SQL STATEMENTS

ALTER DATABASE
ALTER INDEX
ALTER STORGROUP
ALTER TABLE
ALTER TABLESPACE
COMMENT ON
COMMIT
CREATE ALIAS
CREATE DATABASE
CREATE GLOBAL TEMPORARY TABLE
CREATE INDEX
CREATE STORGROUP
CREATE SYNONYM
CREATE TABLE
CREATE TABLESPACE
CREATE VIEW
DECLARE TABLE
DELETE
DESCRIBE
DROP

Figure 3.1 SQL Statements

SQL STATEMENTS (Contd….)

EXPLAIN
GRANT
INCLUDE
INSERT
LABEL ON
RENAME
REVOKE
ROLLBACK
SELECT INTO
SET CURRENT DEGREE
SET CURRENT RULES
SET CURRENT PACKAGESET
SET CURRENT SQLID
SET host variable
UPDATE
WHENEVER

Figure 3.2 SQL Statements (Contd….)

ALTER DATABASE

The ALTER DATABASE statement changes the description of a database at the current server.

Syntax:

```
ALTER DATABASE database-name BUFFERPOOL bpname
                             ROSHARE OWNER or NONE
                             STORGROUP storgroup-name
```

Example:

```
EXEC SQL ALTER DATABASE ABCDE
     BUFFERPOOL BP2
      ROSHARE NONE
END-EXEC.
```

Figure 3.3 ALTER Database

Notes:

Description:

DATABASE database-name: Identifies the database to e altered.

BUFFERPOOL bpname: Identifies the default buffer pool for the table spaces and indexes within the database.

ROSHARE: Indicates whether or not the database is to be shared with other DB2 subsystems using shared read-only data.

OWNER: The database will be shared and the current server will be the DB2 that can update the database.

NONE: The database will not be shared.

STORGROUP storgroup-name: Is the name of the storage group to be used as required as a default storage group to support DASD space requirements for table spaces and indexes within the database.

Example: Change the default buffer pool for database ABCDE to BP2. Also change the ROSHARE option for this database to NONE.

ALTER INDEX

The ALTER INDEX statement changes the description of an index.

Syntax:

```
ALTER INDEX index-name BUFFERPOOL bpname
                       CLOSE YES or NO
                       DSETPASS password
                       PART integer
                       FREESPACE integer
                       PCTFREE integer
                       USING VCAT catalog-name
                       STORGROUP storgroup-name
                       PRIQTY integer
                       SECQTY integer
                       ERASE YES
                             NO
                        GBCACHE CHANGED
                                ALL
                       CONVERT TO TYPE 1
                       PIECESIZE integer
```

Figure 3.4 ALTER Index

ALTER INDEX (Contd….)

Example1:
EXEC SQL ALTER INDEX DSN8510.XEMP1
CLOSE NO
END-EXEC.

Example2:
EXEC SQL ALTER INDEX DSN8510.XPROJ1
BUFFERPOOL BP1
DSETPASS OSESAME
END-EXEC.

Figure 3.5 ALTER Index (Contd….)

Notes:

Description:
Syntax:

Index-name: Identifies the index to be altered.

BUFFERPOOL bpname: Identifies the buffer pool to be used for the index.

CLOSE: Specifies whether the data set is eligible to be closed when the index is not being used and the limit on the number of open data sets is reached. YES – Eligible for closing.
NO – Not eligible for closing.

DSETPASS: Specifies a master level password sent to access method services when the data sets of the index are used by DB2.

PART: Identifies a partition of the index.

FREESPACE: Specifies how often to leave page of free space when index entries are created as the result of executing a DB2 utility.

PCTFREE: Determines the percentage of free space to leave in each nonleaf page and subpage when entries are added to the index or partition as the result of executing a DB2 utility.

USING: Specifies whether the user manages a data set for the index or partition or managed by DB2.

VCAT: Specifies a user-managed data set with a name that starts with the specified catalog name.

STORGROUP: Specifies using a DB2-managed data set that resides on a volume of the specified storage group.

PRIQTY: Specifies the minimum primary space allocation for a DB2-managed data set of the index or partition.

SECQTY: Specifies the minimum secondary space allocation for a DB2-managed data set of the index or partition.

ERASE: Indicates whether the DB2-managed data sets for the index or partition are to be erased when they are deleted during the execution of a utility or an SQL statement that drops the index.

YES – Erase the data sets.

NO – Does not erase the data sets.

GBCACHE: Specifies what index pages are written to the group buffer pool in a data-sharing environment.

CONVERT TO TYPE 1: Specifies changing the type of index.
PIECESIZE: Specifies the maximum piece size for non-partitioned indexes.

Example1: Alter the index DSN8510.XEMP1.CLOSE NO indicates that DB2 is not to close the data sets supporting the index when there are no current users of the index.

Example2: Alter the index DSN8510.XPROJ1.BP1 is the buffer pool to be associated with the index. OSESAME is the password that is passed to VSAM when the data sets are used by DB2.

ALTER TABLE

The ALTER TABLE statement changes the description of a table.

Syntax:
ALTER TABLE table-name
ADD column definition
VALIDPROC program-name or NULL
AUDIT NONE or CHANGES or ALL
ADD PRIMARY KEY (column-name)
ADD check constraint
ADD referential constraint
DATA CAPTURE NONE or CHANGES
DROP PRIMARY KEY
DROP FOREIGN KEY constraint-name
DROP CONSTRAINT constraint-name
DROP CHECK constraint-name
ADD RESTRICT ON DROP
DROP RESTRICT ON DROP

Figure 3.6 ALTER Table

Notes:

Description:

Table-name: Identifies the table to be altered.

