SQL Reference

Learning Gupta SQLBase Structured Query Language (SQL)

By William Sprouse

CENTER FOR ECOLOGICAL MANAGEMENT OF MILITARY LANDS
Department of Forest Sciences
Colorado State University
Fort Collins, CO 80523

Spring 1996
### Table of Contents

**PREFACE**  
1  

**PURPOSE AND SCOPE**  
1  
**AUDIENCE**  
1  
**ORGANIZATION**  
1  
**DOCUMENT CONVENTIONS**  
1  
**ASSOCIATED DOCUMENTS**  
2  
**SYSTEM REQUIREMENTS**  
2  

1. **INTRODUCTION**  
4  

2. **ORIENTATION**  
4  

2.1 **OPERATING BASICS**  
4  

2.1.1 **INSTALLING QUEST**  
4  

2.1.2 **FUNDAMENTALS OF THE WINDOWS ENVIRONMENT**  
4  

2.1.2.1 **Menus**  
5  

2.1.2.2 **The Control Menu**  
5  

2.1.2.3 **Moving and Resizing Windows**  
5  

2.1.2.4 **The Clipboard**  
5  

2.2 **GETTING STARTED**  
5  

2.2.1 **STARTING QUEST**  
5  

2.2.2 **GETTING HELP**  
6  

3. **SQL BASICS**  
6  

3.1 **INTRODUCTION**  
6  

3.2 **SYNTAX DIAGRAMS**  
7  

3.3 **SQLBASE SQL FUNCTIONS**  
7  

3.4 **THE SELECT STATEMENT**  
8  

3.5 **THE UPDATE STATEMENT**  
9  

3.6 **THE INSERT STATEMENT**  
11  

3.7 **THE DELETE STATEMENT**  
12  

3.8 **THE DROP STATEMENT**  
13  

3.9 **THE CREATE VIEW STATEMENT**  
13  

3.10 **THE CHECK DATABASE STATEMENT**  
14  

4. **ADVANCED SQL STATEMENTS**  
15  

4.1 **INTRODUCTION**  
15  

4.2 **SELECTING DATA FROM MORE THAN ONE TABLE (JOINING)**  
15  

4.3 **GROUPING DATA**  
16  

4.4 **USING COLLECTIONS (IN AND NOT IN STATEMENTS)**  
17  

4.5 **THE UNION STATEMENT**  
18
Preface

Purpose and Scope

This manual is designed to give the users of Gupta Structured Query Language (SQL) an easy to understand reference. This manual is specifically targeted to users of Land Condition - Trend Analysis (LCTA). All of the examples within this manual are specific to the LCTA data. An orientation section will explain using Quest and its SQL functionality.

Audience

This tutorial assumes you are familiar with the Microsoft Windows environment and the methods and requirements of LCTA data collection. If you are not familiar with the Windows environment refer to the Microsoft Windows documentation. You may also want to review the Windows tutorial before proceeding further with this tutorial. You can find more information on the methods and requirements of the LCTA data collection in U.S. Army Land Condition - Trend Analysis (LCTA) Plot Inventory Field Method and LCTA Programs User’s Guide.

Organization

This document is divided into four major components, as follows:

1. Introduction
   A brief explanation of SQL.

2. Orientation
   Provides basic information on Gupta’s Quest SQL features.

3. SQL Basics
   Provides a reference for the basic SQL statements and examples.

4. Advanced SQL Statements
   Enhances the basic SQL statements.

Document Conventions

This manual uses the following syntax conventions to describe user interaction with the program.

- italic text written between the brackets < > is text that is entered at the computer
- normal text written between the brackets <> are menu options
**Associated Documents**

The following documents cover information not provided in this tutorial.

- **LCTA Programs User's Guide**  
  Detailed information on the LCTA programs.
- **U.S. Army Land Condition - Trend Analysis (LCTA) Plot Inventory Field Method**  
  Field methods and data collection standards of LCTA.
- **Microsoft Windows User's Guide**  
  Microsoft Windows reference.
- **Quest Advanced User’s Guide**  
  Gupta’s Quest reference.
- **Quest Starter Guide**  
  Gupta’s Quest Getting Started Guide.
- **SQLBase/SQLTalk Language Reference Manual**  
  SQL command reference for SQLBase.

