collation of characters in those data types are both implementation-defined. References to the Entry SQL level of this standard in a procurement should indicate any additional character data requirements. Failure to indicate specific character set requirements for the Entry SQL or Transitional SQL options of FIPS SQL means that support for representation of the 95-character graphic subset of ASCII (FIPS PUB 1-2), in an implementation-defined collating sequence, is by default the minimum requirement.

In Intermediate SQL, various SQL statements may use a (character set specification) to identify, by name, one or more different implementation- defined character sets. In addition, users may define new character sets with a (character set definition) For conformance to the Intermediate SQL or Full SQL options of FIPS SQL, it is required (see Section 10.e) that any (character set specification) be able to specify any one of the following character set names: SQL_CHARACTER, ASCII_GRAPHIC,

LATIN1, ASCII_FULL, or SQL_TEXT.

- -- If SQL_CHARACTER is specified, then the resulting character set consists of the 83) as specified in Subclause 5.1 of X3.135-1992. It consists of the 52 uppercase and lowercase simple latin characters, 10 digits, and 21 (SQL special character), including: (space) (double quote) (percent) (ampersand) (quote) (left paren) (right paren) (asterisk) (plus sign) (comma) (minus sign) (period)()olidus) (colon) ()emicolon) (less than operator) (equals operator) (greater than operator) (question mark) (underscore) and (vertical bar) The 83 characters specified as (SQL language character) are all included in the ISO International Reference Version (IRV) characters specified in ISO 646:1991. The characters in IRV are included in many other international character set definitions. In addition, 82 of these characters (all except (vertical bar)) are in the most stable subset of IRV that, by ISO convention, is included in every latin-based ISO standard set of characters. As far as can be determined, (vertical bar) is included in every character set that enjoys wide use in either the United States or Western Europe. Thus the SQL_CHARACTER repertoire is the mostuniversal of the character sets named in this FIPS. The collation and form-of-use of SQL_CHARACTER is implementation-defined.

- -- If ASCII_GRAPHIC is specified, then the resulting character set consists of the 95-character graphic subset of ASCII as specified in FIPS PUB 1-2. The form-of-use is that corresponding to the coded representation of each character by a single byte (possibly 7-bit, 8-bit, or other), with no designation escape sequences for other character sets. The default collating sequence is that corresponding to the bit combinations defined by FIPS PUB 1-2. The ASCII_GRAPHIC character set is a superset of the (SQL language character). The 12 characters included in ASCII_GRAPHIC that are not (SQL language character) are, in collation order: Exclamation mark !, Number sign #, Dollar sign $, Commercial at @, Left square bracket [, Reverse solidus \, Right square bracket], Circumflex accent ^, Grave accent `, Left curly bracket {, Right curly bracket }, and Tilde ~. Of these 12 characters, only "!" is in the most stable subset of ISO 646, whereas "#" competes with the British pound sign, "$" competes with the international currency symbol, and the others occupy positions in ISO 646 that are reserved for national or application-oriented use. However, all are the default IRV values specified in ISO 646 when no national or application-specific version is explicitly specified.

- -- If LATIN1 is specified, then the resulting character set consists of the 191 graphic characters defined in ISO 8859-1. The form-of-use is that corresponding to the coded representation of each character by a single 8-bit byte, with no designation escape sequences for other character sets. The default collating sequence is that corresponding to the bit combinations defined by ISO 8859-1. The LATIN1 character set is asuperset of ASCII_GRAPHIC and, when restricted to the ASCII_GRAPHICcharacters, produces the same collation as ASCII_GRAPHIC. LATIN1 consists of all characters commonly used in the following languages: Danish, Dutch, English, Faeroese, Finnish, French, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish, and Swedish. It also includes the following special symbols, in collation order: No-break space, Inverted exclamation mark,Cent sign, Pound sign, Currency sign, Yen sign, Broken bar, Paragraph sign, Diaeresis, Copyright sign, Feminine ordinal indicator, Left angle quotation mark, Not sign, Soft hyphen, Registered trade mark sign, Macron, Degree sign, Plus-minus sign, Superscript two, Superscript three, Acute accent, Micro sign, Pilcrow sign, Middle dot, Cedilla, Superscript one,

