ALTER TABLE

The ALTER TABLE statement modifies existing tables by:

- Adding one or more columns to a table
- Adding or dropping a primary key
- Adding or dropping one or more unique or referential constraints
- Adding or dropping one or more check constraint definitions
- Altering the length of a VARCHAR column
- Altering a reference type column to add a scope
- Altering the generation expression of a generated column
- Adding or dropping a partitioning key
- Changing table attributes such as the data capture option, pctfree, lock size, or append mode.
- Setting the table to not logged initially state.

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:

- ALTER privilege on the table to be altered
- CONTROL privilege on the table to be altered
- ALTERIN privilege on the schema of the table
- SYSADM or DBADM authority.

To create or drop a foreign key, the privileges held by the authorization ID of the statement must include one of the following on the parent table:

- REFERENCES privilege on the table
- REFERENCES privilege on each column of the specified parent key
- CONTROL privilege on the table
- SYSADM or DBADM authority.

To drop a primary key or unique constraint of table T, the privileges held by the authorization ID of the statement must include at least one of the following on every table that is a dependent of this parent key of T:

- ALTER privilege on the table
- CONTROL privilege on the table
- ALTERIN privilege on the schema of the table

- SYSADM or DBADM authority.

To alter a table to become a summary table (using a fullselect), the privileges held by the authorization ID of the statement must include at least one of the following:

- CONTROL on the table
- SYSADM or DBADM authority;

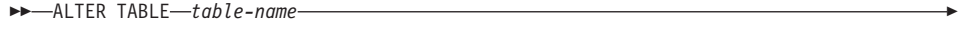
and at least one of the following, on each table or view identified in the fullselect:

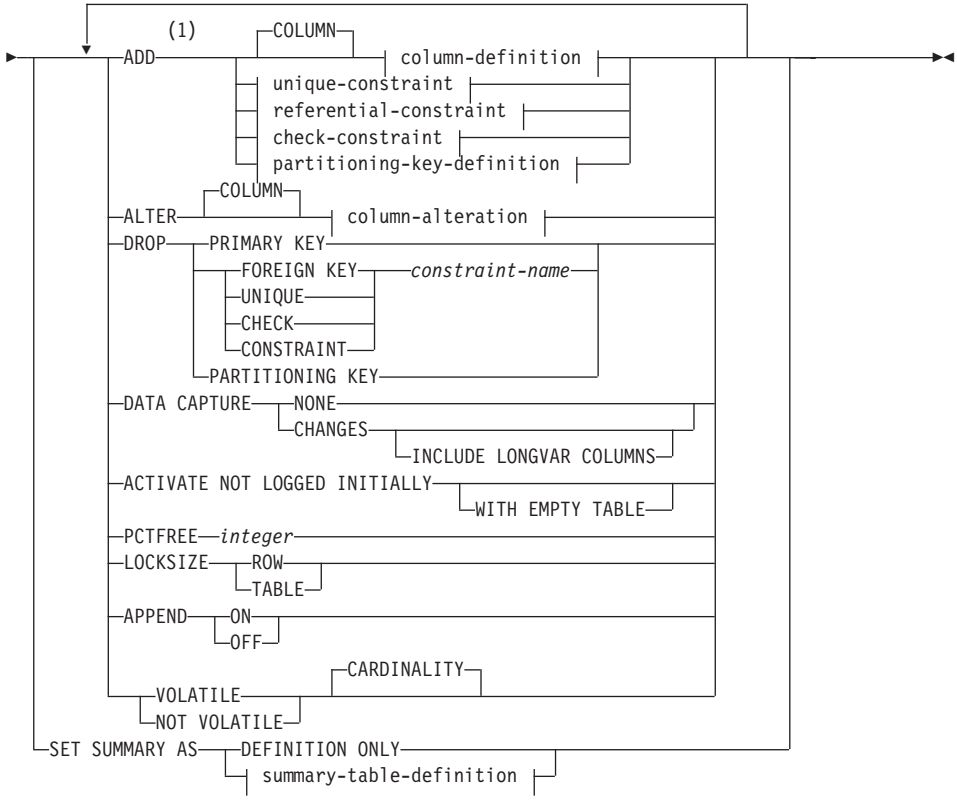
- SELECT and ALTER privilege on the table or view
- CONTROL privilege on the table or view
- SELECT privilege on the table or view and ALTERIN privilege on the schema of the table or view
- SYSADM or DBADM authority.

To alter a table so that it is no longer a summary table, the privileges held by the authorization ID of the statement must include at least one of the following, on each table or view identified in the fullselect used to define the summary table:

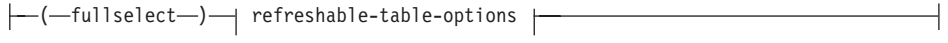
- ALTER privilege on the table or view
- CONTROL privilege on the table or view
- ALTERIN privilege on the schema of the table or view
- SYSADM or DBADM authority

Syntax

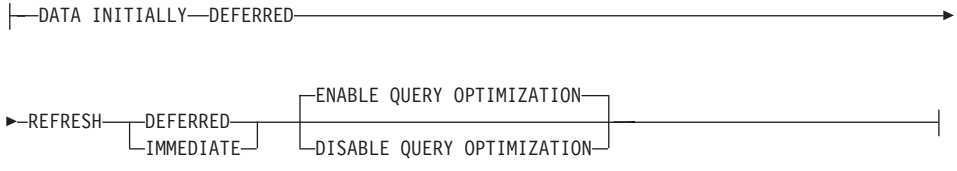




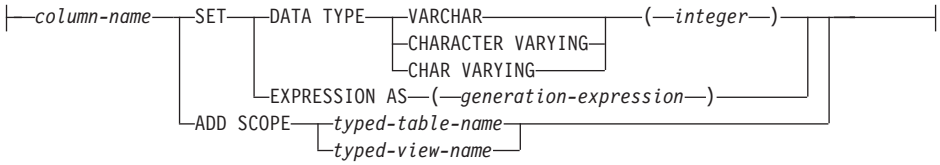
summary-table-definition:



refreshable-table-options:



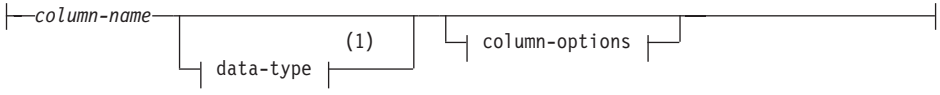
column-alteration:



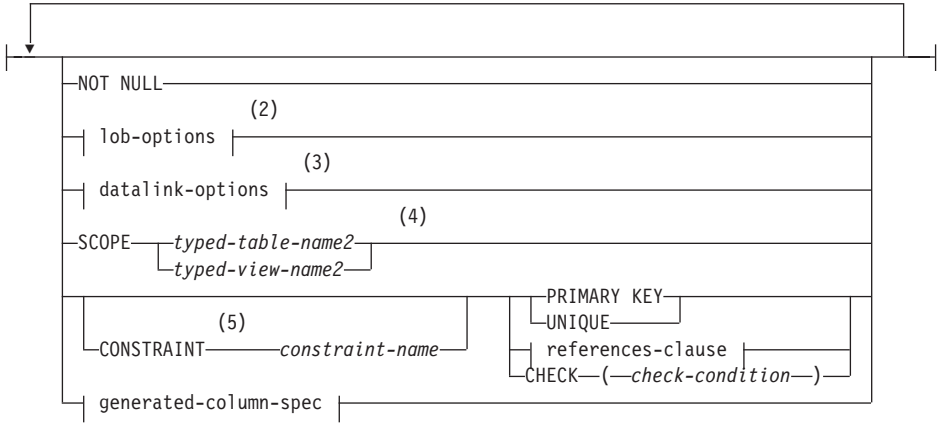
Notes:

1 For compatibility with Version 1, the ADD keyword is optional for:
- unnamed PRIMARY KEY constraints
- unnamed referential constraints
- referential constraints whose name follows the phrase FOREIGN KEY.

column-definition:



column-options:



Notes:

1 If the first column-option chosen is the generated-column-spec, then the data-type can be omitted and computed by the generation-expression.