**System Requirements**

Hardware and software requirements to use Quest and its SQL features include:

**Software**

1. MS DOS 5.0 or higher
2. Windows 3.x (Microsoft)
3. SQLBase 5.1.x (Gupta Tech. Inc.)
4. Quest (Gupta Tech. Inc.)

**Hardware**

1. 386 or 486 PC
2. 4M Ram required, 8M recommended
1. Introduction

The LCTA Program Manager offers many analyses and utilities to summarize the LCTA data, however; these analyses only provide standardized summaries. In order to summarize the LCTA data beyond the LCTA Program Manager and maintain the database Structured Query Language (SQL) use is required. Although Quest does allow for generating queries through the use of a point-and-click tool, this can become time consuming. Once the basic SQL statements are understood summarizing and maintaining the data will become easier and faster.

2. Orientation

This section teaches the basic skills needed to get Quest started, connected to a database, and configured to enter SQL commands.

2.1 Operating Basics

This section helps you install the Quest program and reviews some fundamentals of the Windows operating system.

2.1.1 Installing Quest

The installation of Quest is covered in the Quest manuals provided with the program. This section will cover those operations needed for versions of Quest earlier than 2.x.

In version 1.2.3 of Quest the SQL functionality was provided but not automatically installed. To install the SQL functionality open the QUEST.INI file, found in the directory where Quest was installed, in any text editor (NOTEPAD.EXE). Near the top of the file a number of lines containing the statement Activity?=???. After the last such entry add the statement ActivityX=Sql.dll where X is the next sequential number from the last activity statement. Save and close the QUEST.INI file and start Quest. A SQL button will now appear in the Quest toolbar.

Versions of Quest before 3.0 came with a single-tasking database engine. This engine is used by Quest to communicate with the database file. The single-tasking engine only allows one task to be performed on a database at a time. Elements of the LCTA Program Manager require a multi-tasking engine, therefore you may have purchased the SQLBase for Windows engine. If this is the case you need to make sure the products are installed in the correct order. Install Quest first then the SQLBase for Windows program. If the products are installed in the correct order and errors are reported when starting Quest contact either Gupta or your LCTA Support Center.

2.1.2 Fundamentals of the Windows Environment

If you are familiar with the Microsoft Windows environment you may want to skip to section 2.2 Getting Started. If you have never used Windows you should familiarize yourself with the Windows environment by looking through the Microsoft Windows manuals and running the Windows tutorial.
To start Windows type <WIN> at the DOS prompt. When Windows opens you will be looking at the Program Manager. The Program manager contains groups which contain program icons. Double click on a group icon to open the group, double click on a program icon to start the program.
All Windows programs will have similar characteristics. The use of the mouse, closing programs, moving and resizing windows are common across all applications. All Windows programs will have a title bar, menu bar, and work area.

2.1.2.1 Menus

Each menu item may contain commands that can not be seen on the menu bar. To display these commands single click on a menu item, if commands exists they will appear in a drop down menu. Single click on one of the commands to run that option.

2.1.2.2 The Control Menu

Must Windows programs will have a control menu, indicated by a small rectangle in the upper left corner of the window. Single click on this rectangle to display the control menu. These options will consist of options to move or resize the window, switch to another application that is active, and close the program. A short cut trick for closing an application is to double click on the control menu button.

2.1.2.3 Moving and Resizing Windows

Windows can me moved or resized by using the commands in the control menu or by using mouse techniques. To move the window place the cursor in the title bar and hold down the left mouse button. You may then drag the window to a new location. If the window takes up the full screen you may change the size by using the arrow buttons in the upper right corner of the window. The down arrow will minimize the application, close it to an icon and place it at the bottom of the screen. The up/down arrow button will reduce the size of the window. Once the window size is reduced, not filling the entire screen, you can use the mouse to resize the window. Place the cursor over one of the edges of the window. When the cursor is placed properly the cursor will change to a double arrow. Hold down the left mouse button and drag the edge to the desired size.

2.1.2.4 The Clipboard

The clipboard is a temporary buffer used to store text and graphics that have been copied from an application. When you copy text or graphics from one application to past into another, the clipboard stores the image.

2.2 Getting Started

This section covers the basics for starting Quest and opening the SQL window. You should be familiar with the concepts discussed above before going on with this section. The SQL Basics section will discuss SQL in detail.

