900 Series HP 3000 Computer Systems

Getting Started with HP IMAGE/SQL

HEWLETT PACKARD

Printed in U.S.A. December 1994

Second Edition E1294
Customer Order Number 36385-90008

Printing History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time this document was issued. Many product releases do not require changes to the document. Therefore, do not expect a one-to-one correspondence between product releases and document editions.

Edition	Date	Software Version
First Edition	June 1993	36385-B.F0.20
Second Edition	December 1994	36385-B.G0.03

Preface

This book describes how to start using HP IMAGE/SQL.

HP IMAGE/SQL is offered on HP 3000 computers using the MPE/iX operating system. MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX.

Note

Hereafter in this manual, *HP IMAGE/SQL* will be referred to as *IMAGE/SQL.*

The following briefly describes each chapter in this manual:

Chapter 1	**Introducing IMAGE/SQL** Introduces IMAGE/SQL.
Chapter 2	**Basic Concepts** Presents basic ideas, tasks, and concepts.
Chapter 3	**Moving From TurboIMAGE/XL to IMAGE/SQL** For the user coming from the world of TurboIMAGE/XL to IMAGE/SQL, describes comparative terminology and discusses steps to using IMAGE/SQL.
Chapter 4	**A Database Example** Presents a simple database example.
Chapter 5	**IMAGE/SQL Tasks** Shows these IMAGE/SQL tasks: setting up a database with ISQL, configuring a DBEnvironment, attaching a database, detaching a database, restructuring a database, and restoring an IMAGE/SQL environment.
Chapter 6	**Practicing with IMAGE/SQL Using MusicDBE** Details the steps for creating a TurboIMAGE/XL database and setting up MusicDBE, the IMAGE/SQL sample DBEnvironment.
Appendix A	**Tools For IMAGE/SQL** Lists a selection of available tools.
Glossary	**Glossary of IMAGE/SQL Terms** Gives basic definitions of terms.

Additional Documentation

Refer to the listed manuals for additional information:

- *HP IMAGE/SQL Administration Guide* (36385-90001)

  This guide explains how to administer and maintain IMAGE/SQL. For this manual, you should be familiar with and have a general knowledge of relational databases.

- *TurboIMAGE/XL Database Management System Reference Manual* (30391-90001)

  This manual describes the TurboIMAGE/XL Database Management System for the HP 3000 Series 900 computer. It is the reference document for anyone designing, creating, and maintaining a database and for application programmers writing database access programs.

- *ISQL Reference Manual for ALLBASE/SQL and IMAGE/SQL* (36216-90096)

  This manual describes ISQL (Interactive Structured Query Language) and how to use it.

- *ALLBASE/NET User's Guide* (36216-90031)

  This guide describes ALLBASE/NET, a product that allows an application on MPE/iX to access an ALLBASE/SQL DBEnvironment on another MPE/iX system. This guide is written for the system administrator, programmers, and users of ALLBASE/SQL DBEnvironments.

- *ALLBASE/SQL Database Administration Guide* (36216-90005)

  This manual describes how to design, create, and maintain ALLBASE/SQL databases on HP 3000 computers. This guide is written for experienced users of SQL and SQL application programmers.

- *ALLBASE/SQL Message Manual* (36216-90009)

  This manual explains the causes and actions to be taken for warning and error conditions that occur using the following components of ALLBASE/SQL: SQL, ISQL, SQLAudit, SQLGEN, SQLUtil, the preprocessors, SQLMigrate, NET, or NETUtil. This manual is for database administrators, application programmers, and others who are using ALLBASE/SQL. This manual assumes that the user has a basic understanding of databases and ALLBASE/SQL concepts.

- *ALLBASE/SQL Performance and Monitoring Guidelines* (36216-90102)

  This is a technical manual for database administrators to assist in monitoring the performance of ALLBASE/SQL applications.

- *ALLBASE/SQL Reference Manual* (36216-90001)

This manual presents the syntax and semantics of SQL on HP 3000 computers. This manual contains basic information about ALLBASE/SQL, as well as in-depth information about ALLBASE/SQL data types and statements. The first three chapters are for all readers, including new users of ALLBASE/SQL. The remaining chapters are for experienced SQL users and SQL application programmers.

