CREATE TYPE MAPPING

The CREATE TYPE MAPPING statement creates a mapping between these data types:

- A data type of a column of a data source table or view that is going to be defined to a federated database
- A corresponding data type that is already defined to the federated database.

The mapping can associate the federated database data type with a data type at either (1) a specified data source or (2) a range of data sources; for example, all data sources of a particular type and version.

A data type mapping has to be created only if an existing one is not adequate.

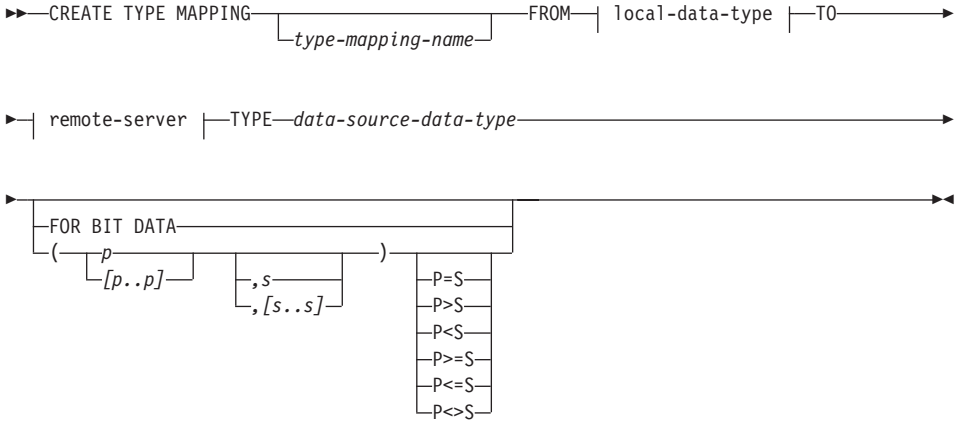
Invocation

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared. However, if the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared (SQLSTATE 42509).

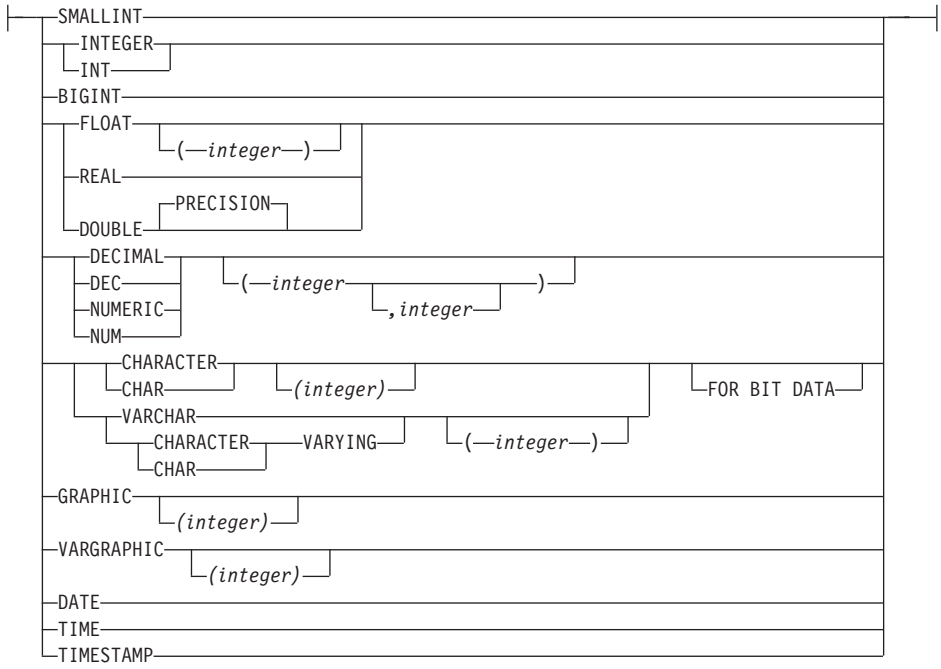
Authorization

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

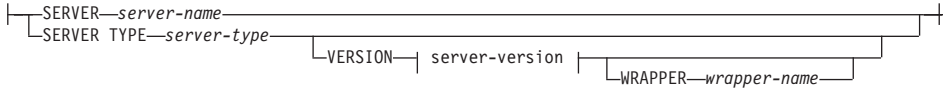
Syntax



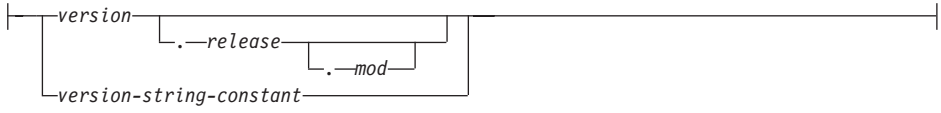
local-data-type:



remote-server:



server-version:



Description

type-mapping-name
Names the data type mapping. The name must not identify a data type mapping that is already described in the catalog. A unique name is generated if *type-mapping-name* is not specified.

local-data-type
Identifies a data type that is defined to a federated database. If *local-data-type* is specified without a schema name, the type name is

resolved by searching the schemas on the SQL path (defined by the FUNCPATH preprocessing option for static SQL and by the CURRENT PATH register for dynamic SQL). If length or precision (and scale) are not specified for the *local-data-type*, then the values are determined from the *source-data-type*.

The *local-data-type* cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, a large object (LOB) type, or a user-defined type (SQLSTATE 42806).

SERVER *server-name*
Names the data source to which *data-source-data-type* is defined.

SERVER TYPE *server-type*
Identifies the type of data source to which *data-source-data-type* is defined.

VERSION
Identifies the version of the data source to which *data-source-data-type* is defined.

version
Specifies the version number. *version* must be an integer.

release
Specifies the number of the release of the version denoted by *version*. *release* must be an integer.

mod
Specifies the number of the modification of the release denoted by *release. mod* must be an integer.

version-string-constant
Specifies the complete designation of the version. The *version-string-constant* can be a single value (for example, '8i'); or it can be the concatenated values of *version*, *release*, and, if applicable, *mod* (for example, '8.0.3').

WRAPPER *wrapper-name*
Specifies the name of the wrapper that the federated server uses to interact with data sources of the type and version denoted by *server-type* and *server-version*.

TYPE *data-source-data-type*
Specifies the data source data type that is being mapped to *local-data-type*. If *data-source-data-type* is qualified by a schema name at the data source, it is allowable, but not required, to specify this qualifier.