2.2.1 Starting Quest

To begin a session in Quest double click the Quest icon, select <Database>, then <Add>. From the Add Database dialog box select the desired database name and single click the OK button.
The Quest window will have both a main menu and a toolbar. To open the SQL window select the NEW button to the left of the toolbar. Select the SQL button and an empty SQL window will open. It is here that the SQL commands are typed. To save SQL statements select <File> from the main menu, <Save>, <SQL>, and enter a name for the file. To open an existing SQL statement select the OPEN button to the left of the toolbar then the SQL button, select the desired file name.

### 2.2.2 Getting Help

You may view the on-line help at any time by pressing the F1 function key. Help can also be accessed from the <Help> menu item.

## 3. SQL Basics

### 3.1 Introduction

Before reading through this section you should have the Quest software and your database installed and have a working understanding of the Windows environment. You should also look at the orientation section of this manual. It will familiarize you with the look and feel of Quest.

This SQL Basics section will cover the basic SQL concepts and nomenclature. The Advanced SQL section will add upon the knowledge learned here to create more complicated SQL statements.
3.2 Syntax Diagrams

This manual will use the following syntax diagrams for SQL statements.

- normal text between the brackets [] are optional SQL components
- normal text between the braces {} contains a list of required elements, only one of the elements listed will be required
- italics text represent a generic place holder, for example, tablename is used to show a table name should be entered at that location
- bold text are key words

Each SQL statement covered in this manual will list the general SQL syntax, an LCTA example, and the English meaning of the statement. Following is an example of the SQL syntax diagrams.

**Syntax Diagram**

```sql
SELECT [distinct] {columnlist, *} FROM tablename
[WHERE condition [AND condition]]
[ORDER BY columnlist];
```

**LCTA Example**

```sql
SELECT plotid, recdate, landuse FROM landuse
WHERE plotid < 300
ORDER BY plotid;
```

**English Translation**

Select the plot number, recording date, and landuse from the table landuse where the plot number is less than 300 and sort the list by plotid.

In this example the syntax diagram shows the general statement syntax. Notice the keywords SELECT, FROM, WHERE, AND, ORDER BY. The word distinct is an optional word in this statement. Columnlist represents a place holder for a list of column names separated by commas, tablename is the place holder for the name of the table, and condition is the place holder for the where condition. Also note that not only is the WHERE clause optional, but the Where clause has an optional AND clause.

As mentioned earlier, this section will cover only the basic SQL statements. In the Advanced SQL section the SQL syntax diagrams will be expanded to incorporate the added functionality of the statements.

3.3 SQLBase SQL Functions

SQLBase utilizes many build in functions to help retrieve the data that is needed. Listed here are some of the more commonly used and useful functions in basic SQL statements.

- **@YEARNO(columnname)**
  
  Retrieve only the year part of a date column
• @UPPER(columnname)
  Convert a column to uppercase (SQLBase is case sensitive, this function can be used to ensure a match in
  a WHERE clause)

Examples of these functions will be used in the following sections.

### 3.4 The SELECT Statement

A SELECT statement is used to retrieve information from one or more tables. When data is retrieved from more
than one table a join is performed, this will be covered in the Advanced SQL section.

**Syntax Diagram**

```sql
SELECT [distinct] {columnlist,*} FROM tablename
[WHERE condition [AND condition]]
[ORDER BY columnlist];
```

**LCTA Example 1**

```sql
SELECT * FROM landuse;
```

**LCTA Example 2**

```sql
SELECT distinct plotid,recdate,landuse FROM landuse
WHERE plotid < 300
ORDER BY plotid;
```

**LCTA Example 3**

```sql
SELECT plotid,@ YEARNO(recdate),landuse FROM landuse
WHERE plotid < 300 and @UPPER(landuse) = ‘TRACKED’ and @YEARNO(recdate) > 1990
ORDER BY 1,2;
```

**English Translation 1**

Select all columns from the table landuse.

**English Translation 2**

Select only unique values of plot number, recording date, and landuse from the table landuse
where the plot number is less than 300, sort the list by plotid.

**English Translation 3**

Select the plot number, year portion of the recording date, and landuse from the table landuse
where the plot number is less than 300 and the upper case landuse value is equal to TRACKED and the year
portion of the recording date is greater than 1990,
sort the list by plotid and the year portion of the recording date.