ADD column-definition: Adds a column to the table.
Column-name: Is the name of the column that is to be added to the table.
Data-type; Specifies the data type of the column.
NOT NULL: Prevents the column from containing null values.

ALTER TABLE (Contd….)

```
EXEC SQL ALTER TABLE DSN8510.EMP
VALIDPROC NULL
END-EXEC.

EXEC SQL ALTER TABLE DSN8510.EMP
VALIDPROC NULL
END-EXEC.

EXEC SQL ALTER TABLE DSN8510.DEPT
FOREIGN KEY (ADMDEPT) REFERENCES DSN8510.DEPT ON
DELETE CASCADE
END-EXEC.
EXEC SQL ALTER TABLE DSN8510.EMP
ADD CHECK (SALARY >= 10000
END-EXEC.

EXEC SQL ALTER TABLEPRODINFO
FOREIGN KEY (PRODNAME, PRODVER)
REFERENCES PRODVER_1 (VERNAME, RELNO) ON DELETE
RESTRICT
END-EXEC.
```

Figure 3.7 ALTER Table (Contd….)

Notes:

VALIDPROC: Names a validation procedure for the table inhibits the execution of any existing validation procedure.

AUDIT: Alters the auditing attribute of the table.

NONE: Specifies that no auditing be done when the table is accessed.

CHANGES: Specifies that auditing is to be done when the table is accessed during the first inset, update or delete operation performed by each unit of recovery.
ALL: Specifies that auditing is to be done when the table is accessed during first operation of any kind.

PRIMARY KEY: Defines a primary key composed of the identified columns.
ADD check-constraint: Designates the values that specific columns of the table can contain.

DATA CAPTURE: Specifies whether the logging of SQL INSERT, UPDATE, and DELETE operations on the table is augmented by additional information.

DROP PRIMARY KEY: Drops the definition of the primary key and all referential constraint in which the primary key is the parent key.

DROP FOREIGN KEY: Drops the referential constraint, constraint-name.
DROP CONSTRAINT: Drops the constraint constraint-name.

DROP CHECK: Drops the check constraint defined on the table.

ADD RESTRICT ON DROP: Restricts dropping the table and the database and table space that contain the table.

DROP RESTRICT ON DROP: Removes the restriction on dropping the table and the database and table space that contain the table.

Example1: Assigns a validation procedure named DSN8EAEM to the table DSN8510.EMP.

Example2: Disassociate the current validation procedure from the table DSN8510.EMP. After the statement is executed, the table no longer has a validation procedure.

Example3: Define ADMRDEPT as the foreign key of a self-referencing constraint on DSN8510.DEPT.

Example4: Add a check constraint to the table DSN8510.EMP, which checks that the minimum salary an employee can have is $10,000.

Example5: Alter the PRODINFO table to define a foreign key that references a non-primary unique key in the product version table (prodver_1). The columns of the unique key are VERNAME, RELNO.

ALTER TABLESPACE

The ALTER TABLESPACE statement changes the description of a table space.

Syntax:
ALTER TABLE SPACE database-name tablespace-name
BUFFERPOOL bpname
LOCKSIZE
LOCKMAX
CLOSE
DSETPASS password
PART integer
FREEPAGE integer
PICTFREE integer
USING VCAT catalog-name
PRIQTY integer
SECQTY integer
ERASE
COMPRESS
GBCACHE
LOCKPART
MAXROWS integer

Figure 3.8 ALTER Tablespace

Notes:

Description:
Database-name: Identifies the table space to be altered.

BUFFERPOOL: Identifies the buffer pool to be used for the table space.

ALTER TABLESPACE (Contd….)

Example1:
EXEC SQL ALTER TABLESPACE DSN8D51A.DSN8S1D
BUFFERPOOL BP2
LOCKSIZE PAGE
END-EXEC.

Example2:
EXEC SQL ALTER TABLESPACE DSN8D51A.DSN8S51E
CLOSE NO
DSETPASS OSESAME
END-EXEC.

Figure 3.9 ALTER Tablespace (Contd….)

Notes:

LOCKSIZE: Specifies the size of licks used within the table space.

LOCKMAX: Specifies the maximum number of page or rowlocks an application process can hold simultaneously in the table space.

DSETPASS: Specifies a master level password sent to access method services when the data sets of the table space are used by DB2.

PART: Identifies a partition of the table space.

FREEPAGE: Specifies how often to leave a page of free space when the table space is leaded or organized.

PCTFREE: Specifies what percentage of each page to leave as free space when the table space is loaded or organized.
USING: Specifies whether a data set for the table space or partition is managed by the user or managed by DB2.

PRIQTY: Specifies the minimum primary space allocation for a DB2-managed data set of the table space or partition.
SECQTY: Specifies the minimum secondary space allocation for a DB2-managed data set of the table space or partition.

ERASE: Indicates whether the DB2-managed data sets for the table space or partition are to be erased before they are deleted during the execution of a utility or an SQL statement that drops the table space.

COMPRESS: Specifies whether data compression applies to the rows of the table space or partition.

GBCACHE: Specifies what pages of the table space or partition are written to the group buffer pool in a data-sharing environment.

LOCKPART: Indicates whether selective partition locking is to be used when locking a partitioned table space.

MAXROWS: Specifies the maximum number of rows that DB2 will consider placing on each data page.