The *data-source-data-type* must be a built-in data type. User-defined types are not allowed. If the type has a short and long form (for example, CHAR and CHARACTER), the short form should be specified.

p For a decimal data type, *p* specifies the maximum number of digits that a

value can have. For all other data types for character data, *p* specifies the maximum number of characters that a value can have. The *p* must be valid with respect to the data type (SQLSTATE 42611). If *p* is not specified and the data type requires it, the system will determine the best match.

[p..p]
For a decimal data type, *[p..p]* specifies the minimum and maximum number of digits that a value can have. For all other data types for character data, *[p..p]* specifies the minimum and maximum number of characters that a value can have. In all cases, the maximum must equal or exceed the minimum; and both numbers must be valid with respect to the data type (SQLSTATE 42611).

s For a decimal data type, *s* specifies the allowable maximum number of digits to the right of the decimal point. This number must be valid with respect to the data type (SQLSTATE 42611). If a number is not specified and the data type requires one, the system will determine the best match.

[s..s]
For a decimal data type, *[s..s]* specifies the minimum and maximum number of digits allowed to the right of the decimal point. The maximum must equal or exceed the minimum, and both numbers must be valid with respect to the data type (SQLSTATE 42611).

P [operand] **S**
For a decimal data type, P [operand] S specifies a comparison between the maximum allowable precision and the maximum number of digits allowed to the right of the decimal point. For example, the operand = indicates that the maximum allowable precision and the maximum number digits allowed in the decimal fraction are the same. Specify P [operand] S only if the level of checking that it enforces is required.

FOR BIT DATA
Indicates whether *data-source-data-type* is for bit data. These keywords are required if the data source type column contains binary values. The database manager will determine this attribute if it is not specified on a character data type.

Notes

A CREATE TYPE MAPPING statement within a given unit of work (UOW) cannot be processed under either of the following conditions:

- The statement references a single data source, and the UOW already includes a SELECT statement that references a nickname for a table or view within this data source.
- The statement references a category of data sources (for example, all data sources of a specific type and version), and the UOW already includes a SELECT statement that references a nickname for a table or view within one of these data sources.

Examples

Example 1: Create a mapping between SYSIBM.DATE and the Oracle data type DATE at all Oracle data sources.

```
CREATE TYPE MAPPING MY_ORACLE_DATE
   FROM SYSIBM.DATE
   TO SERVER TYPE ORACLE
   TYPE DATE
```

Example 2: Create a mapping between SYSIBM.DECIMAL(10,2) and the Oracle data type NUMBER([10..38],2) at data source ORACLE1.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
   FROM SYSIBM.DECIMAL(10,2)
   TO SERVER ORACLE1
   TYPE NUMBER([10..38],2)
```

CREATE USER MAPPING

The CREATE USER MAPPING statement defines a mapping between an authorization ID that uses a federated database and the authorization ID and password to use at a specified data source.

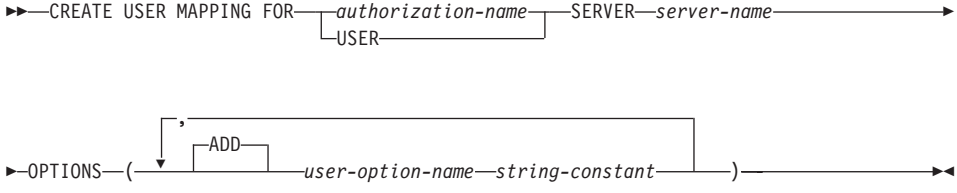
Invocation

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared. However, if the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared (SQLSTATE 42509).

Authorization

If the authorization ID of the statement is different than the authorization name that is being mapped to the data source, then the authorization ID must include SYSADM or DBADM authority. Otherwise, if the authorization ID and the authorization name match, then no privileges or authorities are required.

Syntax



Description

authorization-name
Specifies the authorization name under which a user or application connects to a federated database. This name is to be mapped to an identifier under which the data source denoted by *server-name* can be accessed.

USER
The value in the special register USER. When USER is specified, then the authorization ID of the CREATE USER MAPPING statement will be mapped to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

SERVER *server-name*
Identifies the data source that is accessible under the mapping authorization ID.

OPTIONS
Indicates what user options are to be enabled. Refer to "User Options" on page 1254 for descriptions of *user-option-name*s and their settings.

ADD
Enables one or more user options.

user-option-name
Names a user option that will be used to complete the user mapping that is being created.

string-constant
Specifies the setting for *user-option-name* as a character string constant.

Notes

- A user mapping cannot be created in a given unit of work (UOW) if the UOW already includes a SELECT statement that references a nickname for a table or view at the data source that is to be included in the mapping.

Examples

Example 1: To access a data source called S1, you need to map your authorization name and password for your local database to your user ID and password for S1. Your authorization name is RSPALTEN, and the user ID and password that you use for S1 are SYSTEM and MANAGER, respectively.

```
CREATE USER MAPPING FOR RSPALTEN
   SERVER S1
   OPTIONS
   ( REMOTE_AUTHID 'SYSTEM',
      REMOTE_PASSWORD 'MANAGER' )
```

Example 2: Marc already has access to a DB2 data source. He now needs access to an Oracle data source, so that he can create joins between certain DB2 and Oracle tables. He acquires a username and password for the Oracle data source; the username is the same as his authorization ID for the federated database, but his Oracle and federated database passwords are different. To be able to access Oracle from the federated database, he must map the two passwords together.