- *HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL* (36216-90104)

  This guide describes HP PC API for ALLBASE/SQL and IMAGE/SQL, how to install it, and how to maintain it. This guide is for the system administrator installing HP PC API for ALLBASE/SQL and IMAGE/SQL and for someone using application software on the PC client.

- *Up and Running with ALLBASE/SQL* (36389-90011)

  This is a getting started reference for ALLBASE/SQL.

Conventions

`UPPERCASE`	In a syntax statement, commands and keywords are shown in uppercase characters. The characters must be entered in the order shown; however, you can enter the characters in either uppercase or lowercase. For example: `COMMAND` can be entered as any of the following: `command` `Command` `COMMAND`
italics	In a syntax statement or an example, a word in italics represents a parameter or argument that you must replace with the actual value. In the following example, you must replace *FileName* with the name of the file: `COMMAND` *FileName*
punctuation	In a syntax statement, punctuation characters (other than brackets, braces, vertical bars, and ellipses) must be entered exactly as shown. In the following example, the parentheses and colon must be entered: `(`*FileName*`):(`*FileName*`)`
underlining	Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, yes is the user's response to the prompt: `Do you want to continue? >> yes`
`{ }`	In a syntax statement, braces enclose required elements. When several elements are stacked within braces, you must select one. In the following example, you must select either ON or OFF: $\texttt{COMMAND}\begin{Bmatrix}\texttt{ON}\\\texttt{OFF}\end{Bmatrix}$
...	In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.

Conventions (continued)

`[ ]` In a syntax statement, brackets enclose optional elements. In the following example, OPTION can be omitted:

COMMAND *FileName* [OPTION]

When several elements are stacked within brackets, you can select one or none of the elements. In the following example, you can select OPTION or *Parameter* or neither. The elements cannot be repeated.

$$\texttt{COMMAND}\ \textit{FileName}\ \begin{bmatrix}\texttt{OPTION}\\ \textit{Parameter}\end{bmatrix}$$

`[ ... ]` In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear within the immediately preceding pair of brackets or braces. In the example below, you can select *Parameter* zero or more times. Each instance of *Parameter* must be preceded by a comma:

[,*Parameter*] [...]

In the example below, you only use the comma as a delimiter if *Parameter* is repeated; no comma is used before the first occurrence of *Parameter*:

[*Parameter*] [,...]

`| ... |` In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select A, AB, BA, or B. The elements cannot be repeated.

$$\begin{Bmatrix}\texttt{A}\\ \texttt{B}\end{Bmatrix} | \ldots |$$

$\boxed{}$ The symbol $\boxed{}$ indicates a key on the keyboard. For example, $\boxed{\text{RETURN}}$ represents the carriage return key.

$\boxed{\text{CTRL}}$*char* $\boxed{\text{CTRL}}$*char* indicates a control character. For example, $\boxed{\text{CTRL}}$Y means that you press the control key and the Y key simultaneously.

Contents

Figures

Tables

1

Introducing IMAGE/SQL

This chapter introduces IMAGE/SQL:

- What is IMAGE/SQL?
- What are the components of IMAGE/SQL?
- What are the benefits of using IMAGE/SQL?
- How do IMAGE/SQL and ALLBASE/SQL coexist?

What Is IMAGE/SQL?

IMAGE/SQL is one of Hewlett-Packard's relational database management systems. IMAGE/SQL provides relational access to your TurboIMAGE data using the industry-standard Structured Query Language (SQL). This access method includes full read and write capability using ANSI standard functionality. Closely tuned to the architecture of HP computers, IMAGE/SQL gives you flexibility in designing and using SQL database applications on a small or large scale.

Figure 1-1 shows how IMAGE/SQL, TurboIMAGE/XL, and ALLBASE/SQL are related.