Example:
Example1: Alter table space DSN8S51D in database DSN8D51A. BP2 is the buffer pool associated with the table space. PAGE is the level at which locking is to take place.

Example2: Alter table space DSN8S51E in database DSN8D51A. CLOSE NO means that the data sets of the table space are not to be closed when there are no current users of the table space. OSESAME is the password that is passed to VSAM when the data sets are used by DB2.

COMMENT ON

The COMMENT ON statement adds or replaces comments in the description of tables, views, aliases or columns in the DB2 catalog.

Syntax:

COMMENT ON
TABLE table-name
ALIAS alias-name
COLUMN table-name or column-name or view-name

Figure 3.10 COMMENT ON

Notes:
Description:
TABLE: Identifies the table or view to which the comment applies.

ALIAS: Identifies the alias to which the comment applies.

COLUMN: Identifies the column to which the comment applies.

COMMENT ON (Contd….)

Example1:

```
EXEC SQL COMMENT ON TABLE DSN8510.EMP
IS 'REFLECTS 1ST QRT 81 REORG'
END-EXEC.
```

Example2:

```
EXEC SQL COMMENT ON TABLE DSN8510.VDEPT
IS 'VIEW OF TABLE DSN8510.DEPT'
END-EXEC.
```

Example3:

```
EXEC SQL COMMENT ON COLUMN
DSN8510.DEPT.DEPTNO
IS 'DEPARTMENT ID – UNIQUE'
END-EXEC.
```

Example4:

```
EXEC SQL COMMENT ON DSN8510.DEPT
(MGRNOIS 'EMPLOYEE NUMBER OF
DEPARTMENT MANAGER', ADMRDEPT IS
'DEPARTMENT NUMBER OF ADMINISTERING
DEPARTMENT')
END-EXEC.
```

Figure 3.11 COMMENT ON (Contd….)

Notes:

Example:

Example1: Enter a comment on table DSN8510.EMP.

Example2: Enter a comment on view DSN8510.VDEPT.

Example3: Enter a comment on the DEPTNO column if table DSN8510.DEPT.

Example4: Enter comments on two columns in table DSN8510.DEPT.

COMMIT

The COMMIT statement ends a unit of recovery and commits the relational database changes that were made in that unit of recovery.

If relational databases are the only recoverable resources used by the application process, COMMIT also ends the unit of work.

Syntax:
COMMIT [WORK]

EXEC SQL COMMIT [WORK] END-EXEC.

Figure 3.12 COMMIT

Notes:

Description:

The unit of recovery in which the statement is executed is ended and a new unit of recovery is effectively started for the process.

All changes made be ALTER, COMMENT ON, CREATE, DELETE, DROP, EXPLAIN, GRANT, REVOKE, INSERT, LABEL ON, RENAME and UPDATE statements executed during the unit of recovery are committed.

Example:

Commit all DB2 database changes made since the unit of recovery was started.

CREATE ALIAS

The CREATE ALIAS statement defines an alias for a table or view. The definition is recorded in the DB2 catalog.

The table or view does not have to be described in that catalog.

Syntax:
CREATE ALIAS alias-name FOR table-name or view-name

Example:

```
EXEC SQL CREATE ALIAS LATABLES FOR
DB2USCALABOA5281.SYSIBM.SYSTABLES
END-EXEC.
```

Figure 3.13 COMMIT (Contd….)

Notes:

Description:

Alias-name: Names the alias.

Example:

Create an alias for a catalog table at a DB2 with location name DB2USCALABOA5281.

CREATE DATABASE

The CREATE DATABASE statement defines a DB2 database.

Syntax:
CREATE DATABASE database-name

BUFFERPOOL bpname
ROSHARE
AS WORKFILE FOR member-name

Example1:
EXEC SQL CREATE DATABASE DSN8G51P
STOGROUP DSN8G510
BUFFERPOOL BP2
END-EXEC.

Example2:
EXEC SQL
CREATE DATABASE DSN8TEMP
END-EXEC.

Figure 3.14 CREATE Database

Notes:

Description:

Database-name: Names the database.

AS WORKFILE: Indicates that this is a work file database. In a non-data-sharing environment, the cause is ignored.

FOR: Specifies the member for which this database is to be a work file.

Example1: Create database DSN8D51P.DSN8G510 is the default storage group to be used for table spaces and indexes in the database. BP2 is the default buffer pool to be used for table spaces and indexes in the database.

Example2: Create database DSN8TEMP. Use the default DB2 storage group and buffer pool and no shared read-only data.

CREATE GLOBAL TEMPORARY TABLE

The CREATE GLOBAL TEMPORARY TABLE statement creates a description of a temporary table.

Syntax:

CREATE GLOBAL TEMPORARY TABLE table-name colspec
LIKE table-name
View-name

Example1:

```
EXEC SQL CREATE GLOBAL TEMPORARY TABLE,
CURRENTMAP
      (CODE INTEGER NOT NULL, MEANING VARCHAR (254)
NOT NULL)
END-EXEC.
```

Example2:

```
EXEC SQL CREATE GLOBAL TEMPORARY TABLE EMP
      (TMPDEPTNO CHAR (3) NOT NULL,
       TMPDEPTNAME VARCHAR (36) NOT NULL,
       TMPMGRNO CHAR (6),
       TMPLOCATION CHAR (16))
END-EXEC.
```

Figure 3.15 CREATE Global Temporary Table

Notes:

Description:

Table-name: Names the temporary table.