```
CREATE USER MAPPING FOR MARCR
   SERVER ORACLE1
   OPTIONS
   ( REMOTE_PASSWORD 'NZXCZY' )
```

CREATE VIEW

The CREATE VIEW statement creates a view on one or more tables, views or nicknames.

Invocation

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared. However, if the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared (SQLSTATE 42509).

Authorization

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority or
- For each table, view or nickname identified in any fullselect:
  - CONTROL privilege on that table or view, or
  - SELECT privilege on that table or view

  and at least one of the following:
  - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the view does not exist
  - CREATEIN privilege on the schema, if the schema name of the view refers to an existing schema.

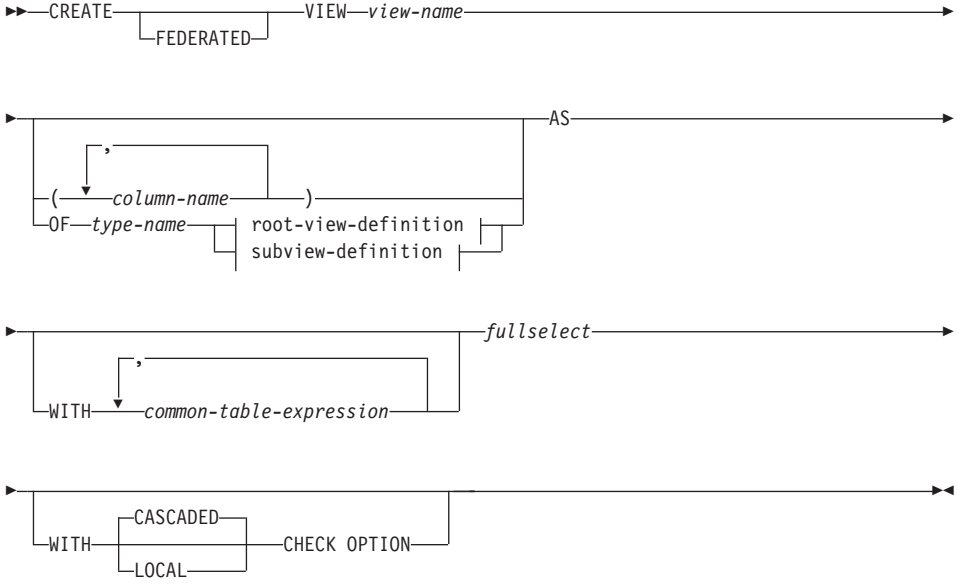
  If creating a subview, the authorization ID of the statement must:
  - be the same as the definer of the root table of the table hierarchy.
  - have SELECT WITH GRANT on the underlying table of the subview or the superview must not have SELECT privilege granted to any user other than the view definer.

Group privileges are not considered for any table or view specified in the CREATE VIEW statement.

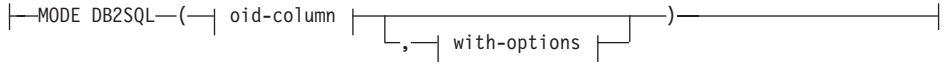
Privileges are not considered when defining a view on federated database nickname. Authorization requirements of the data source for the table or view referenced by the nickname are applied when the query is processed. The authorization ID of the statement may be mapped to a different remote authorization ID.

If a view definer can only create the view because the definer has SYSADM authority, then the definer is granted explicit DBADM authority for the purpose of creating the view.

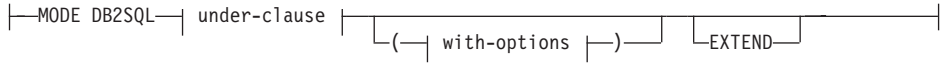
Syntax



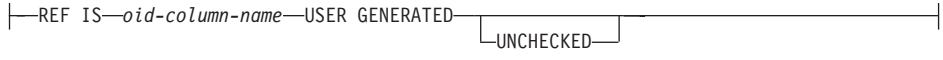
root-view-definition:



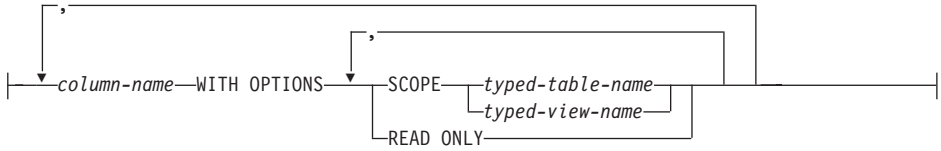
subview-definition:



oid-column:



with-options:



under-clause:

Note: See "Chapter 5. Queries" on page 393 for the syntax of *common-table-expression* and *fullselect*.

Description

FEDERATED
Indicates that the view being created references a nickname or an OLEDB table function. If an OLEDB table function or a nickname is directly, or indirectly, referenced in the fullselect and the FEDERATED keyword is not specified, a warning will be issued (SQLSTATE 01639) when the CREATE VIEW statement is submitted. However, the view will still be created.

Conversely, if an OLEDB table function or a nickname is not directly, or indirectly, referenced in the fullselect and the FEDERATED keyword is specified, an error will be issued (SQLSTATE 429BA) when the CREATE VIEW statement is submitted. The view will not be created.

view-name
Names the view. The name, including the implicit or explicit qualifier, must not identify a table, view, nickname or alias described in the catalog. The qualifier must not be SYSIBM, SYSCAT, SYSFUN, or SYSSTAT (SQLSTATE 42939).

The name can be the same as the name of an inoperative view (see "Inoperative views" on page 833). In this case the new view specified in the CREATE VIEW statement will replace the inoperative view. The user will get a warning (SQLSTATE 01595) when an inoperative view is replaced. No warning is returned if the application was bound with the bind option SQLWARN set to NO.

column-name
Names the columns in the view. If a list of column names is specified, it must consist of as many names as there are columns in the result table of the fullselect. Each *column-name* must be unique and unqualified. If a list of column names is not specified, the columns of the view inherit the names of the columns of the result table of the fullselect.

A list of column names must be specified if the result table of the fullselect has duplicate column names or an unnamed column (SQLSTATE 42908). An unnamed column is a column derived from a constant, function, expression, or set operation that is not named using the AS clause of the select list.

OF *type-name*
Specifies that the columns of the view are based on the attributes of the structured type identified by *type-name*. If *type-name* is specified without a schema name, the type name is resolved by searching the schemas on the SQL path (defined by the FUNCPATH preprocessing option for static SQL and by the CURRENT PATH register for dynamic SQL). The type name must be the name of an existing user-defined type (SQLSTATE 42704) and it must be a structured type that is instantiable (SQLSTATE 428DP).

MODE DB2SQL
This clause is used to specify the mode of the typed view. This is the only valid mode currently supported.

UNDER *superview-name*
Indicates that the view is a subview of *superview-name*. The superview must be an existing view (SQLSTATE 42704) and the view must be defined using a structured type that is the immediate supertype of *type-name* (SQLSTATE 428DB). The schema name of *view-name* and *superview-name* must be the same (SQLSTATE 428DQ). The view identified by *superview-name* must not have any existing subview already defined using *type-name* (SQLSTATE 42742).

The columns of the view include the object identifier column of the superview with its type modified to be REF(*type-name*), followed by columns based on the attributes of *type-name* (remember that the type includes the attributes of its supertype).

INHERIT SELECT PRIVILEGES
Any user or group holding a SELECT privilege on the superview will be granted an equivalent privilege on the newly created subview. The subview definer is considered to be the grantor of this privilege.

OID-column
Defines the object identifier column for the typed view.

REF IS *OID-column-name* **USER GENERATED**
Specifies that an object identifier (OID) column is defined in the view as the first column. An OID is required for the root view of a view hierarchy (SQLSTATE 428DX). The view must be a typed view (the OF clause must be present) that is not a subview (SQLSTATE 42613). The name for the column is defined as *OID-column-name* and cannot be the same as the name of any attribute of the structured type *type-name* (SQLSTATE 42711). The first column specified in *fullselect*

must be of type REF(*type-name*) (you may need to cast it so that it has the appropriate type). If UNCHECKED is not specified, it must be based on a not nullable column on which uniqueness is enforced through an index (primary key, unique constraint, unique index, or OID-column). This column will be referred to as the *object identifier column* or *OID column*. The keywords USER GENERATED indicate that the initial value for the OID column must be provided by the user when inserting a row. Once a row is inserted, the OID column cannot be updated (SQLSTATE 42808).

UNCHECKED
Defines the object identifier column of the typed view definition to assume uniqueness even though the system can not prove this uniqueness. This is intended for use with tables or views that are being defined into a typed view hierarchy where the user knows that the data conforms to this uniqueness rule but it does not comply with the rules that allow the system to prove uniqueness. UNCHECKED option is mandatory for view hierarchies that range over multiple hierarchies or legacy tables or views By specifying UNCHECKED, the user takes responsibility for ensuring that each row of the view has a unique OID. If the user fails to ensure this property, and a view contains duplicate OID values, then a path-expression or DEREF operator involving one of the non-unique OID values may result in an error (SQLSTATE 21000).

with-options
Defines additional options that apply to columns of a typed view.