Masculine ordinal indicator, Right angle quotation mark, Fraction one quarter, Fraction one half, Fraction three quarters, and Inverted question mark. Other characters include the Multiplication sign and the Division sign. In LATIN1, all ASCII_GRAPHIC characters precede thenon-ASCII_GRAPHIC characters in the default collation, followed by the special symbols, followed by the accented capital letters, followed by the accented small letters. The Multiplication sign is in the middle of the accented capital letters and the Division sign is in the middle of the accented small letters.

LATIN1 is subject to the following conformance requirements, as specified in Clause 3, "Conformance", of ISO 8859-1:

a. A set of graphic characters is in conformance with Part 1 of ISO 8859 if it comprises all graphic characters specified therein to the exclusion of any other and if their coded representations are those specified by Part 1 of ISO 8859.

b. Equipment claimed to implement Part 1 of ISO 8859 shall implement all 191 characters.

- -- If ASCII_FULL is specified, then the resulting character set consists of all 256 characters of 8-bit ASCII, as specified in ANSI/ISO 4873 and ANSI/ISO 8859-1, including all control characters and all graphic characters. The form-of-use is that corresponding to the coded representation of each character by a single 8-bit byte, with no designation escape sequences for other character sets. The default collating sequence is that corresponding to the bit combinations defined by 8-bit ASCII. The ASCII_FULL character set is a superset of LATIN1 and, when restricted to the LATIN1 characters, produces the same collation and form-of-use.

- -- If SQL_TEXT is specified, then the resulting character set consists of the **(**SQL language character**)** and all characters that are in other character sets supported by the implementation, as specified in Syntax Rule 11 of Subclause 6.1, **(**data type**)**, of X3.135-1992. Thus, in FIPS SQL, the SQL_TEXT character set must be a superset of ASCII_FULL. The collation and form-of-use of SQL_TEXT is implementation-defined.

The character sets SQL_CHARACTER, ASCII_GRAPHIC, LATIN1, and ASCII_FULL have both a "floor" and "ceiling" requirement to consist of exactly the characters specified. Any character data type associated with one of these character sets has an implied integrity constraint limiting a value of the data type to be a character string consisting only of characters from the specified character set. The SQL_TEXT character set has a similar "floor" requirement in that it must contain all of the ASCII_FULL characters; however, SQL_TEXT does not have a "ceiling" requirement.

Requirements for FIPS Intermediate SQL or FIPS Full SQL in an SQL procurement should indicate any additional character data requirements. Failure to indicate specific character set, collation, conversion, or translation requirements means that support for SQL_CHARACTER, ASCII_GRAPHIC, LATIN1, ASCII_FULL, and SQL_TEXT, as indicated above, are the only requirements.

16.8 DBMS procurement. Database software is normally purchased as a complete package called a database management system (DBMS). A DBMS is an implementation of one or more data models

(e.g. the relational model, or an object model), together with other components, features, or data interfaces for efficient data administration. These additional facilities are not specified by this standard, so each procurement should itself specify the functional requirements of each additional feature desired.

Additional facilities most often contained in a DBMS package include: special-purpose data types (e.g. multimedia types or spatial data types), user-defined data types, object management, database import and export tools, backup and recovery tools, performance optimization tools, audit trails, networking, data dictionary, data storage specification, natural language query, report writer, graphical user interface, query by forms, CASE tools, or application development tools. Emerging specifications for an expanded SQL database language in ANSI and ISO standardization bodies may result in future standardization for some of these facilities; others may always remain implementation-defined.