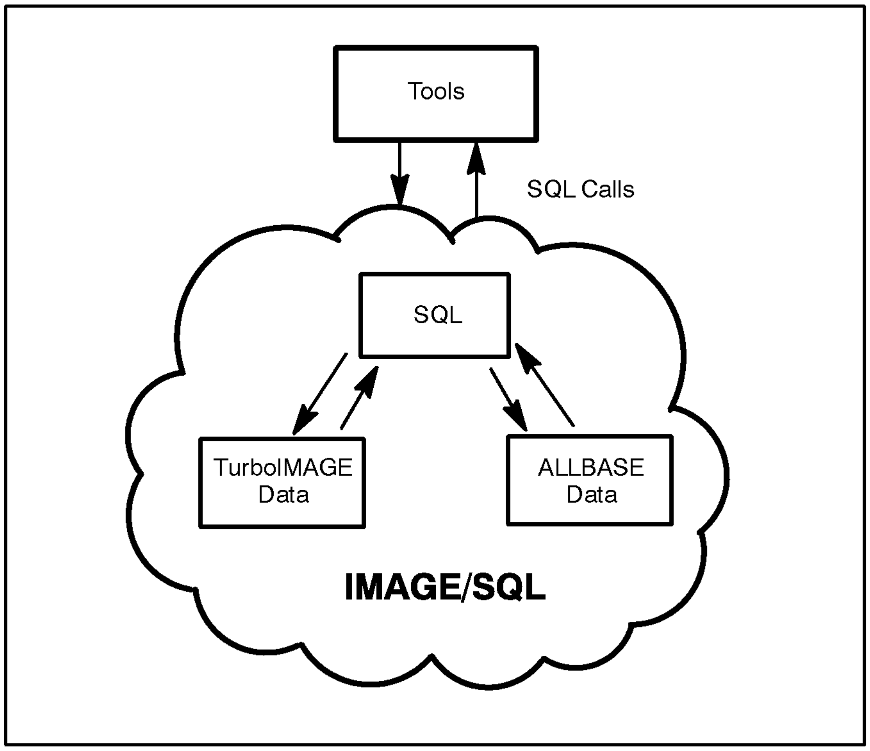


Figure 1-1. Overview of IMAGE/SQL

What Are the Components of IMAGE/SQL?

IMAGE/SQL includes TurboIMAGE/XL, certain components of ALLBASE/SQL, and a database administration tool that links them together. The database administration tool is the IMAGE/SQL utility. The components of IMAGE/SQL are described below:

TurboIMAGE/XL

A set of programs and procedures for defining, creating, accessing, and maintaining TurboIMAGE/XL databases. TurboIMAGE/XL databases can be accessed using TurboIMAGE/XL procedures, utilities, and applications. IMAGE/SQL adds the ability to access TurboIMAGE/XL databases by using the industry-standard SQL language.

ALLBASE/SQL

A set of programs and procedures for creating, maintaining, and accessing relational database environments. IMAGE/SQL includes the ALLBASE/SQL components needed to provide relational access to TurboIMAGE/XL databases.

HP PC API for ALLBASE/SQL and IMAGE/SQL

Software that provides a programmatic interface to IMAGE/SQL for creating Microsoft windows-based client-server applications. With HP PC API for ALLBASE/SQL and IMAGE/SQL, you use one of several available 4GL and decision support tools to build your application, which connects to ALLBASE/SQL over a network through ALLBASE/NET.

ALLBASE/NET

Works with HP PC API for ALLBASE/SQL and IMAGE/SQL in connecting users with databases over a network.

IMAGE/SQL Utility

A **database administrator's** tool for managing TurboIMAGE/XL databases in a relational access environment. This tool attaches and detaches databases to and from DBEnvironments for access via SQL. Other management functions, such as security and data type mapping, are accomplished through this utility. Refer to the *HP IMAGE/SQL Administration Guide* for details on the IMAGE/SQL utility.

ISQL

An interactive command processor that lets you enter SQL statements at the keyboard and observe query results, messages, and other information on a video display. SQL is a language—not a software system. Therefore, you need an interactive way to submit SQL statements.

ISQL is the main tool used by IMAGE/SQL programmers and database administrators to create and modify DBEnvironments. It is also used by anyone who needs to submit queries using the SQL language. ISQL is especially useful for loading and unloading data.

SQLGEN A **database administrator's** tool that generates the commands used to recreate all or part of an existing DBEnvironment. It also generates LOAD/UNLOAD and UPDATE STATISTICS statements. These commands are placed in one or more data files called **schema** files, which can then be used as ISQL command files to recreate the DBEnvironment (not the TurboIMAGE/XL database.)

SQLUtil A **database administrator's** tool for displaying and setting the basic parameters of a DBEnvironment (explained further in Chapter 2, "Basic Concepts"), storing and restoring DBEnvironments, setting the size of system buffers, and purging DBEnvironments. The database administrator (DBA) is the individual who creates and maintains objects in a DBEnvironment. **SQLUtil** is seldom needed by the ordinary user.