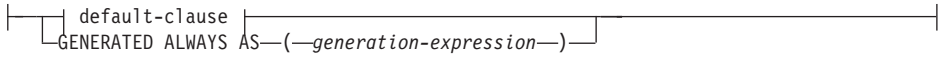
2 The lob-options clause only applies to large object types (BLOB, CLOB and DBCLOB) and distinct types based on large object types.

3 The datalink-options clause only applies to the DATALINK type and distinct types based on the DATALINK type.

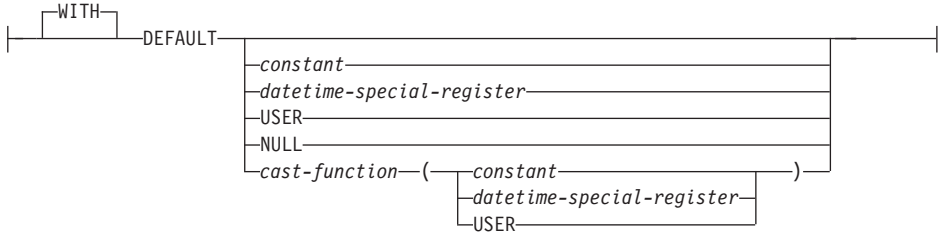
4 The SCOPE clause only applies to the REF type.

5 For compatibility with Version 1, the CONSTRAINT keyword may be omitted in a *column-definition* defining a references-clause.

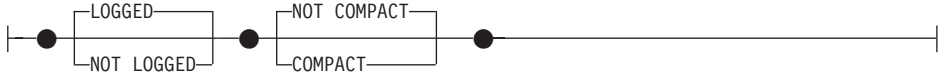
generated-column-spec:



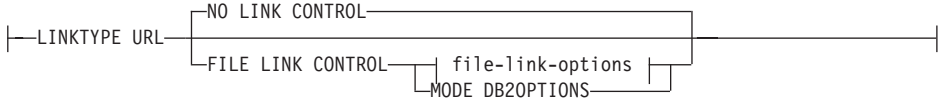
default-clause:



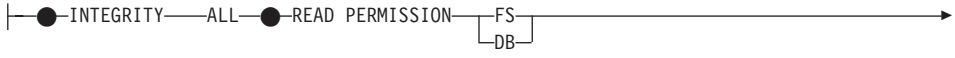
lob-options:

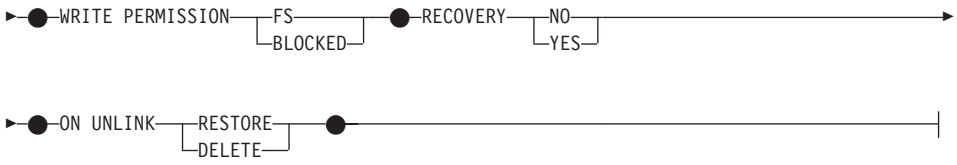


datalink-options:

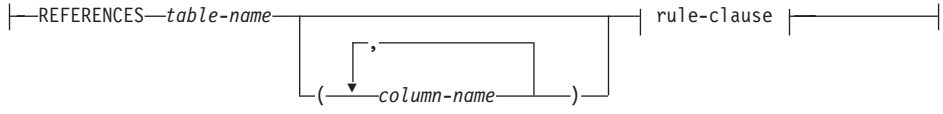


file-link-options:

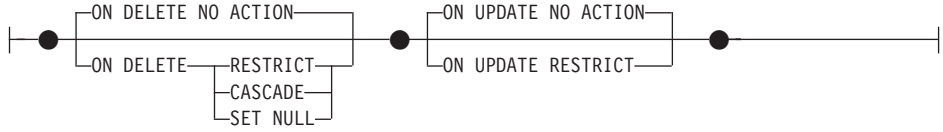




references-clause:

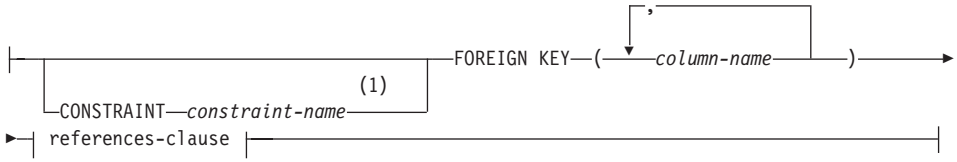


rule-clause:

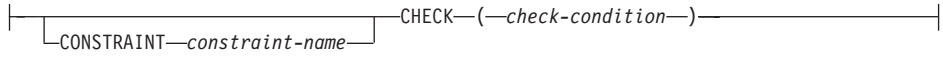


unique-constraint:

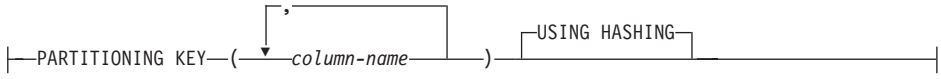
referential-constraint:



check-constraint:



partitioning-key-definition:



Notes:

1 For compatibility with Version 1, *constraint-name* may be specified following FOREIGN KEY (without the CONSTRAINT keyword).

Description

table-name
Identifies the table to be changed. It must be a table described in the catalog and must not be a view or a catalog table. If *table-name* identifies a summary table, alterations are limited to setting the summary table to definition only, activating not logged initially, changing pctfree, locksize, append, or volatile. The *table-name* cannot be a nickname (SQLSTATE 42809) or a declared temporary table (SQLSTATE 42995).

SET SUMMARY AS
Allows alteration of the properties of a summary table.

DEFINITION ONLY
Change a summary table so that it is no longer considered a summary table. The table specified by *table-name* must be defined as a summary table that is not replicated (SQLSTATE 428EW). The definition of the columns of *table-name* is not changed but the table can no longer be used for query optimization and the REFRESH TABLE statement can no longer be used.

summary-table-definition
Changes a regular table to a summary table for use during query optimization. The table specified by *table-name* must not:

- be previously defined as a summary table
- be a typed table
- have any constraints, unique indexes, or triggers defined
- be referenced in the definition of another summary table.

If table-name does not meet these criteria, an error is returned (SQLSTATE 428EW).

fullselect
Defines the query in which the table is based. The columns of the existing table must:

- have the same number of columns
- have exactly the same data types

- have the same column names in the same ordinal positions

as the result columns of *fullselect* (SQLSTATE 428EW). For details about specifying the *fullselect* for a summary table, see "CREATE TABLE" on page 712. One additional restriction is that *table-name* cannot be directly or indirectly referenced in the fullselect.

refreshable-table-options
Lists the refreshable options for altering a summary table.