The following features have "preliminary" syntax and semantics available in Working Draft form as part of an on-going ANSI and ISO/IEC standardization effort for further development of the SQL language. Generic support for some of these features may be specified as functional requirements, or as desirable elements, in a DBMS procurement, but any such procurement specification should be written knowing that the preliminary ANSI and ISO specifications are subject to substantial evolution or reconsideration before adoption in any future SQL standard. As these facilities evolve over the next several years and work their way through the various levels of the ANSI and ISO/IEC standardization process (i.e. CD and DIS), then more confidence can be put into referencing them in DBMS procurements. Features specified in preliminary form include:

1. SQL Call Level Interface (SQL/CLI). A new application program interface to SQL, initially specified for COBOL and C, that allows system calls to SQL services without the need for Embedded SQL preprocessing or Module language compilation. This interface would allow development of database client applications that could be linked to different SQL server implementations at execution time. This interface is a requirement for third-party software developers who wish only to distribute binary code to their customers. An ISO/IEC Working Draft specification was available in early 1993, with final standardization projected for 1994-1995.

2. Persistent SQL Modules. The ability to define packages of SQL procedures that "live" in the schema just like any other defined SQL object. Modules may be stored at remote nodes in a conforming communications network with only a remote procedure call needed to invoke a desired action. Persistent SQL Modules allow optimization of stored procedures at multiple sites in a communications network, thereby reducing both processing time and communications volume. An ISO/IEC Working Draft specification was available in early 1993, with final standardization projected for 1994-1995.

3. Abstract data type (ADT). A facility for user-defined data types, both structures and operations, using previously defined abstract data types and the standardized base types as primitives.

4. Dynamic assertions. Support for integrity constraints that are triggered by specific database actions, such as: after update, before insertion, or constraints based on comparing old and new

values of a given attribute. Assertions are "dynamic" in that they may reference before and after images of the database or may depend upon temporary data values that only exist at the time the invoking statement is executed.

5. Dynamic triggers. Support for triggering a sequence of database actions based on a specific database action, such as after delete, thereby supporting the object notion of encapsulation. Assertions and Triggers make it possible for object self-management to be fully specified in a database schema, with increased opportunity for performance optimization by the underlying database management system.

6. Object identity. A persistent object-identifier created for each instance of an object data type that is independent of the object's name, structure, or location, and that persists over time to forever avoid confusion with the identity of another object.

7. SQL functions. A function, defined completely in SQL, that helps to define the behavior of abstract data types. Constructor and destructor functions create or destroy new ADT instances, and actor functions read or modify ADT attributes.

8. External function call. A function whose interface specification is defined in an SQL schema, but with the content of the function written in some other programming language (e.g Fortran, Ada, or C++). USAGE privileges on external functions are managed by SQL GRANT and REVOKE statements.

9. Subtypes and inheritance. An abstraction mechanism that allows classes of abstract data types to be related hierarchically. Inheritance allows ADT classes to share properties and operations with other classes to allow more accurate and succinct modeling of applications.

10. Polymorphic functions. The ability to invoke an operation on any of several different objects and have each object determine how to respond, during execution, by applying rules for disambiguating names and data types.

11. Program control structures. Support for defining computationally complete SQL procedures by allowing sequences of SQL statements, looping, branching, and other flow of control statements, and dynamic exception handling on a per-procedure basis.

12. Parameterized types. The ability to define "type families", with a new data type for each value of an input parameter. Such type templates may also be nested, thereby greatly simplifying the definition of some complex nested structures.

13. Generator types. Support for type generators such as LIST, ARRAY, and SET, with SQL syntax, but harmonized with emerging specifications for generator and parameterized types in ISO/IEC programming language committees, i.e. ISO/IEC CD 11404, "Common language independent data types" (CLID).

14. Recursive expressions. Support for SQL expressions of indefinite, recursive depth, such as

those arising out of "bill-of-materials" part's hierarchies.

15. Existential and universal quantifiers. Support for more mathematically based SQL expressions involving the existential and universal quantifiers prevalent in 3-valued predicate logic.

16. SIMILAR predicate. A facility for pattern matching in bit strings and character strings that allows construction of regular expressions equivalent to regular expressions in the POSIX standard, ISO/IEC 9945.