Preprocessors Tools that convert source programs containing SQL statements into source code that can be compiled in a programming language of your choice. Different preprocessors let you code applications in C, COBOL, FORTRAN, and Pascal.

What Are The Benefits of Using IMAGE/SQL?

Using IMAGE/SQL, you benefit in the following ways:

- By leveraging your investment—

  Your existing investment in TurboIMAGE/XL is preserved while you take advantage of relational technology. You can develop new applications in SQL to access TurboIMAGE/XL databases through IMAGE/SQL. At the same time, your current TurboIMAGE/XL access is completely compatible with the SQL access to TurboIMAGE/XL. Therefore, you will not have to convert, recompile, or make changes to your existing applications. The only difference is the new application development and decision support tools available for looking at your data.

- By accessing new tools—

  Using IMAGE/SQL allows you to use a wide selection of application development and decision support tools not previously available to TurboIMAGE/XL users. Hewlett-Packard and independent software vendors have a wide selection of SQL client/server tools. These SQL client/server tools allow you to view data in new ways by combining data from TurboIMAGE/XL, ALLBASE/SQL, or third-party databases in the same query. See Appendix A for more information on the available tools.

- By benefiting from productivity gains—

  SQL uses a common interface to access TurboIMAGE/XL, ALLBASE/SQL, and third-party SQL databases. Productivity improves when you can use the same interface to access both types of databases.

How Do IMAGE/SQL and ALLBASE/SQL Coexist?

Hewlett-Packard now offers a complete range of data management choices. TurboIMAGE/XL still provides the highest performance in the industry for mission-critical OLTP business applications. ALLBASE/SQL provides the highest SQL performance of any relational database in the industry and provides support for distributed database and distributed transaction processing. IMAGE/SQL provides data access through the many client/server tools at a small performance premium. Client/server computing is clearly an emerging trend in information processing and IMAGE/SQL is now an important element, ensuring the protection of your information investment.

Together, these database management systems make a powerful team. ALLBASE/SQL and IMAGE/SQL can be linked together within a single environment. Your applications, whether host-based or client/server, can simultaneously access ALLBASE/SQL and IMAGE/SQL information.

2

Basic Concepts

This chapter familiarizes you with these basic ideas, tasks, and concepts before venturing on to IMAGE/SQL tasks:

- What is a database?
- What is a relational database?
- What is SQL?
- What is TurboIMAGE/XL?
- What is a DBEnvironment?
- How do I create a DBEnvironment?
- How do I create a database?
- How do I access a database through SQL?
- How do I control database access?

What Is a Database?

A **database** is a structured arrangement of data elements designed for the easy selection of information. Unlike a collection of flat files, a database contains both data and structural information used in extracting data from the files in which data resides.

What Is a Relational Database?

A relational database is a collection of data arranged in **table**s, also known as **relation**s. Tables are subject to the following **relational operations**, each of which lets you retrieve data in a specific way:

- **Selection**, which lets you extract a subset of rows.
- **Projection**, which lets you extract a subset of columns.
- **Joining**, which lets you extract from more than one table at a time.

In practice, these operations frequently appear together. An SQL statement that uses these operations is known as a **query**. Three queries that use the SQL SELECT statement to illustrate selection, projection, and joining are shown in Figure 2-1.