DATA INITIALLY DEFERRED
The data in the table must be validated using the REFRESH TABLE or SET INTEGRITY statement.

REFRESH
Indicates how the data in the table is maintained.

DEFERRED
The data in the table can be refreshed at any time using the REFRESH TABLE statement. The data in the table only reflects the result of the query as a snapshot at the time the REFRESH TABLE statement is processed. Summary tables defined with this attribute do not allow INSERT, UPDATE, or DELETE statements (SQLSTATE 42807).

IMMEDIATE
The changes made to the underlying tables as part of a DELETE, INSERT, or UPDATE are cascaded to the summary table. In this case, the content of the table, at any point-in-time, is the same as if the specified subselect is processed. Summary tables defined with this attribute do not allow INSERT, UPDATE, or DELETE statements (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION
The summary table can be used for query optimization.

DISABLE QUERY OPTIMIZATION
The summary table will not be used for query optimization. The table can still be queried directly.

ADD *column-definition*
Adds a column to the table. The table must not be a typed table (SQLSTATE 428DH). If the table has existing rows, every value of the newly added column'' is its default value. The new column is the last column of the table. That is, if initially there are *n* columns, the added column is column *n*+1. The value of *n* cannot be greater than 499.

Adding the new column must not make the total byte count of all columns exceed the maximum record size as specified in Table 34 on page 1105. See "Notes" on page 753 for more information.

column-name
Is the name of the column to be added to the table. The name cannot be qualified. Existing column names in the table cannot be used (SQLSTATE 42711).

data-type
Is one of the data types listed under "CREATE TABLE" on page 712.

NOT NULL
Prevents the column from containing null values. The *default-clause* must also be specified (SQLSTATE 42601).

lob-options
Specifies options for LOB data types. See *lob-options* in "CREATE TABLE" on page 712.

datalink-options
Specifies options for DATALINK data types. See *datalink-options* in "CREATE TABLE" on page 712.

SCOPE
Specify a scope for a reference type column.

typed-table-name2
The name of a typed table. The data type of *column-name* must be REF(*S*), where *S* is the type of *typed-table-name2* (SQLSTATE 428DM). No checking is done of the default value for *column-name* to ensure that the value actually references an existing row in *typed-table-name2*.

typed-view-name2
The name of a typed view. The data type of *column-name* must be REF(*S*), where *S* is the type of *typed-view-name2* (SQLSTATE 428DM). No checking is done of the default value for *column-name* to ensure that the values actually references an existing row in *typed-view-name2*.

CONSTRAINT *constraint-name*
Names the constraint. A *constraint-name* must not identify a constraint that was already specified within the same ALTER TABLE statement, or as the name of any other existing constraint on the table (SQLSTATE 42710).

If the constraint name is not specified by the user, an 18-character identifier unique within the identifiers of the existing constraints defined on the table, is generated[59] by the system.

59. The identifier is formed of "SQL" followed by a sequence of 15 numeric characters generated by a timestamp-based function.

When used with a PRIMARY KEY or UNIQUE constraint, the *constraint-name* may be used as the name of an index that is created to support the constraint. See "Notes" on page 497 for details on index names associated with unique constraints.

PRIMARY KEY
This provides a shorthand method of defining a primary key composed of a single column. Thus, if PRIMARY KEY is specified in the definition of column C, the effect is the same as if the PRIMARY KEY(C) clause were specified as a separate clause. The column cannot contain null values, so the NOT NULL attribute must also be specified (SQLSTATE 42831).

See PRIMARY KEY within the description of the *unique-constraint* below.

UNIQUE
This provides a shorthand method of defining a unique key composed of a single column. Thus, if UNIQUE is specified in the definition of column C, the effect is the same as if the UNIQUE(C) clause were specified as a separate clause.

See UNIQUE within the description of the *unique-constraint* below.

references-clause
This provides a shorthand method of defining a foreign key composed of a single column. Thus, if a references-clause is specified in the definition of column C, the effect is the same as if that references-clause were specified as part of a FOREIGN KEY clause in which C is the only identified column.

See *references-clause* in "CREATE TABLE" on page 712.

CHECK (*check-condition***)**
This provides a shorthand method of defining a check constraint that applies to a single column. See *check-condition* in "CREATE TABLE" on page 712.

generate-column-spec
See "CREATE TABLE" on page 712 for details on column-generation.

default-clause
Specifies a default value for the column.

WITH
An optional keyword.

DEFAULT
Provides a default value in the event a value is not supplied on INSERT or is specified as DEFAULT on INSERT or UPDATE. If a specific default value is not specified following the DEFAULT

keyword, the default value depends on the data type of the column as shown in Table 19. If a column is defined as a DATALINK or structured type, then a DEFAULT clause cannot be specified.

If a column is defined using a distinct type, then the default value of the column is the default value of the source data type cast to the distinct type.

Table 19. Default Values (when no value specified)

Data Type	Default Value
Numeric	0
Fixed-length character string	Blanks
Varying-length character string	A string of length 0
Fixed-length graphic string	Double-byte blanks
Varying-length graphic string	A string of length 0
Date	For existing rows, a date corresponding to January 1, 0001. For added rows, the current date.
Time	For existing rows, a time corresponding to 0 hours, 0 minutes, and 0 seconds. For added rows, the current time.
Timestamp	For existing rows, a date corresponding to January 1, 0001, and a time corresponding to 0 hours, 0 minutes, 0 seconds and 0 microseconds. For added rows, the current timestamp.
Binary string (blob)	A string of length 0

Omission of DEFAULT from a *column-definition* results in the use of the null value as the default for the column.

Specific types of values that can be specified with the DEFAULT keyword are as follows.

constant
Specifies the constant as the default value for the column. The specified constant must:

- represent a value that could be assigned to the column in accordance with the rules of assignment as described in Chapter 3
- not be a floating-point constant unless the column is defined with a floating-point data type

- not have non-zero digits beyond the scale of the column data type if the constant is a decimal constant (for example, 1.234 cannot be the default for a DECIMAL(5,2) column)
- be expressed with no more than 254 characters including the quote characters, any introducer character such as the X for a hexadecimal constant, and characters from the fully qualified function name and parentheses when the constant is the argument of a *cast-function*.

datetime-special-register
Specifies the value of the datetime special register (CURRENT DATE, CURRENT TIME, or CURRENT TIMESTAMP) at the time of INSERT or UPDATE as the default for the column. The data type of the column must be the data type that corresponds to the special register specified (for example, data type must be DATE when CURRENT DATE is specified). For existing rows, the value is the current date, current time or current timestamp when the ALTER TABLE statement is processed.

USER
Specifies the value of the USER special register at the time of INSERT or UPDATE as the default for the column. If USER is specified, the data type of the column must be a character string with a length not less than the length attribute of USER. For existing rows, the value is the authorization ID of the ALTER TABLE statement.