column-name **WITH OPTIONS**
Specifies the name of the column for which additional options are specified. The *column-name* must correspond to the name of an attribute defined in (not inherited by) the *type-name* of the view. The column must be a reference type (SQLSTATE 42842). It cannot correspond to a column that also exists in the superview (SQLSTATE 428DJ). A column name can only appear in one WITH OPTIONS SCOPE clause in the statement (SQLSTATE 42613).

SCOPE
Identifies the scope of the reference type column. A scope must be specified for any column that is intended to be used as the left operand of a dereference operator or as the argument of the DEREF function.

Specifying the scope for a reference type column may be deferred to a subsequent ALTER VIEW statement (if the scope is not inherited) to allow the target table or view to be defined, usually in the case of mutually referencing views and tables. If no scope is specified for a reference type column of the view and the underlying table or view

column was scoped, then the underlying column's scope is inherited by the reference type column. The column remains unscoped if the underlying table or view column did not have a scope. See "Notes" on page 832 for more information about scope and reference type columns.

typed-table-name
The name of a typed table. The table must already exist or be the same as the name of the table being created (SQLSTATE 42704). The data type of *column-name* must be REF(*S*), where *S* is the type of *typed-table-name* (SQLSTATE 428DM). No checking is done of any existing values in *column-name* to ensure that the values actually reference existing rows in *typed-table-name*.

typed-view-name
The name of a typed view. The view must already exist or be the same as the name of the view being created (SQLSTATE 42704). The data type of *column-name* must be REF(*S*), where *S* is the type of *typed-view-name* (SQLSTATE 428DM). No checking is done of any existing values in *column-name* to ensure that the values actually reference existing rows in *typed-view-name*.

READ ONLY
Identifies the column as a read-only column. This option is used to force a column to be read-only so that subview definitions can specify an expression for the same column that is implicitly read-only.

AS
Identifies the view definition.

WITH *common-table-expression*
Defines a common table expression for use with the fullselect that follows. A common table expression cannot be specified when defining a typed view. See "common-table-expression" on page 440.

fullselect
Defines the view. At any time, the view consists of the rows that would result if the SELECT statement were executed. The fullselect must not reference host variables, parameter markers, or declared temporary tables. However, a parameterized view can be created as an SQL table function. See "CREATE FUNCTION (SQL Scalar, Table or Row)" on page 649.

For Typed Views and Subviews: The *fullselect* must conform to the following rules otherwise an error is returned (SQLSTATE 428EA unless otherwise specified).

- The fullselect must not include references to the NODENUMBER or PARTITION functions, non-deterministic functions, or functions defined to have external action.

- The body of the view must consist of a single subselect, or a UNION ALL of two or more subselects. Let each of the subselects participating directly in the view body be called a *branch* of the view. A view may have one or more branches.
- The FROM-clause of each branch must consist of a single table or view (not necessarily typed), called the *underlying* table or view of that branch.
- The underlying table or view of each branch must be in a separate hierarchy (i.e., a view may not have multiple branches with their underlying tables or views in the same hierarchy).
- None of the branches of a typed view definition may specify GROUP BY or HAVING.
- If the view body contains UNION ALL, then the root view in the hierarchy must specify the UNCHECKED option for its OID column.

For a hierarchy of views and subviews: Let BR1 and BR2 be any branches that appear in the definitions of views in the hierarchy. Let T1 be the underlying table or view of BR1, and let T2 be the underlying table or view of BR2. Then:

- If T1 and T2 are not in the same hierarchy, then the root view in the view hierarchy must specify the UNCHECKED option for its OID column.
- If T1 and T2 are in the same hierarchy, then BR1 and BR2 must contain predicates or ONLY-clauses that are sufficient to guarantee that their row-sets are disjoint.

For typed subviews defined using EXTEND AS: For every branch in the body of the subview:

- The underlying table of each branch must be a (not necessarily proper) subtable of some underlying table of the immediate superview.
- The expressions in the SELECT list must be assignable to the non-inherited columns of the subview (SQLSTATE 42854).

For typed subviews defined using AS without EXTEND:

- For every branch in the body of the subview, the expressions in the SELECT-list must be assignable to the declared types of the inherited and non-inherited columns of the subview (SQLSTATE 42854).
- The OID-expression of each branch over a given hierarchy in the subview must be equivalent (except for casting) to the OID-expression in the branch over the same hierarchy in the root view.
- The expression for a column not defined (implicitly or explicitly) as READ ONLY in a superview must be equivalent in all branches over the same underlying hierarchy in its subviews.

WITH CHECK OPTION

Specifies the constraint that every row that is inserted or updated through the view must conform to the definition of the view. A row that does not conform to the definition of the view is a row that does not satisfy the search conditions of the view.

WITH CHECK OPTION must not be specified if the view is read-only (SQLSTATE 42813). If WITH CHECK OPTION is specified for an updatable view that does not allow inserts, then the constraint applies to updates only.

WITH CHECK OPTION must not be specified if the view references the NODENUMBER or PARTITION function, a non-deterministic function, or a function with external action (SQLSTATE 42997).

WITH CHECK OPTION must not be specified if the view is a typed view (SQLSTATE 42997).

WITH CHECK OPTION must not be specified if a nickname is the update target of the view.

If WITH CHECK OPTION is omitted, the definition of the view is not used in the checking of any insert or update operations that use the view. Some checking might still occur during insert or update operations if the view is directly or indirectly dependent on another view that includes WITH CHECK OPTION. Because the definition of the view is not used, rows might be inserted or updated through the view that do not conform to the definition of the view.

CASCADED

The WITH CASCADED CHECK OPTION constraint on a view *V* means that *V* inherits the search conditions as constraints from any updatable view on which *V* is dependent. Furthermore, every updatable view that is dependent on *V* is also subject to these constraints. Thus, the search conditions of *V* and each view on which *V* is dependent are ANDed together to form a constraint that is applied for an insert or update of *V* or of any view dependent on *V*.

LOCAL

The WITH LOCAL CHECK OPTION constraint on a view *V* means the search condition of *V* is applied as a constraint for an insert or update of *V* or of any view that is dependent on *V*.

The difference between CASCADED and LOCAL is shown in the following example. Consider the following updatable views (substituting for Y from column headings of the table that follows):

