APPENDIX E
Glossary of Common SQL Commands

*

The asterisk character returns all the columns of a particular table.

ALTER DATABASE

```
ALTER DATABASE database_name;
```

The `ALTER DATABASE` statement changes the size or settings of a database. Its syntax varies widely among different database systems.

ALTER USER

```
ALTER USER user
```

The `ALTER USER` statement changes a user's system settings such as a password.

BEGIN TRANSACTION

```
1> BEGIN TRANSACTION transaction_name
2> transaction type
3> if exists
4> begin
```

The `BEGIN TRANSACTION` statement signifies the beginning of a user transaction. A transaction ends when it is either committed (see `COMMIT TRANSACTION`) or canceled (see `ROLLBACK TRANSACTION`). A transaction is a logical unit of work.

CLOSE CURSOR

```
close cursor_name
```

The `CLOSE cursor_name` statement closes the cursor and clears it of data. To completely remove the cursor, use the `DEALLOCATE CURSOR` statement.

COMMIT TRANSACTION

```
COMMIT;
```

The `COMMIT TRANSACTION` statement saves all work since the beginning of the transaction (since the `BEGIN TRANSACTION` statement was executed).

CREATE DATABASE

```
CREATE DATABASE database_name;
```

The `CREATE DATABASE` statement creates a new database. Many different options can be supplied, such as the device on which to create the database and the size of the initial database.

CREATE INDEX

```
CREATE INDEX index_name
ON table_name(column_name1, [column_name2], ...);
```

An index can order the contents of a table based on the contents of the indexed field(s).

CREATE PROCEDURE

```
CREATCCREATE PROCEDURE procedure_name
   [[(]@parameter_name
     datatype [(length) | (precision [, scale])
     [= default][output]
   [, @parameter_name
     datatype [(length) | (precision [, scale])
     [= default][output]]...[)]]
   [with recompile]
   AS SQL_statements
```

The `CREATE PROCEDURE` statement creates a new stored procedure in the database. This stored procedure can consist of SQL statements and can then be executed using the `EXECUTE` command. Stored procedures support the passing of input and output parameters and can return an integer value for status checking.

CREATE TABLE

```
CREATE TABLE table_name
(   field1 datatype [ NOT NULL ],
   field2 datatype [ NOT NULL ],
   field3 datatype [ NOT NULL ]...)
```

The `CREATE TABLE` statement creates a new table within a database. Each optional field is provided with a name and data type for creation within that table.

CREATE TRIGGER

```
CREATE TRIGGER trigger_name
  ON table_name
  FOR {insert, update, delete}
  AS SQL_Statements
```

The CREATE TRIGGER statement creates a trigger object in the database that will execute its SQL statements when its corresponding table is modified through an INSERT, UPDATE, or DELETE. Triggers can also call stored procedures to execute complex tasks.

CREATE USER

```
CREATE USER user
```

The CREATE USER statement creates a new user account, complete with user ID and password.

CREATE VIEW

```
CREATE VIEW <view_name> [(column1, column2...)] AS
SELECT <table_name column_names>
FROM <table_name>
```

A view is often described as a virtual table. Views are created by using the CREATE VIEW statement. After a view is created, it can be queried, and data within the view can be modified.

DEALLOCATE CURSOR

```
DEALLOCATE CURSOR cursor_name
```

The DEALLOCATE CURSOR statement completely removes the cursor from memory and frees the name for use by another cursor. You should always close the cursor with the CLOSE CURSOR statement before deallocating it.

DECLARE CURSOR

```
DECLARE cursor_name CURSOR
   FOR select_statement
```

The DECLARE CURSOR statement creates a new cursor from the SELECT statement query. The FETCH statement scrolls the cursor through the data until the variables have been loaded. Then the cursor scrolls to the next record.

DROP DATABASE

```
DROP DATABASE database_name;
```

The DROP DATABASE statement completely deletes a database, including all data and the database's physical structure on disk.

DROP INDEX

```
DROP INDEX index_name;
```

The DROP INDEX statement removes an index from a table.

DROP PROCEDURE

```
DROP PROCEDURE procedure_name
```

The DROP PROCEDURE statement drops a stored procedure from the database; its function is similar to the DROP TABLE and DROP INDEX statements.

DROP TABLE

```
DROP TABLE table_name;
```

The DROP TABLE statement drops a table from a database.

DROP TRIGGER

```
DROP TRIGGER trigger_name
```

The DROP TRIGGER statement removes a trigger from a database.

DROP VIEW

```
DROP VIEW view_name;
```

The DROP VIEW statement removes a view from a database.

EXECUTE

```
EXECUTE [@return_status = ]
   procedure_name
   [[@parameter_name =] value |
     [@parameter_name =] @variable [output]...]]
```

The EXECUTE command runs a stored procedure and its associated SQL statements. Parameters can be passed to the stored procedure, and data can be returned in these parameters if the OUTPUT keyword is used.

FETCH

```
FETCH cursor_name [into fetch_target_list]
```

The FETCH command loads the contents of the cursor's data into the provided program variables. After the variables have been loaded, the cursor scrolls to the next record.

FROM

```
FROM <tableref> [, <tableref> ...]
```

The FROM statement specifies which tables are used and/or joined.

GRANT

```
GRANT role TO user
```

or

```
GRANT system_privilege TO {user_name | role | PUBLIC}
```

The GRANT command grants a privilege or role to a user who has been created using the CREATE USER command.

GROUP BY

```
GROUP BY <col> [, <col> ...]
```

The GROUP BY statement groups all the rows with the same column value.

HAVING

```
HAVING <search_cond>
```

The HAVING statement is valid only with GROUP BY and limits the selection of groups to those that satisfy the search condition.

INTERSECT

```
INTERSECT
```

The INTERSECT statement returns all the common elements of two SELECT statements.

ORDER BY

```
ORDER BY <order_list>
```

The ORDER BY statement orders the returned values by the specified column(s).

REVOKE

```
REVOKE role FROM user;
```

or

```
REVOKE {object_priv | ALL [PRIVILEGES]}
[, {object_priv | ALL [PRIVILEGES]} ] ...
ON [schema.]object
FROM {user | role | PUBLIC} [, {user | role | PUBLIC}] ...
```

The `REVOKE` command removes a database privilege from a user, whether it be a system privilege or a role.

ROLLBACK TRANSACTION

```
ROLLBACK TRANSACTIONROLLBAKC;
```

The `ROLLBACK TRANSACTION` statement effectively cancels all work done within a transaction (since the `BEGIN TRANSACTION` statement was executed).

SELECT

```
SELECT [DISTINCT | ALL]
```

The `SELECT` statement is the beginning of each data retrieval statement. The modifier `DISTINCT` specifies unique values and prevents duplicates. `ALL` is the default and allows duplicates.

SET TRANSACTION

```
SET TRANSACTION (READ ONLY | USE ROLLBACK SEGMENT);
```

The `SET TRANSACTION` command enables the user to specify when a transaction should begin. The `READ ONLY` option locks a set of records until the transaction ends to ensure that the data is not changed.

UNION

```
UNION
```

The `UNION` statement returns all the elements of two `SELECT` statements.

WHERE

```
WHERE <search_cond>
```

The `WHERE` statement limits the rows retrieved to those meeting the search condition.