Column-spec: Defines the attributes of a column for each instance of the table, the column-name and data-type.

Example1: Create a temporary table, CURRENTMAP. Name two columns, CODE and MEANING, both of which cannot contain nulls. CODE contains numeric data and MEANING has character data. Assuming a value of NO for the field MIXED DATA on installation panel DSNTIPF.

Example2: Create a temporary table, EMP

CREATE INDEX

The CREATE INDEX statement creates a partitioned or nonpartitioned index and an index and index space.

The columns included in the key of the index are columns of a table.

Syntax:
CREATE TYPE I or 2 or UNIQUE INDEX index-name ON table-name

Example1:

```
EXEC SQL CREATE UNIQUE INDEX DSN8510.XDEPT1
ON DSN8510.DEPT
      (DEPTNO ASC)
USING STOGROUP DSN8G510
      ERASE NO
SUBPAGES 8
BUFFERPOOL BP1
CLOSE YES
DSETPASS OSESAME
PIECESIZE 512 M
END-EXEC.
```

Figure 3.16 CREATE Index

Notes:

Description:

Type n: Specifies the type of index, 1 or 2.

UNIQUE: Prevents the table from containing two or more rows with the same value of the index key.

CREATE INDEX (Contd….)

Example2:

```
EXEC SQL CREATE INDEX DSN8510.XEMP2
   ON DSN8510.EMP
      (EMPNO ASC)
   USING STOGROUP DSN8G510
```

```
        PRIQTY 36
        ERASE NO
    SUBPAGES 8
    CLUSTER
        (PART 1 VALUES ('H99'),
         PART 2 VALUES ('P99'),
         PART 3 VALUES ('Z99'),
         PART 4 VALUES ('999'))
    BUFFERPOOL BP1
    CLOSE YES
    DSETPASS OSESAME
END-EXEC.
```

Figure 3.17 CREATE Index (Contd….)

Notes:

INDEX index-name: Names the index.
ON table-name: Identifies the table on which the index is created.
Example1: Create a unique index, named DSN8510.XDEPT1, on table DSN8510.DEPT. Index entries are to be in ascending order by the single column DEPTNO. DB2 is to define the data sets for the index, using storage group DSN8G510; the data sets need not be erased if the index is dropped. Use 8 subpages for each physical page and associate the index with buffer pool BP1. The data sets can be closed when no one is using the index. The VSAM password for the data sets is OSESAME. The maximum for each data set will be 512 megabytes.

CREATE INDEX (Contd….)

Example3:

```
EXEC SQL
        CREATE TYPE 2 UNIQUE INDEX DSN8510.XDEPT1
        ON DSN8510.DEPT
              (DEPTNO ASC)
```

```
        PIECESIZE 1048576 K
END-EXEC.

Example4:
EXEC SQL
        CREATE TYPE 2 UNIQUE INDEX DSN8510.XDEPT1
        ON DSN8510.DEPT
                (DEPTNO ASC)
        PIECESIZE 1G
END-EXEC.
```

Figure 3.18 CREATE Index (Contd….)

Notes:

Example2: Create a cluster index named XEMP2 on table EMP in database DSN8510. Put the entries in ascending order by column EMPNO.
Let DB2 define the data sets for each partition using storage group DSN8G510.
The primary space allocation is 36 KB. If the index is dropped, the data sets need not be erased.
There are to be four partitions, with index entries divided among as follows:

Partition1: entries up to H99.
Partition2: entries above H99 up to P99.
Partition3: entries above P99 up to Z99.
Partition4: entries above Z99.

Use 8 subpages for each physical page, and associate the index with buffer pool BP1. The data sets can be closed when no one is using the index. The VSAM password for the data sets is OSESAME.

Example3: Create a nonpartitioned index in which the individual data sets are no larger than 1 gigabyte. This example shows how to specify the value in KB.

Example4: Create a nonpartitioned index in which the individual data sets are no larger than 1 GB.

CREATE SYNONYM

The CREATE SYNONYM statement defines a synonym for a table or view.

Syntax:
CREATE SYNONYM synonym FOR author-name. Table-name

Description:

Synonym: Names the synonym.

FOR: Identifies the object to which the synonym applies.

Figure 3.19 CREATE SYNONYM

CREATE TABLE

The CREATE TABLE statement defines a table.

The definition must include its name and the names and attributes of its columns. The definition can include other attributes of the table, such as its primary key and its table space.

Syntax:

```
CREATE TABLE table-name   column-definition
                          Unique-constraint
                          Referential-constraint
                          Check-constraint
                  LIKE    Table-name
                          View-name
                  IN      Table-space-name. Database-name.
```

Figure 3.20 CREATE Table

Notes:

Description:

Table-name

Names the table. The name must identify a table, view, alias, or synonym that exists.

Column-definition: Defines the attributes of a column.

Column-name: Names a column of the table.

CREATE TABLE (Contd….)

Example1:

```
EXEC SQL
CREATE TABLE DSN8510.DEPT
     (DEPTNO       CHAR (3)        NOT NULL,
```

```
     DEPTNAME   VARCHAR (36) NOT NULL,
     MGRNO      CHAR (6)               ,
    ADMRDEPT    CHAR (3)     NOT NULL,
    LOCATION    CHAR (16)              ,
    PRIMARY KEY (DEPTNO)               )
    IN DSN8D51A.DSN8S51D
END-EXEC.
```

Figure 3.21 CREATE Table (Contd….)