```
V1 defined on table T
V2 defined on V1 WITH Y CHECK OPTION
V3 defined on V2
V4 defined on V3 WITH Y CHECK OPTION
V5 defined on V4
```

The following table shows the search conditions against which inserted or updated rows are checked:

	Y is LOCAL	**Y is CASCADED**
V1 checked against:	no view	no view
V2 checked against:	V2	V2, V1
V3 checked against:	V2	V2, V1
V4 checked against:	V2, V4	V4, V3, V2, V1
V5 checked against:	V2, V4	V4, V3, V2, V1

Consider the following updatable view which shows the impact of the WITH CHECK OPTION using the default CASCADED option:

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

The following INSERT statement using *V1* will succeed because *V1* does not have a WITH CHECK OPTION and *V1* is not dependent on any other view that has a WITH CHECK OPTION.

```
INSERT INTO V1 VALUES(5)
```

The following INSERT statement using *V2* will result in an error because *V2* has a WITH CHECK OPTION and the insert would produce a row that did not conform to the definition of *V2*.

```
INSERT INTO V2 VALUES(5)
```

The following INSERT statement using *V3* will result in an error even though it does not have WITH CHECK OPTION because *V3* is dependent on *V2* which does have a WITH CHECK OPTION (SQLSTATE 44000).

```
INSERT INTO V3 VALUES(5)
```

The following INSERT statement using *V3* will succeed even though it does not conform to the definition of *V3* (*V3* does not have a WITH CHECK OPTION); it does conform to the definition of *V2* which does have a WITH CHECK OPTION.

```
INSERT INTO V3 VALUES(200)
```

Notes

- Creating a view with a schema name that does not already exist will result in the implicit creation of that schema provided the authorization ID of the statement has IMPLICIT_SCHEMA authority. The schema owner is SYSIBM. The CREATEIN privilege on the schema is granted to PUBLIC.
- View columns inherit the NOT NULL WITH DEFAULT attribute from the base table or view except when columns are derived from an expression. When a row is inserted or updated into an updatable view, it is checked against the constraints (primary key, referential integrity, and check) if any are defined on the base table.
- A new view cannot be created if it uses an inoperative view in its definition. (SQLSTATE 51024).
- This statement does not support declared temporary tables (SQLSTATE 42995).
- ***Deletable views:*** A view is *deletable* if all of the following are true:
  - each FROM clause of the outer fullselect identifies only one base table (with no OUTER clause), deletable view (with no OUTER clause), deletable nested table expression, or deletable common table expression (cannot identify a nickname)
  - the outer fullselect does not include a VALUES clause
  - the outer fullselect does not include a GROUP BY clause or HAVING clause
  - the outer fullselect does not include column functions in the select list
  - the outer fullselect does not include SET operations (UNION, EXCEPT or INTERSECT) with the exception of UNION ALL
  - the base tables in the operands of a UNION ALL must not be the same table and each operand must be deletable
  - the select list of the outer fullselect does not include DISTINCT
- ***Updatable views:*** A column of a view is *updatable* if all of the following are true:
  - the view is deletable
  - the column resolves to a column of a base table (not using a dereference operation) and READ ONLY option is not specified
  - all the corresponding columns of the operands of a UNION ALL have exactly matching data types (including length or precision and scale) and matching default values if the fullselect of the view includes a UNION ALL

  A view is *updatable* if ANY column of the view is updatable.
- ***Insertable views:***

  A view is *insertable* if ALL columns of the view are updatable and the fullselect of the view does not include UNION ALL.

- ***Read-only views:*** A view is *read-only* if it is NOT deletable.

  The READONLY column in the SYSCAT.VIEWS catalog view indicates if a view is read-only.
- Common table expressions and nested table expressions follow the same set of rules for determining whether they are deletable, updatable, insertable or read-only.
- ***Inoperative views:*** An *inoperative view* is a view that is no longer available for SQL statements. A view becomes inoperative if:
  - A privilege, upon which the view definition is dependent, is revoked.
  - An object such as a table, nickname, alias or function, upon which the view definition is dependent, is dropped.
  - A view, upon which the view definition is dependent, becomes inoperative.
  - A view that is the superview of the view definition (the subview) becomes inoperative.

  In practical terms, an inoperative view is one in which the view definition has been unintentionally dropped. For example, when an alias is dropped, any view defined using that alias is made inoperative. All dependent views also become inoperative and packages dependent on the view are no longer valid.

  Until the inoperative view is explicitly recreated or dropped, a statement using that inoperative view cannot be compiled (SQLSTATE 51024) with the exception of the CREATE ALIAS, CREATE VIEW, DROP VIEW, and COMMENT ON TABLE statements. Until the inoperative view has been explicitly dropped, its qualified name cannot be used to create another table or alias (SQLSTATE 42710).

  An inoperative view may be recreated by issuing a CREATE VIEW statement using the definition text of the inoperative view. This view definition text is stored in the TEXT column of the SYSCAT.VIEWS catalog. When recreating an inoperative view, it is necessary to explicitly grant any privileges required on that view by others, due to the fact that all authorization records on a view are deleted if the view is marked inoperative. Note that there is no need to explicitly drop the inoperative view in order to recreate it. Issuing a CREATE VIEW statement with the same *view-name* as an inoperative view will cause that inoperative view to be replaced, and the CREATE VIEW statement will return a warning (SQLSTATE 01595).

  Inoperative views are indicated by an X in the VALID column of the SYSCAT.VIEWS catalog view and an X in the STATUS column of the SYSCAT.TABLES catalog view.

- *Privileges*

  The definer of a view always receives the SELECT privilege on the view as well as the right to drop the view. The definer of a view will get CONTROL privilege on the view only if the definer has CONTROL privilege on every base table, view or nickname identified in the fullselect, or if the definer has SYSADM or DBADM authority.

  The definer of the view is granted INSERT, UPDATE, column level UPDATE or DELETE privileges on the view if the view is not read-only and the definer has the corresponding privileges on the underlying objects.

  The definer of a view only acquires privileges if the privileges from which they are derived exist at the time the view is created. The definer must have these privileges either directly or because PUBLIC has the privilege. Privileges are not considered when defining a view on federated server nickname. However, when using a view on a nickname, the user's authorization ID must have valid select privileges on the table or view that the nickname references at the data source. Otherwise, an error is returned. Privileges held by groups of which the view definer is a member, are not considered.

  When a subview is created, the SELECT privileges held on the immediate superview are automatically granted on the subview.

- *Scope and REF columns*

  When selecting a reference type column in the fullselect of a view definition, consider the target type and scope that is required.

  – If the required target type and scope is the same as the underlying table or view, the column can simply be selected.
  – If the scope needs to be changed, use the WITH OPTIONS SCOPE clause to define the required scope table or view.
  – If the target type of the reference needs to be changed, the column must be cast first to the representation type of the reference and then to the new reference type. The scope in this case can be specified in the cast to the reference type or using the WITH OPTIONS SCOPE clause. For example, assume you select column Y defined as REF(TYP1) SCOPE TAB1. You want this to be defined as REF(VTYP1) SCOPE VIEW1. The select list item would be as follows:

    ```
    CAST(CAST(Y AS VARCHAR(16) FOR BIT DATA) AS REF(VTYP1) SCOPE VIEW1)
    ```

- ***Identity columns*** A column of a view is considered an identity column, if the element of the corresponding column in the fullselect of the view definition is the name of an identity column of a table, or the name of a column of a view which directly or indirectly maps to the name of an identity column of a base table.

  In all other cases, the columns of a view will not get the identity property. For example:

- the select-list of the view definition includes multiple instances of the name of an identity column (that is, selecting the same column more than once)
- the view definition involves a join
- a column in the view definition includes an expression that refers to an identity column
- the view definition includes a UNION

When inserting into a view for which the select list of the view definition directly or indirectly includes the name of an identity column of a base table, the same rules apply as if the INSERT statement directly referenced the identity column of the base table.

- ***Federated views*** A federated view is a view that includes a reference to a nickname somewhere in the fullselect. The presence of such a nickname changes the authorization model used for the view both at create time and when the view is subsequently referenced in a query. If a view is created that references a nickname and the FEDERATED keyword is not included, a warning is issued to indicate that the authorization requirements for this view are different because of the reference to a nickname.

  A nickname has no associated DML privileges and therefore when the view is created, no privilege checking is done to determine whether the view definer has access to the nickname or to the underlying data source table or view. Privilege checking of references to tables or views at the federated database are handled as usual, requiring the view definer to have at least SELECT privilege on such objects.

  When a federated view is subsequently referenced in a query, the nicknames result in queries against the data source and authorization ID that issued the query (or the remote authorization ID to which it maps) must have the necessary privileges to access the data source table or view. The authorization ID that issues the query referencing the federated view is not required to have any additional privileges on tables or views (non-federated) that exist at the federated server.

Examples

Example 1: Create a view named MA_PROJ upon the PROJECT table that contains only those rows with a project number (PROJNO) starting with the letters 'MA'.