Notice that in example 3 the ORDER BY clause uses 1,2 not plotid,@YEARNO(recdate). This is a short hand
method of listing the variables in SQLBase. The numbers correspond to the list of columns following the
SELECT key word. When using a function such as @YEARNO(recdate) in the column list you must use this short had method.

### 3.5 The **UPDATE** Statement

This statement is used to change values of existing data in the database.

**Syntax Diagram**

```
UPDATE tablename
SET columnname = expression
[WHERE condition];
```

**LCTA Example 1**

```
UPDATE plotsurv
SET plottype = 'C';
```

**LCTA Example 2**

```
UPDATE plotsurv
SET plottype = 'S'
WHERE @UPPER(plottype) = 'SPUSE';
```

**LCTA Example 3**

```
UPDATE landuse
SET landuse = 'TRACKED'
WHERE @UPPER(landuse) = 'TRACK';
```

**English Translation 1**

Change all the values for plot type in the table plotsurv to ‘C’.

**English Translation 2**

Change the plot type of plots in the table plotsurv to ‘S’ where the upper case value of existing plot types is equal to ‘SPUSE’.

**English Translation 3**

Change the landuse value in the table landuse to ‘TRACKED’ where the upper case value of existing landuse is equal to ‘TRACK’.

The update statement is very useful when editing your data. Quest does not allow editing returned values in a query or SQL window. This means you have two alternatives to correct the data in the database, the update statement or manually by opening the table and scrolling through the data to find the incorrect values.
If you have a number of changes to make you can enter all of the update statements in one SQL window, each ending with a semicolon, and choose <SQL>, <Execute All> from the main menu. Quest will execute each update statement that is properly written in the SQL window.
3.6 The INSERT Statement

The INSERT statement will insert data into a defined table in the database. Because it is usually easier to use the table window and the insert or paste from functions to copy large amounts of data to a table this discussion is limited. For information on these tasks consult the Quest User’s Guide. The INSERT statement discussed here will show you how to extract data from one table and enter it into another. This statement will use a subselect which is covered further in the Advanced SQL section.

For this statement to work the number of columns and type of data extracted from one table must match that of the receiving table.

Syntax Diagram

```
INSERT INTO tablename
subselect;
```

LCTA Example 1

```
INSERT INTO mytable
select plotid,plottype from plotsurv;
```

LCTA Example 2

```
INSERT INTO mytable2
select distinct plotid from plotsurv;
```

LCTA Example 3

```
INSERT INTO mytable3
select distinct plotid, @YEARNO(recdate) from plotsurv where @UPPER(plottype) = ‘C’;
```

English Translation 1

Extract plot number and plot type from the table plotsurv and insert the values into the table mytable which has plotid and plottype columns defined.

English Translation 2

Extract unique plot numbers from the table plotsurv and insert the values into the table mytable2 which has a plotid column defined.

English Translation 3

Extract unique values of plotid and year part of recording date from the table plotsurv where the upper case value of plot type is equal to ‘C’ and insert the values into the table mytable3 which has two columns defined.
3.7 The **DELETE** Statement

The **DELETE** statement is used to delete data from a table. Be careful when using this command, you may delete more data than intended.

**Syntax Diagram**

```sql
DELETE FROM tablename
[WHERE condition];
```

**LCTA Example 1**

DELETE FROM landuse;

**LCTA Example 2**

DELETE FROM landuse
WHERE plotid = 3;

**LCTA Example 3**

DELETE FROM landuse
WHERE plotid = 3 and @YEARNO(recdate) = 1990;

**English Translation 1**

Delete all data in the table landuse.

**English Translation 2**

Delete data from the table landuse where the plotid is equal to 3.

**English Translation 3**

Delete data from the table landuse where the plotid is equal to 3 and the year part of the recording data is equal to 1990.
3.8 The DROP Statement

The DROP statement is used to drop a table from the database. Be careful when using this command, the table and all data will be removed from the database. DROP can also be used to drop views and indexes, consult the Quest User’s Guide for more information. This section will only discuss using the DROP statement to remove tables from the database.