17. Multiple null states. A facility that allows user definitions for an arbitrary number of application specific Null values, such as "Unknown", "Missing", "Not Applicable", "Pending", etc. Each such Null value would have a different representation in the database so that they could be distinguished by **)** during retrieval or update.

18. Roles and data security. An enhanced facility for database security management that builds upon the existing Grant and Revoke definitions. It extends the security model to include named "roles" in addition to schema objects, actions, and users. With roles defined as a nested collection of authorized actions on schema objects, security administration becomes more efficient and manageable.

19. Savepoints and subtransactions. A subtransaction is a portion of a transaction that is marked for potential rollback without affecting the other parts of the transaction. By setting and releasing savepoints, an application programmer is able to recover more easily from failed subtransactions, thereby leading to more efficient code.

20. Distributed database management. Distributed database management implies totally integrated, distributed data, under the coordinated control of multiple heterogeneous database management systems. Because it requires cooperating concurrency control managers, "standardized" distributed database management may be some time away; however, emerging standards for one-phase and two-phase commit protocols (see ISO/IEC 10026) allow individual implementations to access remote data and present a distributed view to their application programs.

21. Database export and import. Database export provides utilities for unloading a database definition and the data contents of a database into an external form, representable on various media, for the purpose of later automated re-generation. Database import provides utilities for loading a database definition and contents from an external source. Evolving specifications hope to make a database exported from any conforming SQL implementation importable into any other conforming SQL implementation.

22. Cursor sensitivity. The ability to specify a SENSITIVE option on a cursor definition, so that the cursor will always "see" concurrent modifications, in the same transaction, made to the underlying tables of the cursor definition. This provides a measure of predictability, but possibly with performance implications.

23. Asynchronous DML. Support for being able to name SQL statements so that other work can be done while they are executing. The names provide a mechanism for querying the status of outstanding statements.

A NIST special publication is under development that will discuss potential future directions for Database Language SQL. All of the above features are discussed further in that document.

16.9 DBMS performance. DBMS performance is often a critical factor in a DBMS procurement. This standard is silent on the topic of performance. The NIST SQL Test Suite (see Section 11.4) also makes no attempt to test the performance aspects of a conforming system. Whenever performance requirements are known in advance, they may be included as an integral part of the procurement specification.

A DBMS may also provide additional data structures, such as indices, or software, such as query optimizers, to enhance performance. User requirements for monitoring database activity or tools for tuning database performance should be specified explicitly.

Sometimes a procurement will specify a performance benchmark with agency supplied, or publicly available, data and applications. If a procurement specifies benchmarking, then the agency should consider that results of a benchmark for conforming interfaces may not be comparable to results of a benchmark for nonconforming interfaces. Consequently, procurement terminology should specify that performance benchmarks shall be performed with the actual interfaces to be used with agency applications. When modification of the actual interface is required after a benchmark, for example in the case of a delayed validation where correction of nonconformities within a year is specified, procurement terminology should specify that the performance benchmark be achieved with the corrected interface.

16.10 Database security. Some database management systems must operate in a highly secure environment that requires "trusted" database access control beyond the GRANT and REVOKE privilege facilities and the VIEW definition capabilities specified in this standard. Procurements for systems that operate in these environments should include explicit additional requirements that shall be supported. FIPS SQL contains specifications for some Discretionary Access Control (DAC) mechanisms, but not Mandatory Access Control (MAC) nor the associated security labels. For a definition of DAC or MAC, refer to "Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria" (NCSC-TG-021 Version 1, "Lavender Book"), National Computer Security Center, April 1991.