NULL
Specifies NULL as the default for the column. If NOT NULL was specified, DEFAULT NULL must not be specified within the same column definition.

cast-function
This form of a default value can only be used with columns defined as a distinct type, BLOB or datetime (DATE, TIME or TIMESTAMP) data type. For distinct type, with the exception of distinct types based on BLOB or datetime types, the name of the function must match the name of the distinct type for the column. If qualified with a schema name, it must be the same as the schema name for the distinct type. If not qualified, the schema name from function resolution must be the same as the schema name for the distinct type. For a distinct type based on a datetime type, where the default value is a constant, a function must be used and the name of the function must match the name of the source type of the distinct type with an implicit or explicit schema name of SYSIBM. For other datetime columns, the corresponding datetime function may also be used. For a BLOB or a distinct type

based on BLOB, a function must be used and the name of the function must be BLOB with an implicit or explicit schema name of SYSIBM.

constant
Specifies a constant as the argument. The constant must conform to the rules of a constant for the source type of the distinct type or for the data type if not a distinct type. If the cast-function is BLOB, the constant must be a string constant.

datetime-special-register
Specifies CURRENT DATE, CURRENT TIME, or CURRENT TIMESTAMP. The source type of the distinct type of the column must be the data type that corresponds to the specified special register.

USER
Specifies the USER special register. The data type of the source type of the distinct type of the column must be a string data type with a length of at least 8 bytes. If the cast-function is BLOB, the length attribute must be at least 8 bytes.

If the value specified is not valid, an error (SQLSTATE 42894) is raised.

ADD *unique-constraint*
Defines a unique or primary key constraint. A primary key or unique constraint cannot be added to a table that is a subtable (SQLSTATE 429B3). If the table is a supertable at the top of the hierarchy, the constraint applies to the table and all its subtables.

CONSTRAINT *constraint-name*
Names the primary key or unique constraint. For more information, see *constraint-name* in "CREATE TABLE" on page 712.

UNIQUE (*column-name***...,)**
Defines a unique key composed of the identified columns. The identified columns must be defined as NOT NULL. Each *column-name* must identify a column of the table and the same column must not be identified more than once. The name cannot be qualified. The number of identified columns must not exceed 16 and the sum of their stored lengths must not exceed 1024 (refer to "Byte Counts" on page 757 for the column stored lengths). The length of any individual column must not exceed 255 bytes. No LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, distinct type on any of these types, or structured type column may be used as part of a unique key (even if the length attribute of the column is small enough to fit within the 255 byte limit) (SQLSTATE 42962). The set of columns in the unique key cannot be the same as the set of columns of the primary key or

another unique key (SQLSTATE 01543).[60] Any existing values in the set of identified columns must be unique (SQLSTATE 23515).

A check is performed to determine if an existing index matches the unique key definition (ignoring any INCLUDE columns in the index). An index definition matches if it identifies the same set of columns without regard to the order of the columns or the direction (ASC/DESC) specifications. If a matching index definition is found, the description of the index is changed to indicate that it is required by the system and it is changed to unique (after ensuring uniqueness) if it was a non-unique index. If the table has more than one matching index, an existing unique index is selected (the selection is arbitrary). If no matching index is found, a unique index will automatically be created for the columns, as described in CREATE TABLE. See "Notes" on page 497 for details on index names associated with unique constraints.

PRIMARY KEY *...(column-name,)*

Defines a primary key composed of the identified columns. Each *column-name* must identify a column of the table, and the same column must not be identified more than once. The name cannot be qualified. The number of identified columns must not exceed 16 and the sum of their stored lengths must not exceed 1024 (refer to "Byte Counts" on page 757 for the stored lengths). The length of any individual column must not exceed 255 bytes. The table must not have a primary key and the identified columns must be defined as NOT NULL. No LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, distinct type on any of these types, or structured type column may be used as part of a primary key (even if the length attribute of the column is small enough to fit within the 255 byte limit) (SQLSTATE 42962). The set of columns in the primary key cannot be the same as the set of columns of a unique key (SQLSTATE 01543).[60] Any existing values in the set of identified columns must be unique (SQLSTATE 23515).

A check is performed to determine if an existing index matches the primary key definition (ignoring any INCLUDE columns in the index). An index definition matches if it identifies the same set of columns without regard to the order of the columns or the direction (ASC/DESC) specifications. If a matching index definition is found, the description of the index is changed to indicate that it is the primary index, as required by the system, and it is changed to unique (after ensuring uniqueness) if it was a non-unique index. If the table has more than one matching index, an existing unique index is selected (the selection is arbitrary). If no matching index is found, a

60. If LANGLEVEL is SQL92E or MIA then an error is returned, SQLSTATE 42891.

unique index will automatically be created for the columns, as described in CREATE TABLE. See "Notes" on page 497 for details on index names associated with unique constraints.

Only one primary key can be defined on a table.

ADD *referential-constraint*
Defines a referential constraint. See *referential-constraint* in "CREATE TABLE" on page 712.

ADD *check-constraint*
Defines a check constraint. See *check-constraint* in "CREATE TABLE" on page 712.

ADD *partitioning-key-definition*
Defines a partitioning key. The table must be defined in a table space on a single-partition nodegroup and must not already have a partitioning key. If a partitioning key already exists for the table, the existing key must be dropped before adding the new partitioning key.

A partitioning key cannot be added to a table that is a subtable (SQLSTATE 428DH).

PARTITIONING KEY (*column-name***...)**
Defines a partitioning key using the specified columns. Each *column-name* must identify a column of the table, and the same column must not be identified more than once. The name cannot be qualified. A column cannot be used as part of a partitioning key if the data type of the column is a LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, distinct type on any of these types, or structured type.

USING HASHING
Specifies the use of the hashing function as the partitioning method for data distribution. This is the only partitioning method supported.

ALTER *column-alteration*
Alters the characteristics of a column.

column-name
Is the name of the column to be altered in the table. The *column-name* must identify an existing column of the table (SQLSTATE 42703). The name cannot be qualified.

SET DATA TYPE VARCHAR (*integer***)**
Increases the length of an existing VARCHAR column. CHARACTER VARYING or CHAR VARYING can be used as synonyms for the VARCHAR keyword. The data type of *column-name* must be VARCHAR and the current maximum length defined for the column

must not be greater than the value for *integer* (SQLSTATE 42837). The value for *integer* may range up to 32672. The table must not be a typed table (SQLSTATE 428DH).

Altering the column must not make the total byte count of all columns exceed the maximum record size as specified in Table 34 on page 1105 (SQLSTATE 54010). See "Notes" on page 753 for more information. If the column is used in a unique constraint or an index, the new length must not be greater than 255 bytes and must not cause the sum of the stored lengths for the unique constraint or index to exceed 1024 (SQLSTATE 54008) (refer to "Byte Counts" on page 757 for the stored lengths).

SET EXPRESSION AS (*generation-expression***)**
Changes the expression for the column to the specified *generation-expression*. SET EXPRESSION AS requires the table to be put in check pending state using the SET INTEGRITY statement. After the ALTER TABLE statement, the SET INTEGRITY statement must be used to update and check all the values in that column against the new expression. The column must already be defined as a generated column based on an expression (SQLSTATE 42837). The generation-expression must conform to the same rules that apply when defining a generated column. The result data type of the generation-expression must be assignable to the data type of the column (SQLSTATE 42821).