Syntax Diagram

DROP tablename;

LCTA Example 1

DROP landuse;

LCTA Example 2

DROP mytable;

English Translation 1

Drop the table landuse

English Translation 2

Drop the table mytable

3.9 The CREATE VIEW Statement

The CREATE VIEW statement can be used to create a view on one or more tables. If a view is created from only one table the data in the view can be modified. Views are useful for showing only certain information from a table that can then be updated. Views are also useful for storing the joining of information from two tables, this is similar to saving a query or SQL statement. The difference between a saved query or SQL statement and a view is that a view can be treated as a table and used in other SQL statements. When a view is created the data is not stored in the database as another table, only the SQL statement that creates the view is stored. This is useful because every time the view is opened the data is retrieved from the appropriate tables, ensuring that the view is always up-to-date.

Once a view is created you open the view by selecting the OPEN and TABLE buttons, the view will be listed with the other tables.

Syntax Diagram

CREATE VIEW view-name [columnnames]
AS SELECT selectstatement;

View-name is a name you give for the view. The optional columnnames can be used to give different names to the columns of the view, different from the column names specified in the SELECT statement. The selectstatement is
any valid select statement. Only basic select statements are shown here but the Advanced SQL section will show
you how to write more advanced select statements.

**LCTA Example 1**

CREATE VIEW myview
AS SELECT plotid,plottype FROM plotsurv;

**LCTA Example 2**

CREATE VIEW myview2 PlotNum,CoreOrSpUse
AS SELECT plotid,plottype FROM plotsurv;

**English Translation 1**

Create a view named myview containing plot number and plot type from the table plotsurv.

**English Translation 2**

Create a view named myview2 containing plot number and plot type, name these PlotNum and CoreOrSpuse
respectively, from the table plotsurv.

### 3.10 The **CHECK DATABASE** Statement

The **CHECK** database statement performs integrity checks on the entire database. This statement is useful for
ensuring your database does not contain a problem which can cause errors or loss of data.

**Syntax Diagram**

CHECK DATABASE;

**LCTA Example 1**

CHECK DATABASE;

**English Translation 1**

Check the integrity of the entire database.

This command may take a while to run, when Quest has completed the check and no errors are found “Ready” will
appear in the bottom left corner of the Quest window. If errors are found they will be reported on the screen. Most
of the error statements are fairly cryptic, for help with errors contact either Gupta’s technical support or your
LCTA Support Center.
4. Advanced SQL Statements

4.1 Introduction

In the previous section basic commonly used SQL statements were covered. In this section more advanced SQL statements will be explained. All of the advanced SQL statements discussed here utilize the **SELECT** statement and further the functionality of this command.

4.2 Selecting Data From More Than One Table (Joining)

Using a **SELECT** statement that joins two or more tables can be very useful when looking at your data. Many times you may want to look at data from two tables and compare the values, by joining tables you can output the data in columnar form and save data transfer time.

**Syntax Diagram**

```
SELECT [distinct] {columnlist,*} FROM tablenames
[WHERE condition [AND condition]]
[ORDER BY columnlist];
```

The only difference between this statement and the basic **SELECT** statement in the previous section is the *columnlist, tablenames* and the **WHERE** conditions. When joining tables you may need to identify column names by the table they are found in. You do this by using the syntax `tablename.columnname`, for example `plotsurv.plotid`. This is only needed if the same column exists in more than one table you are joining, however; it is good practice to use this syntax to avoid errors. In the **FROM** clause of the **SELECT** statement there is a list of tables separated by commas. The where condition must have at least one join statement. A join statement sets two columns of the same name in two tables to be equal. You cannot join tables that do not have at least one column in common.

**LCTA Example 1**

```
SELECT plotsurv.plotid,plotsurv.plottype,landuse.landuse FROM plotsurv,landuse
WHERE plotsurv.plotid=landuse.plotid;
```

**LCTA Example 2**

```
SELECT plotsurv.plotid,plotsurv.plottype,landuse.landuse FROM plotsurv,landuse
WHERE plotsurv.plotid=landuse.plotid
AND @UPPER(plotsurv.plottype) = 'C'
AND @YEARNO(plotsurv.recdate) = 1990
ORDER BY plotsurv.plotid;
```

**LCTA Example 3**

```
SELECT pcsdplotsum.plotid,(pcsdplotsum.gcbare*100/pcsdplotsum.gcobs) FROM plotsurv,pcsdplotsum
WHERE plotsurv.plotid=pcsdplotsum.plotid
AND @UPPER(plotsurv.plottype) = 'C'
AND @YEARNO(plotsurv.recdate) = pcsdplotsum.analyear;
```
**English Translation 1**

Select plot number from the table plotsurv, plot type from the table plotsurv, and landuse from the table landuse where plot numbers from the tables plotsurv and landuse are equal.