Notes:

Data-type: Specifies one of the following types:

INTEGER For a large integer
SMALLINT For a small integer
FLOAT For a floating-point number.
DECIMAL For a decimal number, digits range from 1 to 31.
CHARACTER For a fixed-length character string range from 1 to 254.
VARCHAR For a varying-length character string ranging from 1 to 8 bytes.
LONG VARCHAR For a varying-length character string where maximum length is greater than 254.
GRAPHIC For a fixed-length graphic string ranging from 1 to 127.
VARGRAPHIC For a varying-length graphic string longer than 127.
LONG VARGRAPHIC For a varying-length graphic string whose maximum length is determined by the amount of space available in a page.
DATE For a date.

CREATE TABLE (Contd….)

Example2:

```
EXEC SQL
CREATE TABLE DSN8510.PROJ
    (PROJNO     CHAR (6)        NOT NULL,
     PROJNAME   VARCHAR (24)    NOT NULL,
```

```
      DEPTNO      CHAR (3)          NOT NULL,
      RESPEMP     CHAR (6)          NOT NULL,
      PRSTAFF     DECIMAL (5,2)               ,
      PRSDATE     DATE                        ,
      PRENDATE    DATE                        ,
      MAJPROJ     CHAR (6)          NOT NULL)
   IN DATABASE DSN8D51A
END-EXEC.
```

Figure 3.22 CREATE Table (Contd….)

Notes:

TIME For a time.
TIMESTAMP For a timestamp.
PRIMARY KEY Provides a shorthand method of defining a primary key.
UNIQUE Provides shorthand method of defining a unique key composed of a single column.
Unique-constraint:
PRIMARY KEY column-name
Defines a primary key composed of the identified columns. The column should have the definition of NOT NULL. The number of columns identified should not be greater than 64.
UNIQUE column-name
Defines a unique key composed of the identified columns. The column name should be defined as NOT NULL.
The table name specified after REFERENCES must identify a table that exists.

CREATE TABLE (Contd….)

Example3:

```
EXEC SQL
CREATE TABLE ACTIVITY
      (PROJNO      CHAR (6)      NOT NULL,
       ACTNO       SMALLINT      NOT NULL,
```

```
        ACTDEPT     CHAR (3)        NOT NULL,
        ACTOWNER    CHAR (6)        NOT NULL,
        ACSTAFF     DECIMAL (5,2)           ,
        ACSTDATE    DATE            NOT NULL,
        ACENDATE    DATE                    ,
        FOREIGN KEY (ACTDEPT, ACTOWNE)
        REFERENCES PROJECT (DEPTNO, RESPEMP) ON
        DELETE RESTRICT)
    IN DSN8D51A.DSN8S51D
END-EXEC.
```

Figure 3.23 CREATE Table (Contd….)

Notes:

Referential-constraint:

FOREIGN KEY column-name
Each specification of the FOREIGN KEY clause defines a referential constraint with the specified name.

REFERENCE table-name

Check-constraint:

CONSTRAINT constraint-name
Names the table check constraint.
CHECK check-condition
Defines a table check constraint.
LIKE table-name or view-name
Specifies that the columns of the table have exactly the same name and description

as the columns of the identified table or view.

Example1: Create a table named DSN8510.DEPT in the table space DSN8S51D of the database DSN8D51A. Name the table's five column's DEPTNO, DEPTNAME, MGRNO, ADMRDEPT and LOCATION, allowing only MGRNO to contain nulls, and designating DEPTNO as the only column in the table's primary key. All five columns hold character string data.

Example2: Create a table named DSN8510.PROJ in an implicitly created table space of the database DSN8D51A.

Example3: Assume that table PROJECT has a non-primary unique key that consists of columns DEPTNO and DEPTNO and RESPEMP (the department number and employee responsible for a project). Create a project ACTIVITY with a foreign key on that on that unique key.

CREATE TABLESPACE

The CREATE TABLESPACE statement defines a simple, segmented or partitioned table space.

Syntax:

CREATE ┌─────────┐ **TABLESPACE table-space-name IN database-**
└─**LARGE**─┘

Figure 3.24 CREATE Tablespace

Notes:

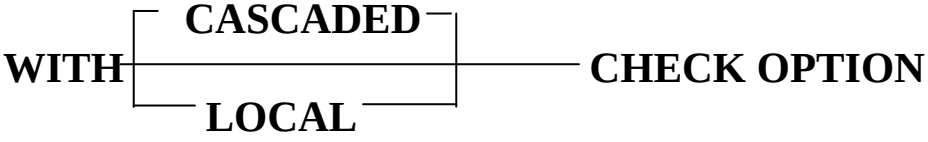
Description:
LARGE: Identifies a table space as large. A large table space is a partitioned table space that can hold more than 64 GB of data, either compressed or uncompressed.

Table-space-name: Names the table space.
IN database-name: Identifies the database in which the table space is created.

CREATE VIEW

The CREATE VIEW statement creates a view on tables or views.

Syntax:
CREATE VIEW view-name AS subselect



Example1:

```
EXEC SQL
CREATE VIEW DSN8510.VPROJRE1,
      (PROJNO, PROJNAME, PROJDEP, RESPEMP,
      FIRSTNME, MIDNIT, LASTNAME)
      AS SELECT ALL PROJNO, PROJNAME, PROJDEP, RESPEMP,
      FIRSTNME, MIDINIT, LASTNAME
      FROM DSN8510.PROJ, DSN8510.EMP
      WHERE RESPEMP = EMPNO
END-EXEC.
```

Figure 3.25 CREATE T View

Notes:
Description:
View-name: Names the view.