ADD SCOPE
Add a scope to an existing reference type column that does not already have a scope defined (SQLSTATE 428DK). If the table being altered is a typed table, the column must not be inherited from a supertable (SQLSTATE 428DJ). Refer to "ALTER TYPE (Structured)" on page 509 for examples.

typed-table-name
The name of a typed table. 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 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*.

DROP PRIMARY KEY
Drops the definition of the primary key and all referential constraints dependent on this primary key. The table must have a primary key.

DROP FOREIGN KEY *constraint-name*
Drops the referential constraint *constraint-name*. The *constraint-name* must identify a referential constraint. For information on implications of dropping a referential constraint see "Notes" on page 497.

DROP UNIQUE *constraint-name*
Drops the definition of the unique constraint *constraint-name* and all referential constraints dependent on this unique constraint. The *constraint-name* must identify an existing UNIQUE constraint. For information on implications of dropping a unique constraint, see "Notes" on page 497.

DROP CONSTRAINT *constraint-name*
Drops the constraint *constraint-name*. The *constraint-name* must identify an existing check constraint, referential constraint, primary key or unique constraint defined on the table. For information on implications of dropping a constraint, see "Notes" on page 497.

DROP CHECK *constraint-name*
Drops the check constraint *constraint-name*. The *constraint-name* must identify an existing check constraint defined on the table.

DROP PARTITIONING KEY
Drops the partitioning key. The table must have a partitioning key and must be in a table space defined on a single-partition nodegroup.

DATA CAPTURE
Indicates whether extra information for data replication is to be written to the log.

If the table is a typed table, then this option is not supported (SQLSTATE 428DH for root tables or 428DR for other subtables).

NONE
Indicates that no extra information will be logged.

CHANGES
Indicates that extra information regarding SQL changes to this table will be written to the log. This option is required if this table will be replicated and the Capture program is used to capture changes for this table from the log.

If the table is defined to allow data on a partition other than the catalog partition (multiple partition nodegroup or nodegroup with partition other than the catalog partition), then this option is not supported (SQLSTATE 42997).

If the schema name (implicit or explicit) of the table is longer than 18 bytes, then this option is not supported (SQLSTATE 42997).

Further information about using replication can be found in the *Administration Guide* and the *Replication Guide and Reference*.

INCLUDE LONGVAR COLUMNS
Allows data replication utilities to capture changes made to LONG VARCHAR or LONG VARGRAPHIC columns. The clause may be specified for tables that do not have any LONG VARCHAR or LONG VARGRAPHIC columns since it is possible to ALTER the table to include such columns.

ACTIVATE NOT LOGGED INITIALLY
Activates the NOT LOGGED INITIALLY attribute of the table for this current unit of work. The table must have been originally created with the NOT LOGGED INITIALLY attribute (SQLSTATE 429AA).

Any changes made to the table by an INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX, or ALTER TABLE in the same unit of work after the table is altered by this statement are not logged. Any changes made to the system catalog by the ALTER statement in which the NOT LOGGED INITIALLY attribute is activated are logged. Any subsequent changes made in the same unit of work to the system catalog information are logged.

At the completion of the current unit of work, the NOT LOGGED INITIALLY attribute is deactivated and all operations that are done on the table in subsequent units of work are logged.

If using this feature to avoid locks on the catalog tables while inserting data, it is important that only this clause be specified on the ALTER TABLE statement. Use of any other clause in the ALTER TABLE statement will result in catalog locks. If no other clauses are specified for the ALTER TABLE statement, then only a SHARE lock will be acquired on the system catalog tables. This can greatly reduce the possibility of concurrency conflicts for the duration of time between when this statement is executed and when the unit of work in which it was executed is ended.

If the table is a typed table, this option is only supported on the root table of the typed table hierarchy (SQLSTATE 428DR).

For more information on the NOT LOGGED INITIALLY attribute, see the description of this attribute in "CREATE TABLE" on page 712.

Note: If a table has been altered by activating the NOT LOGGED INITIALLY attribute within a unit of work, a rollback to savepoint request will be converted to a rollback to unit of work request (SQLSTATE 40506). An error in any operation in the unit of work in which the NOT LOGGED INITIALLY attribute is active will result

in the entire unit of work being rolled back (SQLSTATE 40506). Furthermore, the table for which the NOT LOGGED INITIALLY attribute was activated is **marked inaccessible after the rollback** has occurred and can only be dropped. Therefore, the opportunity for errors within the unit of work in which the NOT LOGGED INITIALLY attribute is activated should be minimized.

WITH EMPTY TABLE
Causes all data currently in table to be removed. Once the data has been removed, it cannot be recovered except through use of the RESTORE facility. If the unit of work in which this Alter statement was issued is rolled back, the table data will NOT be returned to its original state.

When this action is requested, no DELETE triggers defined on the affected table are fired. Any indexes that exist on the table are also emptied.

PCTFREE *integer*
Indicates what percentage of each page to leave as free space during load or reorganization. The value of *integer* can range from 0 to 99. The first row on each page is added without restriction. When additional rows are added, at least *integer* percent of free space is left on each page. The PCTFREE value is considered only by the LOAD and REORGANIZE TABLE utilities. If the table is a typed table, this option is only supported on the root table of the typed table hierarchy (SQLSTATE 428DR).

LOCKSIZE
Indicates the size (granularity) of locks used when the table is accessed. Use of this option in the table definition will not prevent normal lock escalation from occurring. If the table is a typed table, this option is only supported on the root table of the typed table hierarchy (SQLSTATE 428DR).

ROW
Indicates the use of row locks. This is the default lock size when a table is created.

TABLE
Indicates the use of table locks. This means that the appropriate share or exclusive lock is acquired on the table and intent locks (except intent none) are not used. Use of this value may improve the performance of queries by limiting the number of locks that need to be acquired. However, concurrency is also reduced since all locks are held over the complete table.

Further information about locking can be found in the *Administration Guide*.

APPEND
Indicates whether data is appended to the end of the table data or placed where free space is available in data pages. If the table is a typed table, this option is only supported on the root table of the typed table hierarchy (SQLSTATE 428DR).

ON
Indicates that table data will be appended and information about free space on pages will not be kept. The table must not have a clustered index (SQLSTATE 428CA).

OFF
Indicates that table data will be placed where there is available space. This is the default when a table is created.

The table should be reorganized after setting APPEND OFF since the information about available free space is not accurate and may result in poor performance during insert.

VOLATILE
This indicates to the optimizer that the cardinality of table *table-name* can vary significantly at run time, from empty to quite large. To access *table-name* the optimizer will use an index scan rather than a table scan, regardless of the statistics, if that index is index-only (all columns referenced are in the index) or that index is able to apply a predicate in the index scan. If the table is a typed table, this option is only supported on the root table of the typed table hierarchy (SQLSTATE 428DR).

NOT VOLATILE
This indicates to the optimizer that the cardinality of *table-name* is not volatile. Access Plans to this table will continue to be based on the existing statistics and on the optimization level in place.

CARDINALITY
An optional key word to indicate that it is the number of rows in the table that is volatile and not the table itself.