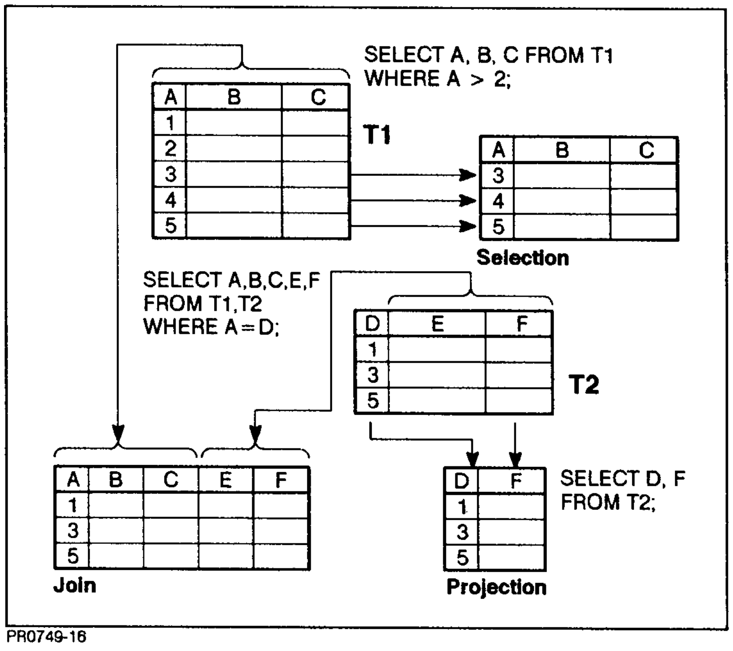


Figure 2-1. Relational Operations

Rows and Columns

When you look at data in relational terms, you can assume several things:

- Tables are arranged in **row**s and **column**s, which are like records and fields in an ordinary file.
- Each column has a specific data type and size.
- Each row contains one element for every column.

Sample Database Table

The following is a portion of a database table consisting of names and account balances for an employee credit union:

Employee Accounts

Last Name	**First Name**	**Telephone**	**Employee Number**	**Balance**
Harrison	Gerald	7233	2432099	142.59
Abelson	Annette	4312	3510044	2345.09
Stanley	Peter	1235	3540011	321.98
Walters	Georgia	2554	9124772	1230.10

Each column can accept data of a specific type and size. Refer to Chapter 3, "Moving from TurboIMAGE/XL to IMAGE/SQL," for more details on data types.

Using Several Tables

You can put the same data into several different tables, as shown below:

Table 1. Employees Table

Last Name	**First Name**	**Employee Number**
Harrison	Gerald	2432099
Abelson	Annette	3510044
Stanley	Peter	3540011
Walters	Georgia	9124772

Table 2. Telephone Table

Last Name	**First Name**	**Telephone**
Harrison	Gerald	7233
Abelson	Annette	4312
Stanley	Peter	1235
Walters	Georgia	2554

Table 3. Accounts Table

Employee Number	**Account Balance**
2432099	142.59
3510044	2345.09
3540011	321.98
9124772	1230.10

You decide which arrangements of data work best for you by using the processes of **data analysis** and **database design**.

In data analysis, you investigate the various ways your data can be used. In database design, you create specific table structures based on your analysis. The design phase results in a set of table descriptions, known as a schema, for your database.

What Is SQL?

SQL is the acronym for **Structured Query Language**, which is defined by ANSI standards in the United States and by X/OPEN standards in Europe.

The way into a relational database is through a **query language**, which is a set of operators, **expressions**, and statements that let you manipulate the database in various ways. You create queries, as well as other kinds of statements, in IMAGE/SQL by using SQL. You issue the statements directly through an interactive command processor (ISQL) or indirectly through an application program.

SQL includes statements that let you do the following:

- Create databases
- Access databases
- Provide security
- Promote data integrity
- Regulate concurrent access

SQL statements are shown in uppercase letters (for example, SELECT) in this manual.

What Is TurboIMAGE/XL?

TurboIMAGE/XL is a database management system that operates on HP 3000 computer systems. This high performance database offers intrinsic access through specialized system procedures. This database engine serves as the storage mechanism for IMAGE/SQL.

Making the transition from TurboIMAGE/XL to IMAGE/SQL is relatively simple. Chapter 3, "Moving from TurboIMAGE/XL to IMAGE/SQL," addresses this transition in detail.

What Is a DBEnvironment?

In IMAGE/SQL, you create one or more databases in a structure called a DBEnvironment. The structure of a DBEnvironment is shown in Figure 2-2.