```
CREATE VIEW MA_PROJ  AS SELECT *
   FROM PROJECT
    WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Example 2: Create a view as in example 1, but select only the columns for project number (PROJNO), project name (PROJNAME) and employee in charge of the project (RESPEMP).

```
CREATE VIEW MA_PROJ
 AS SELECTPROJNO, PROJNAME, RESPEMP
 FROM PROJECT
 WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Example 3: Create a view as in example 2, but, in the view, call the column for the employee in charge of the project IN_CHARGE.

```
CREATE VIEW MA_PROJ
(PROJNO, PROJNAME, IN_CHARGE)
AS SELECTPROJNO, PROJNAME, RESPEMP
 FROM PROJECT
 WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Note: Even though only one of the column names is being changed, the names of all three columns in the view must be listed in the parentheses that follow MA_PROJ.

Example 4: Create a view named PRJ_LEADER that contains the first four columns (PROJNO, PROJNAME, DEPTNO, RESPEMP) from the PROJECT table together with the last name (LASTNAME) of the person who is responsible for the project (RESPEMP). Obtain the name from the EMPLOYEE table by matching EMPNO in EMPLOYEE to RESPEMP in PROJECT.

```
CREATE VIEW PRJ_LEADER
 AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
 FROM PROJECT, EMPLOYEE
 WHERE RESPEMP = EMPNO
```

Example 5: Create a view as in example 4, but in addition to the columns PROJNO, PROJNAME, DEPTNO, RESPEMP, and LASTNAME, show the total pay (SALARY + BONUS + COMM) of the employee who is responsible. Also select only those projects with mean staffing (PRSTAFF) greater than one.

```
CREATE VIEW PRJ_LEADER
(PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY )
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
        FROM PROJECT, EMPLOYEE
        WHERE RESPEMP = EMPNO
        AND PRSTAFF > 1
```

Specifying the column name list could be avoided by naming the expression SALARY+BONUS+COMM as TOTAL_PAY in the fullselect.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP,
                      LASTNAME, SALARY+BONUS+COMM AS TOTAL_PAY
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

Example 6: Given the set of tables and views shown in the following figure:

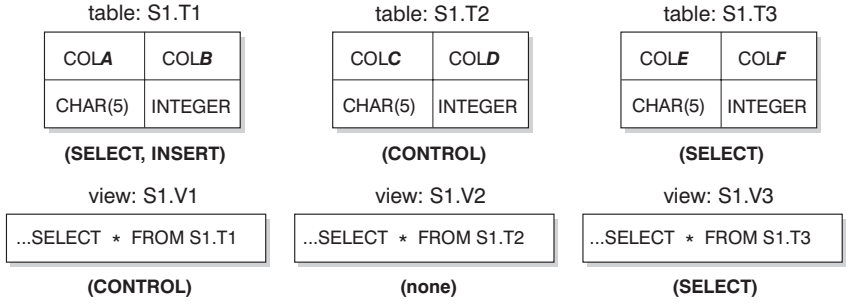


Figure 14. Tables and Views for Example 6

User ZORPIE (who does not have either DBADM or SYSADM authority) has been granted the privileges shown in brackets below each object:

1. ZORPIE will get CONTROL privilege on the view that she creates with:

   ```
   CREATE VIEW VA AS SELECT * FROM S1.V1
   ```

   because she has CONTROL on S1.V1.[86] It it does not matter which, if any, privileges she has on the underlying base table.
2. ZORPIE will not be allowed to create the view:

   ```
   CREATE VIEW VB AS SELECT * FROM S1.V2
   ```

   because she has neither CONTROL nor SELECT on S1.V2. It does not matter that she has CONTROL on the underlying base table (S1.T2).
3. ZORPIE will get CONTROL privilege on the view that she creates with:

   ```
   CREATE VIEW VC (COLA, COLB, COLC, COLD)
     AS SELECT * FROM S1.V1, S1.T2
     WHERE COLA = COLC
   ```

   because the fullselect of ZORPIE.VC references view S1.V1 and table S1.T2 and she has CONTROL on both of these. Note that the view VC is read-only, so ZORPIE does not get INSERT, UPDATE or DELETE privileges.
4. ZORPIE will get SELECT privilege on the view that she creates with:

   ```
   CREATE VIEW VD (COLA,COLB, COLE, COLF)
     AS SELECT * FROM S1.V1, S1.V3
     WHERE COLA = COLE
   ```

   because the fullselect of ZORPIE.VD references the two views S1.V1 and S1.V3, one on which she has only SELECT privilege, and one on which she has CONTROL privilege. She is given the lesser of the two privileges, SELECT, on ZORPIE.VD.

86. CONTROL on S1.V1 must have been granted to ZORPIE by someone with DBADM or SYSADM authority.

5. ZORPIE will get INSERT, UPDATE and DELETE privilege WITH GRANT OPTION and SELECT privilege on the view VE in the following view definition.

```
CREATE VIEW VE
AS SELECT * FROM S1.V1
WHERE COLA > ANY
       (SELECT COLE FROM S1.V3)
```

ZORPIE's privileges on VE are determined primarily by her privileges on S1.V1. Since S1.V3 is only referenced in a subquery, she only needs SELECT privilege on S1.V3 to create the view VE. The definer of a view only gets CONTROL on the view if they have CONTROL on all objects referenced in the view definition. ZORPIE does not have CONTROL on S1.V3, consequently she does not get CONTROL on VE.

CREATE WRAPPER

The CREATE WRAPPER statement registers a wrapper—a mechanism by which a federated server can interact with a certain category of data sources—to a federated database.

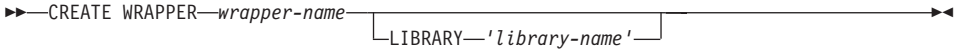
Invocation

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared. However, if the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared (SQLSTATE 42509).

Authorization

The authorization ID of the statement must have SYSADM or DBADM authority.

Syntax



Description

wrapper-name
Names the wrapper. It can be:

- A predefined name. If a predefined name is specified, the federated server automatically assigns a default to *'library-name'*.

  The predefined names are:

DRDA	For all DB2 family data sources
NET8	For all Oracle data sources that are supported by Oracle's Net8 client software
OLEDB	For all OLE DB providers supported by Microsoft OLE DB
SQLNET	For all Oracle data sources that are supported by Oracle's SQL*Net client software

- A user-supplied name. If such a name is provided, it is necessary to also specify *'library-name'*.

LIBRARY *'library-name'*
Names the file that contains the wrapper module. The LIBRARY option is only necessary when a user-supplied *wrapper-name* is used. This option should not be used when a predefined *wrapper-name* is given. The default file names for the predefined *wrapper-names* are:

Table 26. Default file names for LIBRARY option

Platform	DRDA	SQLNET	NET8	OLEDB
AIX	**libdrda.a**	**libsqlnet.a**	**libnet8.a**	–
HP-UX	**libdrda.sl**	**libsqlnet.sl**	**libnet8.sl**	–
SOLARIS	**libdrda.so**	**libsqlnet.so**	**libnet8.so**	–
UNIX	**libdrda.a**	**libsqlnet.a**	**libnet8.a**	–
WINNT	**drda.dll**	**sqlnet.dll**	**net8.dll**	**db2oledb.dll**

Notes

Refer to *Installation and Configuration Supplement* for more information on how to select and define wrappers.

Examples

Example 1: Register a wrapper that the federated server can use to interact with an Oracle data source that is supported by Oracle's SQL*Net client software. Use the predefined name.

```
CREATE WRAPPER SQLNET
```

Example 2: Register a wrapper that the federated server on an AIX system can use to interact with DB2 for VM and VSE data sources. Specify a name to indicate that these data sources are used for testing.

```
CREATE WRAPPER TEST
   LIBRARY 'libsqlds.a'
```

The extension in the library name (a) indicates that wrapper TEST is for data sources that reside in an AIX system.

DECLARE CURSOR

The DECLARE CURSOR statement defines a cursor.

Invocation

Although an interactive SQL facility might provide an interface that gives the appearance of interactive execution, this statement can only be embedded within an application program. It is not an executable statement and cannot be dynamically prepared.

Authorization

The term "SELECT statement of the cursor" is used in order to specify the authorization rules. The SELECT statement of the cursor is one of the following:

- The prepared select-statement identified by the *statement-name*
- The specified *select-statement*.

For each table or view identified (directly or using an alias) in the SELECT statement of the cursor, the privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority.
- For each table or view identified in the *select-statement:*
  - SELECT privilege on the table or view, or
  - CONTROL privilege of the table or view.

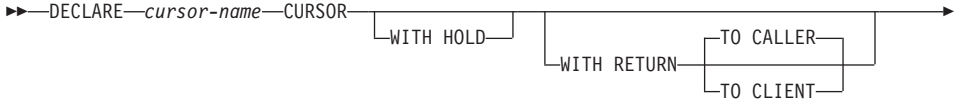
If *statement-name* is specified:

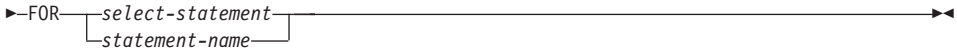
- The authorization ID of the statement is the run-time authorization ID.
- The authorization check is performed when the select-statement is prepared.
- The cursor cannot be opened unless the select-statement is successfully prepared.

If *select-statement* is specified:

- GROUP privileges are not checked.
- The authorization ID of the statement is the authorization ID specified during program preparation.