**English Translation 2**

Select plot number from the table plotsurv, plot type from the table plotsurv, and landuse from the table landuse where plot numbers from the tables plotsurv and landuse are equal and the upper case value of plot type is equal to ‘C’ and the year part of the recording date from the plotsurv table is equal to 1990, sort the information by plot number.

**English Translation 3**

Select plot number from the table pcsdplotsum and the average bare ground value from the table pcsdplotsum where the plot numbers from the tables plotsurv and pcsdplotsum are equal and the upper case value of the plot type is equal to ‘C’ and the year part of the recording date from the plotsurv table is equal to analyear from the pcsdplotsum table.

Example 3 uses a mathematical expression to calculate the average bare ground per plot from the data summary table pcsdplotsum. When you run the Plant Cover Surface Disturbance (PCSD) analysis from the LCTA Program Manager summary data is stored in the table pcsdplotsum for each plot. Gcbare is the number of bare hits on the ground for a plot and gcobs is the number of total hits on the ground for that plot. Because both of these values are integers dividing them returns the value of an integer divide, 0. In order to force SQLBase to perform a non-integer divide the numerator is multiplied by 100, the value returned will be the percent bare ground.

### 4.3 Grouping Data

Grouping data is useful when trying to combine values in a larger unit other than plots, for example year. Use the GROUP BY clause for certain functions such as sum, average, or other mathematical expressions.

**Syntax Diagram**

```
SELECT [distinct] {columnlist,*} FROM tablenames
[WHERE condition [AND condition]]
GROUP BY columnlist
[ORDER BY columnlist];
```

**LCTA Example 1**

```
SELECT @YEARNO(recdate),COUNT(plottype) FROM plotsurv
GROUP BY 1;
```

**LCTA Example 2**

```
SELECT @YEARNO(recdate),COUNT(plottype) FROM plotsurv
WHERE plotid < 300
GROUP BY 1;
```
LCTA Example 3

```
SELECT pcsdplotsum.analyear, AVG(pcsdplotsum.gcbare*100/pcsdplotsum.gcobs) AS AverageBG FROM plotsurv, pcsdplotsum
WHERE plotsurv.plotid=pcsdplotsum.plotid AND @UPPER(plottype) = 'C' AND pcsdplotsum.gcobs <> 0
GROUP BY pcsdplotsum.analyear;
```

**English Translation 1**

Select the year part of the recording data and count the occurrence of each plottype from the table plotsurv, group the counting of plot type by the year.

**English Translation 2**

Select the year part of the recording data and count the occurrence of each plottype from the table plotsurv where the plot number is less than 300, group the counting of plot type by the year.

**English Translation 3**

Select plot number from the table pcsdplotsum and the average of mean bare ground from the table pcsdplotsum where the plot numbers from the tables plotsurv and pcsdplotsum are equal and the upper case value of the plot type is equal to ‘C’ and the value of gcobs is not equal to 0, group the average by the analysis year.

Notice that in examples 1 and 2 the GROUP BY clause uses 1 not @YEARNO(recdate). This is a short hand method of listing the variables in SQLBase. The number corresponds to the list of columns following the SELECT key word. When using a function such as @YEARNO(recdate) in the column list you must use this short had method.

Also notice that example 3 uses the average function (AVG). Essentially this is a mean of a mean. The where clause of this example contains the line “AND pcsdplotsum.gcobs <> 0”, this ensures that there is no division by 0. Example 3 also takes advantage of the AS clause to rename the output column of the average bare ground, this is for display purposes only.

### 4.4 Using Collections (IN and NOT IN Statements)

The IN predicate is used to compare a value to a collection of values. The collections of values can be listed in the SQL statement or can be the result of a subselect. A subselect is a SELECT statement nested within a SELECT statement.

Comparing values to a collection of values is useful for extracting data contained in a particular set, or not in a set. The IN or NOT IN predicate could be used to find the existence of unknown vegetation codes, see the examples below.