Rules

- A partitioning key column of a table cannot be updated (SQLSTATE 42997).
- Any unique or primary key constraint defined on the table must be a superset of the partitioning key, if there is one (SQLSTATE 42997).
- A nullable column of a partitioning key cannot be included as a foreign key column when the relationship is defined with ON DELETE SET NULL (SQLSTATE 42997).
- A column can only be referenced in one ADD or ALTER COLUMN clause in a single ALTER TABLE statement (SQLSTATE 42711).
- A column length cannot be altered if the table has any summary tables that are dependent on the table (SQLSTATE 42997).

- Before adding a generated column, the table must be set into the check-pending state, using the SET INTEGRITY statement (SQLSTATE 55019).

Notes

- Altering a table to a summary table will put the table in check-pending state. If the table is defined as REFRESH IMMEDIATE, the table must be taken out of check-pending state before INSERT, DELETE, or UPDATE commands can be invoked on the table referenced by the fullselect. The table can be taken out of check-pending state by using REFRESH TABLE or SET INTEGRITY, with the IMMEDIATE CHECKED option, to completely refresh the data in the table based on the fullselect. If the data in the table accurately reflects the result of the fullselect, the IMMEDIATE UNCHECKED option of SET INTEGRITY can be used to take the table out of check-pending state.
- Altering a table to change it to a REFRESH IMMEDIATE summary table will cause any packages with INSERT, DELETE, or UPDATE usage on the table referenced by the fullselect to be invalidated.
- Altering a table to change from a summary table to a regular table (DEFINITION ONLY) will cause any packages dependent on the table to be invalidated.
- ADD column clauses are processed prior to all other clauses. Other clauses are processed in the order that they are specified.
- Any columns added via ALTER TABLE will not automatically be added to any existing view of the table.
- When an index is automatically created for a unique or primary key constraint, the database manager will try to use the specified constraint name as the index name with a schema name that matches the schema name of the table. If this matches an existing index name or no name for the constraint was specified, the index is created in the SYSIBM schema with a system-generated name formed of "SQL" followed by a sequence of 15 numeric characters generated by a timestamp based function.
- Any table that may be involved in a DELETE operation on table T is said to be *delete-connected* to T. Thus, a table is delete-connected to T if it is a dependent of T or it is a dependent of a table in which deletes from T cascade.
- A package has an insert (update/delete) usage on table T if records are inserted into (updated in/deleted from) T either directly by a statement in the package, or indirectly through constraints or triggers executed by the package on behalf of one of its statements. Similarly, a package has an update usage on a column if the column is modified directly by a statement in the package, or indirectly through constraints or triggers executed by the package on behalf of one of its statements.

- Any changes to primary key, unique keys, or foreign keys may have the following effect on packages, indexes, and other foreign keys.
  - If a primary key or unique key is added:
    - There is no effect on packages, foreign keys, or existing unique keys.[61]
  - If a primary key or unique key is dropped:
    - The index is dropped if it was automatically created for the constraint. Any packages dependent on the index are invalidated.
    - The index is set back to non-unique if it was converted to unique for the constraint and it is no longer system-required. Any packages dependent on the index are invalidated.
    - The index is set to no longer system required if it was an existing unique index used for the constraint. There is no effect on packages.
    - All dependent foreign keys are dropped. Further action is taken for each dependent foreign key, as specified in the next item.
  - If a foreign key is added or dropped:
    - All packages with an insert usage on the object table are invalidated.
    - All packages with an update usage on at least one column in the foreign key are invalidated.
    - All packages with a delete usage on the parent table are invalidated.
    - All packages with an update usage on at least one column in the parent key are invalidated.
- Adding a column to a table will result in invalidation of all packages with insert usage on the altered table. If the added column is the first user-defined structured type column in the table, packages with DELETE usage on the altered table will also be invalidated.
- Adding a check or referential constraint to a table that already exists and that is not in check pending state (see "SET INTEGRITY" on page 1019) will cause the existing rows in the table to be immediately evaluated against the constraint. If the verification fails, an error (SQLSTATE 23512) is raised. If a table is in check pending state, adding a check or referential constraint will not immediately lead to the enforcement of the constraint. Instead, the corresponding constraint type flags used in the check pending operation will be updated. To begin enforcing the constraint, the SET INTEGRITY statement will need to be issued.
- Adding or dropping a check constraint will result in invalidation of all packages with either an insert usage on the object table or an update usage on at least one of the columns involved in the constraint.

61. If the primary or unique key uses an existing unique index that was created in a previous version and has not been converted to support deferred uniqueness, then the index is converted and packages with update usage on the associated table are invalidated.

- Adding a partitioning key will result in invalidation of all packages with an update usage on at least one of the columns of the partitioning key.
- A partitioning key that was defined by default as the first column of the primary key is not affected by dropping the primary key and adding a different primary key.
- Altering a column to increase the length will invalidate all packages that reference the table (directly or indirectly through a referential constraint or trigger) with the altered column.
- Altering a column to increase the length will regenerate views (except typed views) that are dependent on the table. If an error occurs while regenerating a view, an error is returned (SQLSTATE 56098). Any typed views that are dependent on the table are marked inoperative.
- Altering a column to increase the length may cause errors (SQLSTATE 54010) in processing triggers when a statement that would involve the trigger is prepared or bound. This may occur when row length based on the sum of the lengths of the transition variables and transition table columns is too long. If such a trigger were dropped a subsequent attempt to create it would result in an error (SQLSTATE 54040).
- VARCHAR and VARGRAPHIC columns that have been altered to be greater than 4000 and 2000 respectively should not be used as input parameters in functions in the SYSFUN schema (SQLSTATE 22001).
- Changing the LOCKSIZE for a table will result in invalidation of all packages that have a dependency on the altered table. Further information about locking can be found in the *Administration Guide*.
- The ACTIVATE NOT LOGGED INITIALLY clause can not be used when DATALINK columns with the FILE LINK CONTROL attribute are being added to the table (SQLSTATE 42613).
- Changing VOLATILE or NOT VOLATILE CARDINALITY will result in invalidation of all packages that have a dependency on the altered table.
- Replication customers should take caution when increasing the length of VARCHAR columns. The change data table associated with an application table might already be at or near the DB2 rowsize limit. The change data table should be altered before the application table, or the two should be altered within the same unit of work, to ensure that the alteration can be completed for both tables. Consideration should be given for copies, which may also be at or near the rowsize limit, or reside on platforms which lack the feature to increase the length of an existing column.

  If the change data table is not altered before the Capture program processes log records with the increased VARCHAR column length, the Capture program will likely fail. If a copy containing the VARCHAR column is not altered before the subscription maintaining the copy runs, the subscription will likely fail.