16.11 System integration. In many cases a database or a database management system must be integrated with other information processing systems operating in the same environment. Examples of other systems might include: the operating system, document processing systems, engineering CAD/CAM systems, graphics systems, an information resource dictionary system, statistical analysis systems, a transaction processing system, or an artificial intelligence system. In addition, distributed data under the control of different vendor's database management systems may require integration into a coordinated global view through remote database access or open distributed processing. Except for bindings to programming languages (see Clause 12 and Clause 19 of ANSI X3.135-1992) and the

Connection management statements (see Clause 15 of ANSI X3.135-1992), such integration is beyond the scope of this standard and, if desired, must be specified explicitly as part of procurement requirements.

17. Where to Obtain Copies. Copies of this publication are for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161, telephone 703-487-4650. (Sale of the included specification document, ANSI X3.135-1992, is by arrangement with the American National Standards Institute.) When ordering, refer to Federal Information Processing Standards Publication 127-2 (FIPS PUB 127-2), Database Language SQL. Payment may be made by check, money order, or deposit account.

The Foreword, Abstract, and Key Words follow:

FIPS PUB 127-2
FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION

1993 June 2
U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology

Database Language SQL

U.S. DEPARTMENT OF COMMERCE, Ronald H. Brown, *)ecretary*
National Institute of Standards and Technology, Arati Prabhakar, *Director*

Foreword

The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official publication relating to standards and guidelines adopted and promulgated under the provisions of Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235. These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computers and related telecommunications systems in the Federal Government. The NIST, through its Computer Systems Laboratory, provides leadership, technical guidance, and coordination of Government efforts in the development of standards and guidelines in these areas.

Comments concerning Federal Information Processing Standards Publications are welcomed and should be addressed to the Director, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.

James H. Burrows, *Director*
Computer Systems Laboratory

Abstract

This publication announces adoption of American National Standard Database Language SQL, ANSI X3.135-1992, as the Federal Information Processing Standard for Database Language SQL (FIPS SQL). It is a revision of FIPS PUB 127-1 that adds extensive new functionality to the SQL language. Conformance to FIPS SQL is mandatory for all Federal procurement of relational model database management systems.

FIPS SQL is specified to have four conformance levels: Entry SQL, Transitional SQL, Intermediate SQL, and Full SQL. Although only Entry SQL is required for conformance to FIPS SQL, the other conformance levels may be specified as mandatory requirements in individual procurement. FIPS SQL also provides default sizing parameters and limits for SQL constructs to provide a common baseline for database interoperability.

The purpose of FIPS SQL is to promote portability and of interoperability database application programs, to facilitate maintenance of database systems among heterogeneous data processing environments, and to allow for the efficient exchange of programmers among different data management projects.

Key words:ANSI standard; data manipulation language; database; database language standard; Embedded SQL; Federal Information Processing Standard (FIPS); ISO standard; module language; schema definition language; software ; Structured Query Language (SQL).

Important FIPS 127-2 Change Notice

U.S. DEPARTMENT OF COMMERENCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Gaithersburg, MD 20899

DATE OF CHANGE: 1993 October 4

Database Language SQL

This office has a record of your interest in receiving changes to the above FIPS PUB. The change(s) indicated below have been provided by the Maintenance Agency for this publication and will be included in the next published revision to this FIPS PUB. Questions or requests for additional information should be addressed to the Maintenance Agency:
Department Of Commerce
National Bureau Of Standards
Institute for Computer Sciences and Technology

Washington, D.C. 20234

Discussion

FIPS PUB 127-2 contains an inconsistency in the specification of FIPS Transitional SQL. The inconsistency is that) of length longer than 18 characters are not required to be supported by an implementation until conformance to FIPS Intermediate SQL is specified in a procurement, but support for certain tables in the Information Schema having some column names longer than 18 characters is required if FIPS Transitional SQL is specified. This is not a problem for the ISO/IEC or ANSI SQL specifications because the Information Schema is not required in those standards until Intermediate SQL is supported, and Intermediate SQL requires support for column names of up to 128 characters.