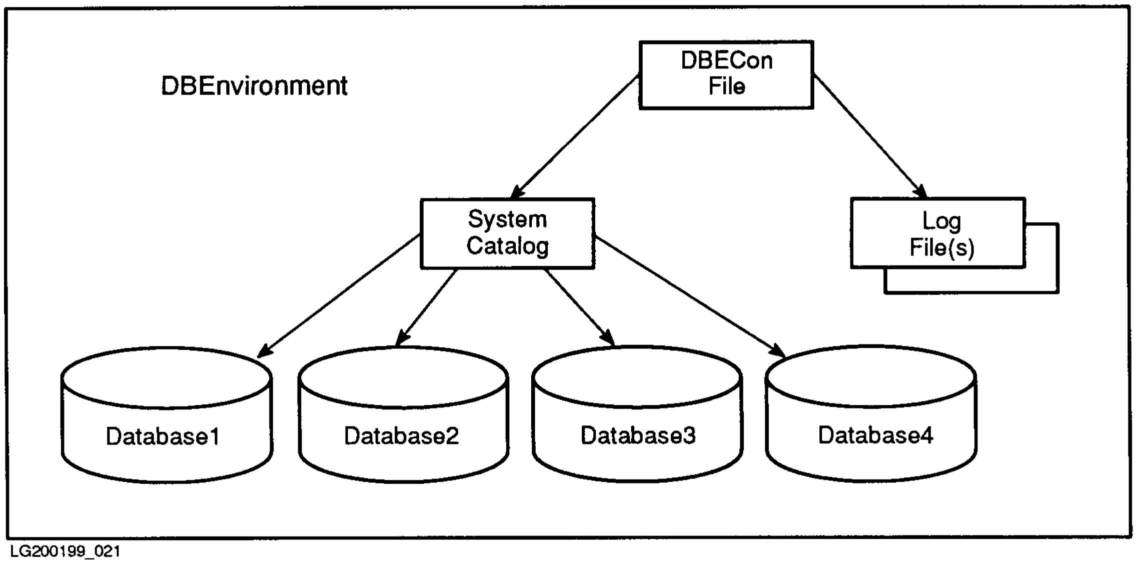


Figure 2-2. Structure of a DBEnvironment

The following **object**s are the most important parts of the DBEnvironment:

- The DBECon file
- DBEFiles
- DBEFileSets
- Databases
- Tables
- Authorities
- System catalog
- Log files

Objects are structures created and stored in an IMAGE/SQL DBEnvironment. IMAGE/SQL connects the DBEnvironment objects with the TurboIMAGE/XL database structures. This is discussed in greater detail in Chapter 5, "IMAGE/SQL Tasks."

To better understand the DBEnvironment and its objects, imagine that it is like a library that stores books, periodicals, or other information. Like a library, the DBEnvironment is a *physical location* for information, so you need to set aside physical space for it. Also like a library, the DBEnvironment uses a *logical method* for storing and retrieving information.

The DBECon file

When you create a DBEnvironment, IMAGE/SQL creates a physical file known as the DBEnvironment Configuration File or **DBECon file**. This file contains basic information that is used every time the DBEnvironment is opened. The DBECon file has the same name as the DBEnvironment itself. If the DBEnvironment is like a library, the DBECon file is like a building directory that points to the other DBEnvironment components.

DBEFiles

You must allocate some physical storage space by creating **DBEFiles**—operating system files that hold DBEnvironment data. Like the individual bookcases in a library, they have a specific capacity. DBEFiles have both physical names (operating system names) and logical names by which the files are known internally to the IMAGE/SQL system catalog, to be explained shortly.

DBEFileSets

DBEFiles are grouped together in logical groupings known as **DBEFileSets**. These are something like the different subject categories in a library. You create a DBEFileSet with a logical name (this corresponds to the subject category name). Then you add DBEFiles to the DBEFileSet as needed, as you would add bookcases to the category to add more space. For example, a section may be labeled "Computer Periodicals" and hold many bookcases with different magazines in them. DBEFileSets do not have a specific capacity; you can always add more DBEFiles to create more space in them.

Databases

The data in a DBEnvironment is stored in databases, which are groups of tables having the same **owner**. The database in a DBEnvironment is like division ownership of certain books or periodicals in a library. In a university, some of the periodicals may belong to a computer science library, others may belong to a medical library, though these may be housed inside the same building and use the same card catalog. In such a subdivision of the library, certain books and periodicals could only be checked out by people belonging to the appropriate division or having special permission.

When you create an object in the DBEnvironment, you are its owner by default, and therefore it belongs to your database.

Tables

The most important objects you create are tables. A table is like a periodical stored in a library, and the rows in the table are like individual articles in the periodical.

Tables have only logical names; a table name is like the title of the periodical. When you create a table, you only need to define its name and characteristics and associate it with a DBEFileSet. This is like adding a new periodical to a subject category in the library.

System Catalog

To enable readers to find issues of *ComputerWorld*, the librarian puts an entry in a card catalog that shows where the issues are stored. In the DBEnvironment, SQLCore inserts an entry in the **system catalog** for each database object you create. The system catalog is created at the time you create a DBEnvironment.