Examples

Example 1: Add a new column named RATING, which is one character long, to the DEPARTMENT table.

```
ALTER TABLE DEPARTMENT
 ADD RATING CHAR(1)
```

Example 2: Add a new column named SITE_NOTES to the PROJECT table. Create SITE_NOTES as a varying-length column with a maximum length of 1000 characters. The values of the column do not have an associated character set and therefore should not be translated.

```
ALTER TABLE PROJECT
 ADD SITE_NOTES  VARCHAR(1000) FOR BIT DATA
```

Example 3: Assume a table called EQUIPMENT exists defined with the following columns:

Column Name	Data Type
EQUIP_NO	INT
EQUIP_DESC	VARCHAR(50)
LOCATION	VARCHAR(50)
EQUIP_OWNER	CHAR(3)

Add a referential constraint to the EQUIPMENT table so that the owner (EQUIP_OWNER) must be a department number (DEPTNO) that is present in the DEPARTMENT table. DEPTNO is the primary key of the DEPARTMENT table. If a department is removed from the DEPARTMENT table, the owner (EQUIP_OWNER) values for all equipment owned by that department should become unassigned (or set to null). Give the constraint the name DEPTQUIP.

```
ALTER TABLE EQUIPMENT
 ADD CONSTRAINT DEPTQUIP
  FOREIGN KEY (EQUIP_OWNER)
    REFERENCES DEPARTMENT
     ON DELETE SET NULL
```

Also, an additional column is needed to allow the recording of the quantity associated with this equipment record. Unless otherwise specified, the EQUIP_QTY column should have a value of 1 and must never be null.

```
ALTER TABLE EQUIPMENT
ADD COLUMN EQUIP_QTY
SMALLINT NOT NULL DEFAULT 1
```

Example 4: Alter table EMPLOYEE. Add the check constraint named REVENUE defined so that each employee must make a total of salary and commission greater than $30,000.

```
 ALTER TABLE EMPLOYEE
 ADD CONSTRAINT REVENUE
 CHECK (SALARY + COMM > 30000)
```

Example 5: Alter table EMPLOYEE. Drop the constraint REVENUE which was previously defined.

```
ALTER TABLE EMPLOYEE
DROP CONSTRAINT REVENUE
```

Example 6: Alter a table to log SQL changes in the default format.

```
ALTER TABLE SALARY1
   DATA CAPTURE NONE
```

Example 7: Alter a table to log SQL changes in an expanded format.

```
ALTER TABLE SALARY2
   DATA CAPTURE CHANGES
```

Example 8: Alter the EMPLOYEE table to add 4 new columns with default values.

```
ALTER TABLE EMPLOYEE
ADD COLUMN HEIGHT MEASURE   DEFAULT MEASURE(1)
ADD COLUMN BIRTHDAY BIRTHDATE DEFAULT DATE('01-01-1850')
ADD COLUMN FLAGS BLOB(1M)  DEFAULT BLOB(X'01')
ADD COLUMN PHOTO PICTURE   DEFAULT BLOB(X'00')
```

The default values use various function names when specifying the default. Since MEASURE is a distinct type based on INTEGER, the MEASURE function is used. The HEIGHT column default could have been specified without the function since the source type of MEASURE is not BLOB or a datetime data type. Since BIRTHDATE is a distinct type based on DATE, the DATE function is used (BIRTHDATE cannot be used here). For the FLAGS and PHOTO columns the default is specified using the BLOB function even though PHOTO is a distinct type. To specify a default for BIRTHDAY, FLAGS and PHOTO columns, a function must be used because the type is a BLOB or a distinct type sourced on a BLOB or datetime data type.

Example 9: Assume that you have a table called CUSTOMERS that is defined with the following columns:

Column Name	Data Type
BRANCH_NO	SMALLINT
CUSTOMER_NO	DECIMAL(7)
CUSTOMER_NAME	VARCHAR(50)

In this table, the primary key is made up of the BRANCH_NO and CUSTOMER_NO columns. You want to partition the table, so you need to create a partitioning key for the table. The table must be defined in a table space on a single-node nodegroup. The primary key must be a superset of the partitioning columns: at least one of the columns of the primary key must be used as the partitioning key. Assume that you want to make BRANCH_NO the partitioning key. You would do this with the following statement:

```
ALTER TABLE CUSTOMERS
       ADD PARTITIONING KEY (BRANCH_NO)
```

ALTER TABLESPACE

The ALTER TABLESPACE statement is used to modify an existing tablespace in the following ways.

- Add a container to a DMS tablespace (that is, one created with the MANAGED BY DATABASE option).
- Increase the size of a container in the DMS tablespace (that is, one created with the MANAGED BY DATABASE option)
- Add a container to a SMS tablespace on a partition (or node) that currently has no containers.
- Modify the PREFETCHSIZE setting for a tablespace.
- Modify the BUFFERPOOL used for tables in the tablespace.
- Modify the OVERHEAD setting for a tablespace.
- Modify the TRANSFERRATE setting for a tablespace.

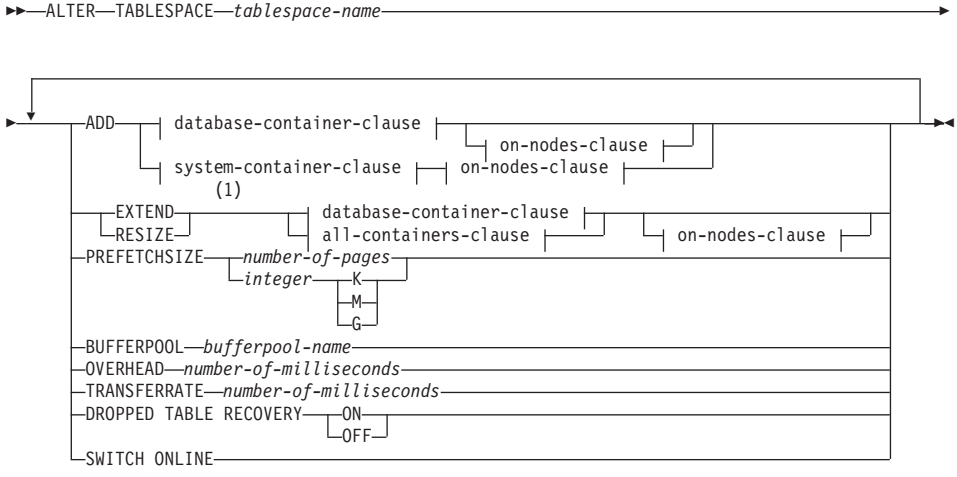
Invocation

This statement can be embedded in an application program or issued interactively. 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).

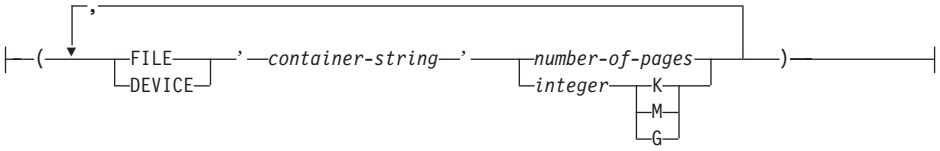
Authorization

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

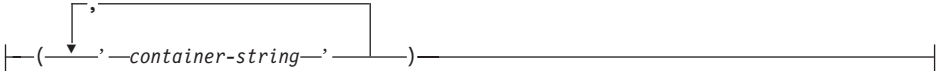
Syntax



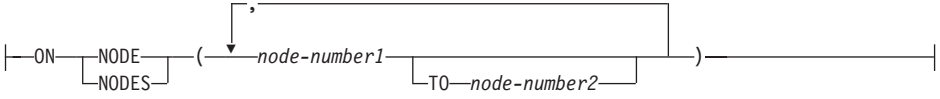
database-container-clause:



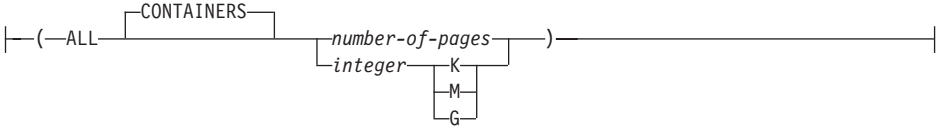
system-container-clause:



on-nodes-clause:



all-containers-clause:



Notes:

1 ADD, EXTEND, and RESIZE clauses cannot be specified in the same statement.

Description

tablespace-name
Names the tablespace. This is a one-part name. It is a long SQL identifier (either ordinary or delimited).