This FIPS SQL problem was discussed at a recent meeting of the ANSI/X3 technical committee responsible for standardization of Database Language SQL in the United States. That committee recommended that FIPS Transitional SQL requirements be modified to require support for the content of required Information Schema tables in a special FIPS schema different from the INFORMATION_SCHEMA and having shortened column names. NIST accepts this recommendation and will design the NIST SQL Test Suite for Transitional SQL to test for the existence of appropriate views in a special INFO_SCHEM schema.

The ANSI SQL committee also recommended that the following name shortening algorithm be used for consistency and convenience in being able to remember the shortened names:

DEFAULT ---> DEF
CHARACTER ---> CHAR
MAXIMUM ---> MAX
PRECISION ---> PREC
CATALOG ---> CAT
CHEMA ---> SCHEM
NUMERIC ---> NUM

NIST accepts this recommendation, and will apply it to all column names in an INFO_SCHEM view definition for all Information Schema tables required by Transitional SQL.

Applications that are designed to depend upon these shortened names may be ported to systems that support longer names or may outlive the 18 character name restriction on identifiers. Thus FIPS SQL will require that these special views in the INFO_SCHEM schema be supported by all implementations of FIPS Transitional SQL, including implementations that also support these tables in the INFORMATION_SCHEMA without difficulty. This additional FIPS Transitional SQL requirement is the small penalty that we must all pay for letting NIST make such a mistake in the first place!

This required feature of FIPS Transitional SQL is marked as a "deprecated" feature. The term "deprecated" means that a feature so labeled may not be supported in some future version of the standard, but it is still a fully supported and required feature of the existing standard.

Application programs written to reference tables and columns in the INFO_SCHEM can be modified automatically to execute in any Intermediate SQL processing environment by substituting "INFORMATION_SCHEMA" for "INFO_SCHEM" in any schema reference and by expanding the seven shortened words identified above in any column reference that is explicitly or implicitly qualified by an INFO_SCHEM schema name.

A second alternative to support programs written to reference tables and columns in the INFO_SCHEM in any Intermediate SQL processing environment is for the Database Administrator to define INFO_SCHEM directly as specified below. This alternative should add only a minimal runtime performance cost, and will avoid making changes to any individual application program.

FIPS SQL Modification

Add the following paragraphs to Section 10, "Specifications", of FIPS PUB 127-2:

For conformance to Transitional SQL, FIPS SQL requires that the implementation provide a special schema, the INFO_SCHEM schema, as a system-owned schema in every catalog supported by that implementation. This is a deprecated feature in FIPS 127-2. The INFO_SCHEM schema has, effectively, the following schema definition:

```
CREATE SCHEMA INFO_SCHEM
  AUTHORIZATION "_SYSTEM"
  DEFAULT CHARACTER SET SQL_TEXT

CREATE VIEW SCHEMATA
   (   CAT_NAME,
  SCHEM_NAME,
  SCHEM_OWNER,
  DEF_CHAR_SET_CAT,
  DEF_CHAR_SET_SCHEM,
  DEF_CHAR_SET_NAME
   )
   AS SELECT
  CATALOG_NAME, SCHEMA_NAME, SCHEMA_OWNER,
  DEFAULT_CHARACTER_SET_CATALOG,
  DEFAULT_CHARACTER_SET_SCHEMA,
  DEFAULT_CHARACTER_SET_NAME
   FROM INFORMATION_SCHEMA.SCHEMATA

CREATE VIEW TABLES
   (   TABLE_CAT,
  TABLE_SCHEM,
  TABLE_NAME,
  TABLE_TYPE
   )
   AS SELECT
  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_TYPE
   FROM INFORMATION_SCHEMA.TABLES

CREATE VIEW VIEWS
```

```
        (   TABLE_CAT,
      TABLE_SCHEM,
      TABLE_NAME,
      VIEW_DEFINITION,
      CHECK_OPTION,
      IS_UPDATABLE
       )
       AS SELECT
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, VIEW_DEFINITION,
      CHECK_OPTION, IS_UPDATABLE
       FROM INFORMATION_SCHEMA.VIEWS

   CREATE VIEW COLUMNS
      (   TABLE_CAT,
      TABLE_SCHEM,
      TABLE_NAME,
      COLUMN_NAME,
      ORDINAL_POSITION,
      COLUMN_DEF,
      IS_NULLABLE,
      DATA_TYPE,
      CHAR_MAX_LENGTH,
      CHAR_OCTET_LENGTH,
      NUM_PREC,
      NUM_PREC_RADIX,
      NUM_SCALE,
      DATETIME_PREC,
      INTERVAL_CODE,
      INTERVAL_PREC,
      CHAR_SET_CAT,
      CHAR_SET_SCHEM,
      CHAR_SET_NAME,
      COLLATION_CAT,
      COLLATION_SCHEM,
      COLLATION_NAME,
      DOMAIN_CAT,
      DOMAIN_SCHEM,
      DOMAIN_NAME
       )
       AS SELECT
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
      ORDINAL_POSITION, COLUMN_DEFAULT, IS_NULLABLE, DATA_TYPE,
      CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
      DATETIME_PRECISION, INTERVAL_CODE, INTERVAL_PRECISION,
      CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
CHARACTER_SET_NAME,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
      DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
       FROM INFORMATION_SCHEMA.COLUMNS

   CREATE VIEW TABLE_PRIVILEGES
      (   GRANTOR,
      GRANTEE,
      TABLE_CAT,
      TABLE_SCHEM,
      TABLE_NAME,
      PRIVILEGE_TYPE,
      IS_GRANTABLE
       )
```

```
      AS SELECT
     GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
     PRIVILEGE_TYPE, IS_GRANTABLE
      FROM INFORMATION_SCHEMA.TABLE_PRIVILEGES

   CREATE VIEW COLUMN_PRIVILEGES
      (   GRANTOR,
     GRANTEE,
     TABLE_CAT,
     TABLE_SCHEM,
     TABLE_NAME,
     COLUMN_NAME,
     PRIVILEGE_TYPE,
     IS_GRANTABLE
      )
      AS SELECT
     GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
     COLUMN_NAME, PRIVILEGE_TYPE, IS_GRANTABLE
      FROM INFORMATION_SCHEMA.COLUMN_PRIVILEGES

   CREATE VIEW USAGE_PRIVILEGES
      (   GRANTOR,
     GRANTEE,
     OBJECT_CAT,
     OBJECT_SCHEM,
     OBJECT_NAME,
     OBJECT_TYPE,
     PRIVILEGE_TYPE,
     IS_GRANTABLE
      )
      AS SELECT
     GRANTOR, GRANTEE, OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
     OBJECT_TYPE, PRIVILEGE_TYPE, IS_GRANTABLE
      FROM INFORMATION_SCHEMA.USAGE_PRIVILEGES

   GRANT SELECT, REFERENCES ON SCHEMATA TO PUBLIC WITH GRANT OPTION
   GRANT SELECT, REFERENCES ON TABLES TO PUBLIC WITH GRANT OPTION
   GRANT SELECT, REFERENCES ON VIEWS TO PUBLIC WITH GRANT OPTION
   GRANT SELECT, REFERENCES ON COLUMNS TO PUBLIC WITH GRANT OPTION
   GRANT SELECT, REFERENCES ON TABLE_PRIVILEGES TO PUBLIC WITH GRANT
OPTION
   GRANT SELECT, REFERENCES ON COLUMN_PRIVILEGES TO PUBLIC WITH
GRANT OPTION
   GRANT SELECT, REFERENCES ON USAGE_PRIVILEGES TO PUBLIC WITH GRANT
OPTION
```

Note: The COLUMNS view defined above anticipates an SQL Amendment that is currently being processed by ANSI/X3. That SQL correction, when finally approved, will add the INTERVAL_CODE and INTERVAL_PRECISION columns to the COLUMNS view in the INFORMATION_SCHEMA. These two columns were mistakenly omitted from ANSI X3.135-1992 as originally published.

Go Back to the Top.
Return to the FIPS
Home Page