Syntax





Description

cursor-name
Specifies the name of the cursor created when the source program is run. The name must not be the same as the name of another cursor declared in the source program. The cursor must be opened before use (see "OPEN" on page 949).

WITH HOLD
Maintains resources across multiple units of work. The effect of the WITH HOLD cursor attribute is as follows:

- For units of work ending with COMMIT:
  - Open cursors defined WITH HOLD remain open. The cursor is positioned before the next logical row of the results table.

    If a DISCONNECT statement is issued after a COMMIT statement for a connection with WITH HOLD cursors, the held cursors must be explicitly closed or the connection will be assumed to have performed work (simply by having open WITH HELD cursors even though no SQL statements were issued) and the DISCONNECT statement will fail.
  - All locks are released, except locks protecting the current cursor position of open WITH HOLD cursors. The locks held include the locks on the table, and for parallel environments, the locks on rows where the cursors are currently positioned. Locks on packages and dynamic SQL sections (if any) are held.
  - Valid operations on cursors defined WITH HOLD immediately following a COMMIT request are:
    - FETCH: Fetches the next row of the cursor.
    - CLOSE: Closes the cursor.
  - UPDATE and DELETE CURRENT OF CURSOR are valid only for rows that are fetched within the same unit of work.
  - LOB locators are freed.
- For units of work ending with ROLLBACK:
  - All open cursors are closed.
  - All locks acquired during the unit of work are released.
  - LOB locators are freed.
- For special COMMIT case:
  - Packages may be recreated either explicitly, by binding the package, or implicitly, because the package has been invalidated and then

dynamically recreated the first time it is referenced. All held cursors are closed during package rebind. This may result in errors during subsequent execution.

WITH RETURN
This clause indicates that the cursor is intended for use as a result set from a stored procedure. WITH RETURN is relevant only if the DECLARE CURSOR statement is contained with the source code for a stored procedure. In other cases, the precompiler may accept the clause, but it has no effect.

Within an SQL procedure, cursors declared using the WITH RETURN clause that are still open when the SQL procedure ends, define the result sets from the SQL procedure. All other open cursors in an SQL procedure are closed when the SQL procedure ends. Within an external stored procedure (one not defined using LANGUAGE SQL), the WITH RETURN clause has no effect, and any cursors open at the end of an external procedure are considered the result sets.

TO CALLER
Specifies that the cursor can return a result set to the caller. For example, if the caller is another stored procedure, the result set is returned to that stored procedure. If the caller is a client application, the result set is returned to the client application.

TO CLIENT
Specifies that the cursor can return a result set to the client application. This cursor is invisible to any intermediate nested procedures.

select-statement
Identifies the SELECT statement of the cursor. The *select-statement* must not include parameter markers, but may include references to host variables. The declarations of the host variables must precede the DECLARE CURSOR statement in the source program. See "select-statement" on page 439 for an explanation of *select-statement*.

statement-name
The SELECT statement of the cursor is the prepared SELECT statement identified by the *statement-name* when the cursor is opened. The *statement-name* must not be identical to a *statement-name* specified in another DECLARE CURSOR statement of the source program.

For an explanation of prepared SELECT statements, see "PREPARE" on page 954.

Notes

- A program called from another program, or from a different source file within the same program, cannot use the cursor that was opened by the calling program.

- Unnested stored procedures, with LANGUAGE other than SQL, will have WITH RETURN TO CALLER as the default behavior if DECLARE CURSOR is specified without a WITH RETURN clause, and the cursor is left open in the procedure. This provides compatibility with stored procedures from previous versions that allow stored procedures to return result sets to applicable client applications. To avoid this behavior, close all cursors opened in the procedure.
- If the SELECT statement of a cursor contains CURRENT DATE, CURRENT TIME, or CURRENT TIMESTAMP, all references to these special registers will yield the same value on each FETCH. This value is determined when the cursor is opened.
- For more efficient processing of data, the database manager can block data for read-only cursors when retrieving data from a remote server. The use of the FOR UPDATE clause helps the database manager decide whether a cursor is updatable or not. Updatability is also used to determine the access path selection as well. If a cursor is not going to be used in a Positioned UPDATE or DELETE statement, it should be declared as FOR READ ONLY.
- A cursor in the open state designates a result table and a position relative to the rows of that table. The table is the result table specified by the SELECT statement of the cursor.
- A cursor is *deletable* if all of the following are true:
  - each FROM clause of the outer fullselect identifies only one base table or deletable view (cannot identify a nested or common table expression or a nickname) without use of the OUTER clause
  - the outer fullselect does not include a VALUES clause
  - the outer fullselect does not include a GROUP BY clause or HAVING clause
  - the outer fullselect does not include column functions in the select list
  - the outer fullselect does not include SET operations (UNION, EXCEPT, or INTERSECT) with the exception of UNION ALL
  - the select list of the outer fullselect does not include DISTINCT
  - the select-statement does not include an ORDER BY clause
  - the select-statement does not include a FOR READ ONLY clause[87]
  - one or more of the following is true:
    - the FOR UPDATE clause[88] is specified
    - the cursor is statically defined
    - the LANGLEVEL bind option is MIA or SQL92E

87. The FOR READ ONLY clause is defined in "read-only-clause" on page 447.

88. The FOR UPDATE clause is defined in "update-clause" on page 446.

A column in the select list of the outer fullselect associated with a cursor is *updatable* if all of the following are true:

- the cursor is deletable
- the column resolves to a column of the base table
- the LANGLEVEL bind option is MIA, SQL92E or the select-statement includes the FOR UPDATE clause (the column must be specified explicitly or implicitly in the FOR UPDATE clause).

A cursor is *read-only* if it is not deletable.

A cursor is *ambiguous* if all of the following are true:

- the select-statement is dynamically prepared
- the select-statement does not include either the FOR READ ONLY clause or the FOR UPDATE clause
- the LANGLEVEL bind option is SAA1
- the cursor otherwise satisfies the conditions of a deletable cursor.

An ambiguous cursor is considered read-only if the BLOCKING bind option is ALL, otherwise it is considered deletable.

- Cursors in stored procedures that are called by application programs written using CLI can be used to define result sets that are returned directly to the client application. Cursors in SQL procedures can also be returned to a calling SQL procedure only if they are defined using the WITH RETURN clause. See the "Notes" on page 527.

Example

The DECLARE CURSOR statement associates the cursor name C1 with the results of the SELECT.

```
EXEC SQL DECLARE C1 CURSOR FOR
   SELECT DEPTNO, DEPTNAME, MGRNO
   FROM DEPARTMENT
   WHERE ADMRDEPT = 'A00';
```