The system catalog is physically located in a DBEFile known as DBEFILE0. This file contains extra space for the additional entries that are made by SQLCore each time you create a new object. Like a good librarian, IMAGE/SQL keeps the system catalog up-to-date for you, without your being aware of it.

The system catalog is a set of tables for internal use by SQLCore. These tables are associated with a DBEFileSet known as **SYSTEM**. SQLCore uses the system catalog to look up database objects much as you would use the card catalog in a library to look up the bookcase and shelf number for a book or periodical.

Log Files

IMAGE/SQL provides **logging** of all transactions that take place in the DBEnvironment. Like the library's record of items checked out and returned, the log file is a record of the rows in database tables that are added, deleted, or changed. The log makes it possible to keep data consistent when multiple users are accessing the system, and it makes recovery possible in the event of a system failure. Information about the system log is stored in the DBECon file.

Because logging is essentially automatic, it will not be discussed any further in this manual. For more information, refer to the chapter "Backup and Recovery" in the *ALLBASE/SQL Database Administration Guide*.

How Do I Create a DBEnvironment?

You use the SQL START DBE NEW statement to create a DBEnvironment:

```
START DBE 'DBEnvironmentName' NEW
```

Once the DBEnvironment exists, you can create databases within it.

The START DBE NEW statement lets you supply options to specify many of the run-time characteristics of the DBEnvironment. The simple form of the statement shown above uses default values for these options. The defaults are described in detail in the *ALLBASE/SQL Database Administration Guide* chapter entitled "DBEnvironment Configuration and Security."

When you create a DBEnvironment, you are granted the broadest kind of **authority**—permission in an IMAGE/SQL DBEnvironment—to create and remove objects. This authority is known as DBA (database administrator) authority.

How Do I Create a Database?

With IMAGE/SQL, the underlying database is TurboIMAGE/XL. You create a TurboIMAGE/XL database using a schema and the DBSCHEMA and DBUTIL utility programs. Some third-party tools can also be used to create TurboIMAGE/XL databases. Refer to the *TurboIMAGE/XL Database Management System Reference Manual* for details. Also see Chapter 4, "A Database Example," in this manual for a database creation example.

How Do I Access a Database through SQL?

You use the SQL CONNECT statement to establish a connection to a DBEnvironment. This statement must be executed by some type of SQL program, such as:

- ISQL
- Application programs you create yourself
- Third-party applications
- Existing programs

During database access, you perform queries or other operations that manipulate data by inserting, deleting, or modifying rows in tables. This process is called **data manipulation**. Here is a simple query using the SELECT statement, which displays a selection of data from the Employees table. The asterisk indicates *all* columns are to be displayed:

```
SELECT * FROM Employees;
```

Below is the **query result**:

```
select * from employees;
--------------+-----------+-----------
LASTNAME      |FIRSTNAME  |EMPNUMBER
--------------+-----------+-----------
Harrison      |Gerald     |    2432099
Abelson       |Annette    |    3510044
Stanley       |Peter      |    3540011
Walters       |Georgia    |    9124772
```

The query result is also known as a **result table**.

Other data manipulation statements include the SQL INSERT, UPDATE, and DELETE statements. These let you add rows to a table, update specific column values in existing rows, or delete rows. Examples of data manipulation are shown in Chapter 6, "Practicing with IMAGE/SQL Using the MusicDBE."

How Do I Control Database Access?

You use **data control language** to determine who has access to the information in a database. This is very important for security. Data control language confers authorities on specific users to perform specific tasks. The most powerful authority is **DBA authority** (database administrator authority). The **DBECreator**, that is, the person who creates a new DBEnvironment, automatically has DBA authority in that DBEnvironment. Someone with DBA authority can then use the GRANT statement to give authorities to other users. The following example grants permission to update the Employees table to a user known as Harry:

```
GRANT UPDATE ON Employees to Harry
```

The controls you define for database security can be as simple or as elaborate as you wish.

See Chapter 6, “Practicing with IMAGE/SQL Using MusicDBE,” for more information on database accesses.

3

Moving from TurboIMAGE/XL to IMAGE/SQL

If you are coming to IMAGE/SQL from the world of TurboIMAGE/XL, this chapter will help you make the transition. This chapter describes comparative terminology and discusses steps in using IMAGE/SQL. At the completion of this chapter, you will have the knowledge of IMAGE/SQL needed to complete the tasks in Chapters 4, 5, and 6.