AS subselect: Defines the view. View consists of the rows that would result if the subselect were executed.
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.

CASCADED: Update and insert operations on view must satisfy the search conditions of view and all underlying views, regardless of whether the underlying views were defined with a check option.

CREATE VIEW (Contd….)

Example2:

```
CREATE VIEW UNDER AS SELECT *
FROM DSN8510.EMP
WHERE SALARY < 35000
END-EXEC.
```

```
CREATE VIEW OVER AS SELECT *
```

```
FROM UNDER
WHERE SALARY > 30000
WITH LOCAL CHECK OPTION
END-EXEC.
```

Figure 3.26 CREATE T View (Contd….)

Notes:

LOCAL: Update and insert operations on view must satisfy the search condition of view and underlying views that are defined with a check option.

Example1: Create the view DSN8510.VPROJRE1.PROJNO, PROJNAME, PROJDEP, RESPEMP, FIRSTNME, MIDINT and LASTNAME are column names. The view is a join of tables and is therefore read-only. In the example, the WHERE clause refers to the column EMPNO, which is contained in one of the base tables but is not part of the view. In general, a column named in the WHERE, GROUP BY, or HAVING clause need not be part of the view.

Example2: When a view that is defined WITH LOCAL CHECK OPTION is defined on a view that was defined without a check option. Update or insert of row can be done that do not conform to the definition of the view.

DELETE

The DELETE statement deletes rows from a table or view.

Syntax:

```
DELETE FROM table-name or view-name
WHERE search-condition
```

Example:

EXEC SQL DELETE FROM DSN8510.EMP
WHERE WORKDEPT = ‘E11’ OR WORKDEPT = ‘D21’
END-EXEC.

Figure 3.27 DELETE

Notes:

Description:

FROM table-name or view-name

Identifies the object of the DELETE statement.

WHERE: Specifies the rows to be deleted.

Example: From the table DSN8510.EMP delete all rows for departments E11 and D21.

DROP

The DROP statement deletes an object. Any objects that are directly or indirectly dependent on that object are deleted.

Syntax:

DROP ALIAS alias-name
DATABASE database-name
INDEX index-name
SYNONUM synonym

TABLE table-name
TABLESPACE table-space-name
VIEW view-name

Example1: Drop table DSN8510.DEPT
EXEC SQL DROP TABLE DSN8510.DEPT END-EXEC.

Example2: Drop table space DSN8S51D in database DSN8D51A.
EXEC SQL DROP TABLESPACEDSN8D51A.DSN851D
END-EXEC.

Example3: Drop the view DSN8510.VPROJRE1.
EXEC SQL DROP VIEW DSN8510.VPROJRE1 END-EXEC.

Figure 3.28 DROP

Notes:

Description:

ALIAS alias-name

Identifies the alias to be dropped. Dropping an alias has no effect on any view or synonym that was defined using the alias.

DATABASE database-name

Identifies the database to be dropped. The database and all of its table spaces, tables, index spaces and indexes are dropped.

INDEX index-name

Identifies the index to be dropped. The index space is also dropped. If a unique index is dropped and that index was used to enforce the uniqueness of a parent key, the definition of the parent table is changed to incomplete.

SNONYM synonym

Identifies the synonym to be dropped. This has no effect on any view or alias that was defined using the synonym.

TABLE Table-name

Identifies the table to be dropped. All privileges on the table, all referential constraints in which table are a parent or dependent, all synonym, views and indexes defined on the table are also dropped.

TABLESPACE database-name.table-space-name

Identifies the table space to be dropped. All tables in the table space are also dropped.

VIEW view-name

Identifies the view to be dropped. All privileges on the view and all synonyms and views that are defined on the view are also dropped.

GRANT

The GRANT statement grants privileges to authorization IDs.

Syntax:
GRANT authorization-specification TO authorization-name or PUBLIC WITH GRANT OPTION

Example1:
```
EXEC SQL
GRANT SELECT ON DSN8510.EMP TO PULASKI
END-EXEC.
```

Example2:
EXEC SQL
GRANT UPDATE (EMPNO, WORKDEPT) ON TABLE DSN8510.EMP TO PUBLIC
END-EXEC.

Example3:
EXEC SQL
GRANT ALL ON TABLE DSN8510.EMP TO KWAN, THOMPSON WITH GRANT OPTION
END-EXEC.

Figure 3.29 GRANT

Notes:

Description:

Authorization-specification
Names one or more privileges. The same privilege must not be specified more than once.

TO
Specifies to what authorization IDs the privileges are granted.

Authorization-name

Lists one or more authorization IDs.

PUBLIC

Grants the privileges to all users.

WITH GRANT OPTION

Allows the named users to grant the privileges to others.

Example1: Grant SELECT privileges on table DSN8510.EMP to user PULASKI.

Example2: Grant UPDATE privileges on columns EMPNO and WORKDEPT in table DSN8510.EMP to all users.

Example3: Grant all privileges on table DSN8510.EMP to user KWAN and THOMPSON with the WITH GRANT OPTION.

INCLUDE

The INCLUDE statement inserts declarations or code into a source program.

Syntax:
INCLUDE SQLCA
SQLDA
Member-name

Example: Include an SQL communication area in COBOL program.

EXEC SQL INCLUDE SQLCA END-EXEC.

Figure 3.30 INCLUDE

Notes:

Description:

SQLCA

Indicates that the description of an SQL communication area is to be included.
It should not be specified more than once in the same application program.