ADD
ADD specifies that a new container is to be added to the tablespace.

EXTEND
EXTEND specifies that existing containers are being increased in size. The

size specified is the size by which the existing container is increased. If the *all-containers-clause* is specified, then all containers in the tablespace will increase by this size.

RESIZE
RESIZE specifies that the size of existing containers is being changed (container sizes can only be increased). The size specified is the new size for the container. If the *all-containers-clause* is specified, then all containers in the tablespace will be changed to this size.

database-container-clause
Adds one or more containers to a DMS tablespace. The tablespace must identify a DMS tablespace that already exists at the application server. See the description of *container-clause* on page 767.

system-container-clause
Adds one or more containers to an SMS tablespace on the specified partitions or nodes. The tablespace must identify an SMS tablespace that already exists at the application server. There must not be any containers on the specified partitions for the tablespace. (SQLSTATE 42921). See the description of *system-containers* on page 767.

on-nodes-clause
Specifies the partition or partitions for the added containers. See the description of *on-nodes-clause* on page 769.

all-containers-clause
Extends or resizes all of the containers in a DMS tablespace. The tablespace must identify a DMS tablespace that already exists at the application server.

PREFETCHSIZE *number-of-pages*
Specifies the number of PAGESIZE pages that will be read from the tablespace when data prefetching is being performed. The prefetch size value can also be specified as an integer value followed by K (for kilobytes), M (for megabytes), or G (for gigabytes). If specified in this way, the floor of the number of bytes divided by the pagesize is used to determine the number of pages value for prefetch size. Prefetching reads in data needed by a query prior to it being referenced by the query, so that the query need not wait for I/O to be performed.

BUFFERPOOL *bufferpool-name*
The name of the buffer pool used for tables in this tablespace. The buffer pool must currently exist in the database (SQLSTATE 42704). The nodegroup of the tablespace must be defined for the bufferpool (SQLSTATE 42735).

OVERHEAD *number-of-milliseconds*
Any numeric literal (integer, decimal, or floating point) that specifies the I/O controller overhead and disk seek and latency time, in milliseconds.

The number should be an average for all containers that belong to the tablespace, if not the same for all containers. This value is used to determine the cost of I/O during query optimization.

TRANSFERRATE *number-of-milliseconds*
Any numeric literal (integer, decimal, or floating point) that specifies the time to read one page (4K or 8K) into memory, in milliseconds. The number should be an average for all containers that belong to the tablespace, if not the same for all containers. This value is used to determine the cost of I/O during query optimization.

DROPPED TABLE RECOVERY
Dropped tables in the specified tablespace may be recovered using the RECOVER DROPPED TABLE ON option of the ROLLFORWARD command.

SWITCH ONLINE
tablespaces in OFFLINE state are brought online if the containers have become accessible. If the containers are not accessible an error is returned (SQLSTATE 57048).

Notes

- Guidance on choosing optimal values for the PREFETCHSIZE, OVERHEAD, and TRANSFERRATE parameters, and information on rebalancing is provided in the *Administration Guide*.
- Once the new container has been added and the transaction is committed, the contents of the tablespace are automatically rebalanced across the containers. Access to the tablespace is not restricted during the rebalancing.
- If the tablespace is in OFFLINE state and the containers have become accessible, the user can disconnect all applications and connect to the database again to bring the tablespace out of OFFLINE state. Alternatively, SWITCH ONLINE option can bring the tablespace up (out of OFFLINE) while the rest of the database is still up and being used.
- If adding more than one container to a tablespace, it is recommended that they be added in the same statement so that the cost of rebalancing is incurred only once. An attempt to add containers to the same tablespace in separate ALTER TABLESPACE statements within a single transaction will result in an error (SQLSTATE 55041).
- A tablespace cannot have container sizes changed and have new containers added in the same ALTER TABLESPACE statement (SQLSTATE 429BC). When changing the size of more than one container, the EXTEND clause and the RESIZE clause cannot be used simultaneously in one statement (SQLSTATE 429BC).
- RESIZE can not be used to decrease container sizes. Any attempt to specify a smaller size for a container will raise an error (SQLSTATE 560B0).

- Any attempts to extend or resize containers that do not exist will raise an error (SQLSTATE 428B2).
- When extending or resizing a container, the container type must match the type that was used when the container was created (SQLSTATE 428B2).
- Once a container has been extended or resized, and the transaction is committed, the contents of the tablespace are automatically rebalanced across the containers. Access to the table space is not restricted during the rebalance.
- If extending multiple containers in a tablespace, it is recommended that the containers be changed in the same statement, so the cost of rebalancing is incurred only once. This is also true for resizing multiple containers. An attempt to change container sizes in the same tablespace, using separate ALTER TABLESPACE statements but within a single transaction, will raise an error (SQLSTATE 55041).
- In a partitioned database if more than one partition resides on the same physical node, then the same device or specific path cannot be specified for such partitions (SQLSTATE 42730). For this environment, either specify a unique *container-string* for each partition or use a relative path name.
- Although the tablespace definition is transactional and the changes to the tablespace definition are reflected in the catalog tables on commit, the buffer pool with the new definition cannot be used until the next time the database is started. The buffer pool in use, when the ALTER TABLESPACE statement was issued, will continue to be used in the interim.

Examples

Example 1: Add a device to the PAYROLL table space.

```
ALTER TABLESPACE PAYROLL
   ADD (DEVICE '/dev/rhdisk9' 10000)
```

Example 2: Change the prefetch size and I/O overhead for the ACCOUNTING table space.

```
ALTER TABLESPACE ACCOUNTING
   PREFETCHSIZE 64
   OVERHEAD 19.3
```

Example 3: Create a tablespace TS1, then resize the containers so that all of the containers have 2000 pages (three different ALTER TABLESPACES which will accomplish this resizing are provided).

```
CREATE TABLESPACE TS1
   MANAGED BY DATABASE
   USING (FILE '/conts/cont0' 1000,
           DEVICE '/dev/rcont1' 500,
           FILE 'cont2' 700)
```

```
ALTER TABLESPACE TS1
   RESIZE (FILE '/conts/cont0' 2000,
            DEVICE '/dev/rcont1' 2000,
            FILE 'cont2' 2000)
```

OR

```
ALTER TABLESPACE TS1
    RESIZE (ALL 2000)
```

OR

```
ALTER TABLESPACE TS1
    EXTEND (FILE '/conts/cont0' 1000,
            DEVICE '/dev/rcont1' 1500,
            FILE 'cont2' 1300)
```

Example 4: Extend all of the containers in the DATA_TS tablespace by 1000 pages.

```
ALTER TABLESPACE DATA_TS
   EXTEND (ALL 1000)
```

Example 5: Resize all of the containers in the INDEX_TS tablespace to 100 megabytes (MB).

```
ALTER TABLESPACE INDEX_TS
   RESIZE (ALL 100 M)
```