**Syntax Diagram**

```
SELECT [distinct] {columnlist,*} FROM tablename
WHERE expression [NOT] IN {subselect.listofvalues} [AND condition]]
```
[ORDER BY columnlist];

LCTA Example 1

SELECT distinct vegid FROM aercover
WHERE @UPPER(vegid) NOT IN (SELECT @UPPER(vegid) FROM plntlist);

LCTA Example 2

SELECT plotid.landuse FROM landuse
WHERE @UPPER(landuse) IN ('TRACKED', 'WHEELED');

LCTA Example 3

SELECT gndcover.plotid,gndcover.vegid FROM gndcover,plotsurv
WHERE plotsurv.plotid = gndcover.plotid
AND @UPPER(plotsurv.invtype) = 'I'
AND @UPPER(plotsurv.plottype) = 'C'
AND @UPPER(gndcover.vegid) IN (SELECT @UPPER(vegid) FROM plntlist)
ORDER BY gndcover.plotid,gndcover.vegid;

English Translation 1

Select unique values of vegetation codes from the table aercover where the upper case of the vegetation code is in the set of upper case vegetation codes taken from the table plntlist (i.e. known vegetation codes from the table aercover).

English Translation 2

Select plot number and landuse values from the table landuse where the value landuse is equal to ‘TRACKED’ or ‘WHEELED’.

English Translation 3

Select plot number and vegetation code from the table gndcover where the plot numbers in the tables gndcover and plotsurv are equal and the upper case value of the inventory type is equal to ‘I’ and the upper case value of the plot type is equal to ‘C’ and the upper case value of the vegetation code in the table gndcover is in the set of values of upper case vegetation codes from the table plntlist (i.e. plotid and known vegetation codes from gndcover for initial inventory measurements that are core plots).

4.5 The Union Statement

The UNION clause is used to merge the results of two or more SELECT statements. Duplicate rows will be eliminated unless the ALL qualifier is used. You cannot use UNION in a view. The number of columns and data types must be the same in each select statement.

UNION is useful when you want to retrieve a specific set of data that requires a number of SQL statements into one output. In some cases you could use OR in the WHERE condition to get the same results.
Syntax Diagram

```sql
SELECT [distinct] {columnlist,*} FROM tablename 
[WHERE condition [AND condition]]
[ORDER BY columnlist]
UNION
SELECT [distinct] {columnlist,*} FROM tablename 
[WHERE condition [AND condition]]
[ORDER BY columnlist];
```

LCTA Example 1

```sql
SELECT plotid,landuse FROM landuse 
WHERE plotid = 1 
UNION
SELECT plotid,landuse FROM landuse 
WHERE plotid = 10;
```

LCTA Example 2

```sql
SELECT 'TRAIL' AS Dist,(gdtrail*100/gdobs) FROM pcsdplotsum 
WHERE analyear = 1989 
UNION
SELECT 'PASS',(gdpass*100/gdobs) FROM pcsdplotsum 
WHERE analyear = 1989 
UNION
SELECT 'OTHERTYPE',(gdother*100/gdobs) FROM pcsdplotsum 
WHERE analyear = 1989 
ORDER BY 1;
```

English Translation 1

Append the results of select plot number and landuse from the table landuse where plot number is 1 and plot number is 10.

English Translation 2

Append the results of a defined value Dist and the mean trail, pass, and other disturbances from the table pcsdplotsum where the analysis year is equal to 1989.

Example 2 contains some of the minor details for the `UNION` clause to watch for. The defined value for Dist, which is manually defined, uses the `AS` key word to rename the output column. The `AS` key word is only used in the first `SELECT` statement. You must also make sure in this case that the length of the first defined variable Dist is as long or longer than the longest string defined, note the blank spaces at the end of TRAIL. Also notice that in example 2 the `ORDER BY` clause uses 1 not Dist. This is a short hand method of listing the variables in SQLBase. The number corresponds to the list of columns following the `SELECT` key word. The short hand method is required in `UNION` clauses.
Example 2 uses a mathematical expression to calculate the average trail, pass, and other disturbances per plot from the data summary table pcsdplotsum. When you run the Plant Cover Surface Disturbance (PCSD) analysis from the LCTA Program Manager summary data is stored in the table pcsdplotsum for each plot. Gdpass, gdtrail and gdother are the number of pass, trail, and other disturbance hits on the ground for a plot and gdobs is the number of total disturbance hits on the ground for that plot. Because all of these values are integers dividing them returns the value of an integer divide, 0. In order to force SQLBase to perform a noninteger divide the numerator is multiplied by 100, the value returned will be the percent bare ground.
References