SQLDA

Indicates that the description of an SQL descriptor is to be included.
It should not be specified for COBOL program.

Member-name

Names a member of the partitioned dataset to be the library input when your application program is precompiled.

INSERT

The INSERT statement inserts rows into a table or view.

There are two forms of INSERT:

The INSERT via VALUES is used to insert a single row in to the table or view using the values provided or referenced.

The INSET via SELECT is used to insert one or more rows in top the table or view using values from other tables and or views.

Syntax:

INSERT INTO table-name or view-name (column-name)
VALUES (constant or host variable or NULL special register)
Subselect

Figure 3.31 INSERT

Notes:

Description:

INTO table-name or view-name

Identifies the object of the INSERT statement.

Column-name: Specifies the columns for which insert values are provided.

VALUES: Specifies one new row in the form of a list of values.

Subselect: Specifies a set of new rows in the form of the result table of a subselect.

INSERT (Contd….)

Example1:

```
EXEC SQL
VALUES ('0205', 'MARY', 'T', 'SMITH', 'D11', '2866',
'1981-08-10', 'ANALYST', 16, 'F', '1995-05-02'.16345, 500, 2300)
END-EXEC.
```

Example2:

```
EXEC SQL
    INSERT INTO SMITH.TEMPEMPL
    SELECT *
```

```
      FROM DSN8510.EMP
END-EXEC.

Example3:
EXEC SQL
INSERT INTO SMITH.TEMPEMP
SELECT *
FROM DSN8510.EMP
WHERE WORKDEPT = 'D11'
END-EXEC.
```

Figure 3.32 INSERT (Contd….)

Notes:

Example1: Insert values in to table DSN8510.EMP

Example2: Populate the temporary table SMITH.TEMPEMPL with data from table DSN8510.EMP
Example3: Populate the temporary table SMITH.TEMPEMPL with data from department D11 from DSN8510.EMP

LABEL ON

The LABEL ON statement adds or replaces labels in the description of tables, views, aliases or columns in the catalog.

Syntax:

```
LABEL ON TABLE table-name or view-name IS string-constant
          ALIAS alias-name
          COLUMN table-name.column-name
                    View-name.column-name
```

Example1:

```
EXEC SQL
LABEL ON COLUMN DSN8510.DEPT.DEPTNO
```

IS ‘DEPARTMENT NUMBER’
END-EXEC.

Example2:
EXEC SQL
LABEL ON DSN8510.DEPT
(MGRNO IS ‘MANAGER’S EMPLOYEE NUMBER’
ADMRDEPT IS ‘ADMINISTERING DEPARTMENT)
END-EXEC.

Figure 3.33 LABEL ON

Notes:

Description:
TABLE: Indicates that the label is for a table or a view.
ALIAS: Identifies the alias to which the comment applies.
COLUMN: Indicates that the label is for a column.
IS: Introduces the label that is to be provided.
String-constant: Any SQL character string constant of up to 30 bytes in length.

Example1: Enter a label on the DEPTNO column of table DSN8510.DEPT.
Example2: Enter labels on two columns in table DSN8510.DEPT.

ENAME

The RENAME statement renames an existing table.

Syntax:
RENAME TABLE source-table-name TO target-identifier

Example:

EXEC SQL
RENAME TABLE EMP TO EMPLOYEE
END-EXEC.

Figure 3.34 RENAME

Notes:

Description:

Source-table-name

Identifies the existing table that is to be renamed.

Target-identifier

Specifies the new name for the table without a qualifier.

Example: Change the name of the EMP table to EMPLOYEE.

REVOKE

The REVOKE statement revokes privileges from authorization IDs.

Syntax:
REVOKE authorization-specification FROM authorization-name
PUBLIC
BY authorization-name
ALL

Figure 3.35 REVOKE

Notes:

Description:

Authorization-specification

Names one or more privileges.

FROM

Specifies from what authorization IDs the privileges are revoked.

Authorization-name

Lists one or more authorization IDs.

PUBLIC

Revokes a grant of privileges to PUBLIC.

REVOKE (Contd….)

Example1:
EXEC SQL
REVOKE SELECT ON TABLE DSN8510.EMP FROM PULASKI
END-EXEC.

Example2:
EXEC SQL
REVOKE IUPDATE ON TABLE DSN8510.EMP FROM PUBLIC
END-EXEC.

Example3:
EXEC SQL
REVOKE ALL ON TABLE DSN8510.EMP FROM KWAN, THOMPSON
END-EXEC.

Figure 3.36 REVOKE (Contd….)

Notes:

BY
Lists grantors who have granted privileges and revokes each named privilege that was explicitly granted to some named user by one of the named grantors.

Authorization-name
Lists one or more authorization IDs of users who were the grantors of the privileges named.

ALL
Revokes each named privilege from all named users who were explicitly granted the privilege, regardless of who granted it.

Example1: Revoke SELECT privileges on table DSN8510.EMP from user PULASKI.

Example2: Revile update privileges on table DSN8510.EMP previously granted to all local DB2 users.

Example3: Revoke all privileges on table DSN8510.EMP from users KWAN and THOMPSON.

ROLLBACK

The ROLLBACK statement ends a unit of recovery and backs out the relational database changes that were made by that unit of recovery.

Syntax:
ROLLBACK WORK

Example:

EXEC SQL ROLLBACK WORK END-EXEC.

Figure 3.37 ROLLBACK

Notes:

Description:

The unit of recovery in which the ROLLBACK statement is executed is ended and a new unit of recovery is effectively started.
All changes made by ALTER, COMMENT ON ,CREATE, DELETE, DROP, GRANT, INSERT, LABEL ON, RENAME, REVOLE and UPDATE statements executed during the unit of recovery are backed out.

Example: Rollback all DB2 database changes made since the unit of recovery was started.

SELECT INTO

The SELECT INTO statement produces a result table containing at most one row, and assigns the values in that row to host variables. If the table is empty, the statement assigns +100 to SQLCODE.

Syntax:
Select-clause- INTO host variable from clause where clause

Example1:

```
EXEC SQL
SELECT MAX (SALARY)
INTO :MAXSALARY
FROM DSN8510.EMP
END-EXEC.
```

Figure 3.38 SELECT INTO

Notes:

Description:

The table is derived by evaluating the from-clause, where-clause and select-clause, in this order.

INTO host-variable

Each host-variable must identify a structure or variable that is described in the program in accordance with the rules for declaring host structures and variables.

SELECT INTO (Contd….)

Example2:

```
EXEC SQL
SELECT * INTO :EMPREC
FROM DSN8510.EMP
WHERE EMPNO = ‘528671’
END-EXEC.
```

Example3:

```
EXEC SQL
SELECT CURRENT DATE + 20 DAYS
INTO :DUEDATE
FROM SYSIBM.SYSDUMMY1
```

END-EXEC.

Figure 3.39 SELECT INTO (Contd….)

Notes:

Example1: Put the maximum salary in DSN8510.EMP in to the host variable MAXSALARY.

Example2: Put the row for employee 528671 from DSN8510.EMP, into the host structure EMPREC.

Example3: Put a date that is twenty days from the current date into host variable DUEDATE. Use DB2 catalog table SYSIBM.SYSDUMMY1 as the table referenced.

SET CURRENT DEGREE

The SET CURRENT DEGREE statement assigns a value to the CURRENT DEGREE special register.

Syntax:
SET CURRENT DEGREE string constant or host variable

Example1: The following statement inhibits parallel operations:
SET CURRENT DEGREE = '1'

Example2: The following statement allows parallel operations:
SET CURRENT DEGREE = 'ANY'

Figure 3.40 SET CURRENT DEGREE

Notes:

Description:

The value of the CURRENT DEGREE is replaced by the value of the string constant or host variable.

SET CURRENT RULES

The SET CURRENT RULES statement assigns a value to the CURRENT RULES special register.

Syntax:
SET CURRENT RULES string-constant or host-variable

Example: Set SQL ruled to be followed to DB2.
EXEC SQL SET CURRENT RULES = ‘DB2’ END-EXEC.

Figure 3.41 SET CURRENT RULES

Notes:

Description:
This statement replaces the value of the CURRENT RULES special register with the value of the string constant or host variable.

The value must be a character string 3 bytes in length and the value must be ‘DB2’ or ‘STD’. An error occurs if any other values are specified.

SET CURRENT SQLID

The SET CURRENT SQLID statement assigns a value to the CURRENT SQLID special register.

Syntax:

SET CURRENT SQLID USER or string-constant or host-variable

Example:
Set the CURRENT SQLID to the primary authorization ID.

SET CURRENT SQLID = USER

Figure 3.42 SET CURRENT SQLID

Notes:

Description:

The value of CURRENT SQLID is replaced by the value of USER, string-constant or host-variable.

The value specified by a string-constant or host-variable must be a character string that is not longer than 8 bytes.

SET host-variable

The SET host-variable statement assigns the value of a special register to a host variable.

Syntax:
SET host-variable = register

Example: Set the host variable XTIME to the local time.
EXEC SQL SET :XTIME = CURRENT TIME END-EXEC.

Figure 3.43 SET host-variable

Notes:

Description:

Host-variable: Identifies the host variable to which the value of the special register is assigned. The reference must not include an indicator variable.

Special-register: Identifies the special register whose value is placed in the host-variable.

WHENEVER

The WHENEVER statement specifies the host language statement to be executed when a specified exception condition occurs.

Syntax:

```
WHENEVER NOT FOUND      CONTINUE
         SQLERROR       GO TO host-label
         SQLWARNING     GO TO
```

Figure 3.44 WHENEVER

Notes:

Description: The NOT FOUND, SQLERROR or SQLWARNING clause is used to identify the type of exception condition.

NOT FOUND: Identifies any condition that results in an SQLCODE of +100.

SQLERROR: Identifies any condition that results in a negative SQLCODE.

SQLWARNING: Identifies any condition that results in a warning condition or that results in positive SQLCODE other than +100.

The CONTINUE or GO TO clause specifies the next statement to be executed when the identified type pf exception condition exists.

CONTINUE: Specifies the next sequential statement of the source program.

GOTO: Specifies the statement identified by host-label.

WHENEVER (Contd….)

Example1:

```
EXEC SQL
WHENEVER SQLERROR TOTO HANDLER
END-EXEC.
```

Example2:

```
EXEC SQL
WHENEVER SQLWARNING CONTINUE
END-EXEC.
```

Example3:

```
EXEC SQL
WHENEVER NOT FOUND GO TO ENDDATA
END-EXEC.
```

Figure 3.45 WHENEVER (Contd….)

Notes:

Example1: Go to the label HANDLER for any statement that produces an error.

Example2: Continue processing for any statement that produces a warning.

Example3: Go to the label ENDDATA for any statement that does not return.