The following topics are covered in this chapter:

- Basic structures
- Creating databases
- Tables versus data sets
- Mapping of data types
- Defining security
- Methods of accessing databases

The goal in discussing these topics is not to present a complete picture of either system, but rather to suggest some points of correspondence to make learning IMAGE/SQL easier. The discussion is deliberately oversimplified. For definitions of IMAGE/SQL terms, refer to Chapter 2, "Basic Concepts," and the Glossary. Complete information about TurboIMAGE/XL is in the *TurboIMAGE/XL Database Management System Reference Manual.*

Although IMAGE/SQL and TurboIMAGE/XL use different terminology, they share several similar concepts. Table 3-1 lists IMAGE/SQL terms and lists the equivalent TurboIMAGE/XL concept. The IMAGE/SQL terms listed here are defined in more detail in the Glossary.

Table 3-1.
Comparing IMAGE/SQL and TurboIMAGE/XL Concepts

IMAGE/SQL Term	TurboIMAGE/XL Term
DBEnvironment	Collection of databases
Table	Data set
Column	Data item within entry
Row	Record or entry
System catalog	Root file
DBECon file	Root file[1]
DBEUserID	Password[1]
DBEFile	DB01, DB02, DB03[1]

[1]Shares some similarities, but differences do exist.

Basic Structures

Figure 3-1 shows the basic architecture of the TurboIMAGE/XL system.

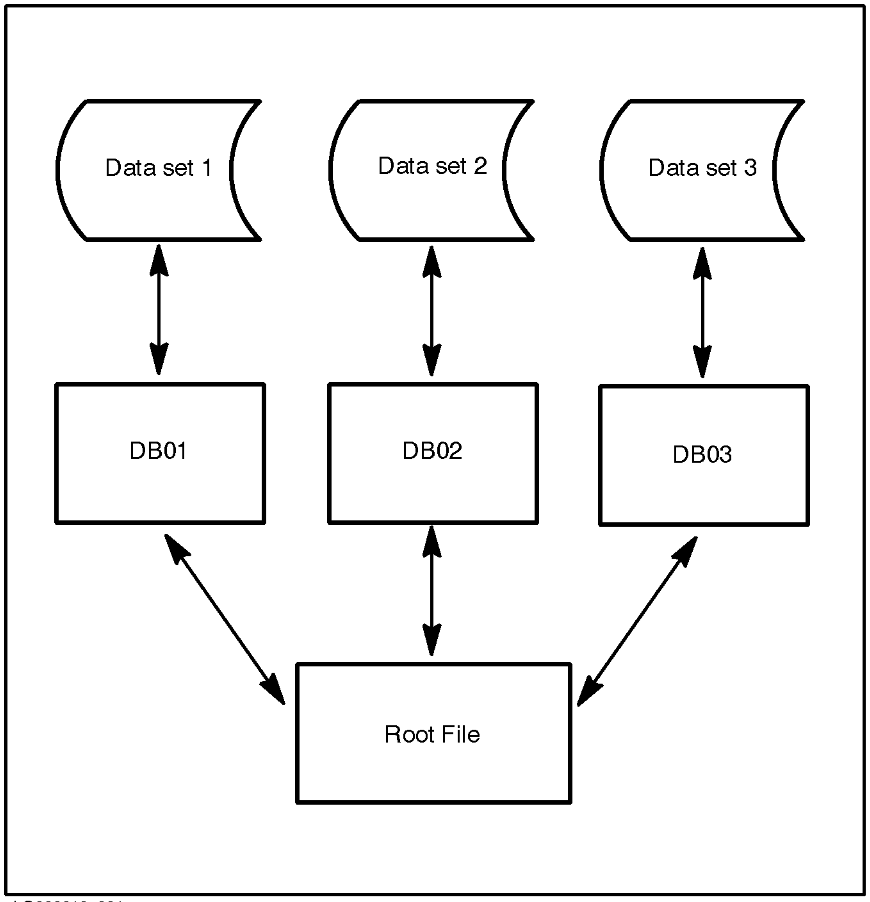


Figure 3-1. TurboIMAGE/XL Architecture

The TurboIMAGE/XL database manager accesses data in each data set as needed, based on information given in the root file.

Figure 3-2 shows the IMAGE/SQL architecture.

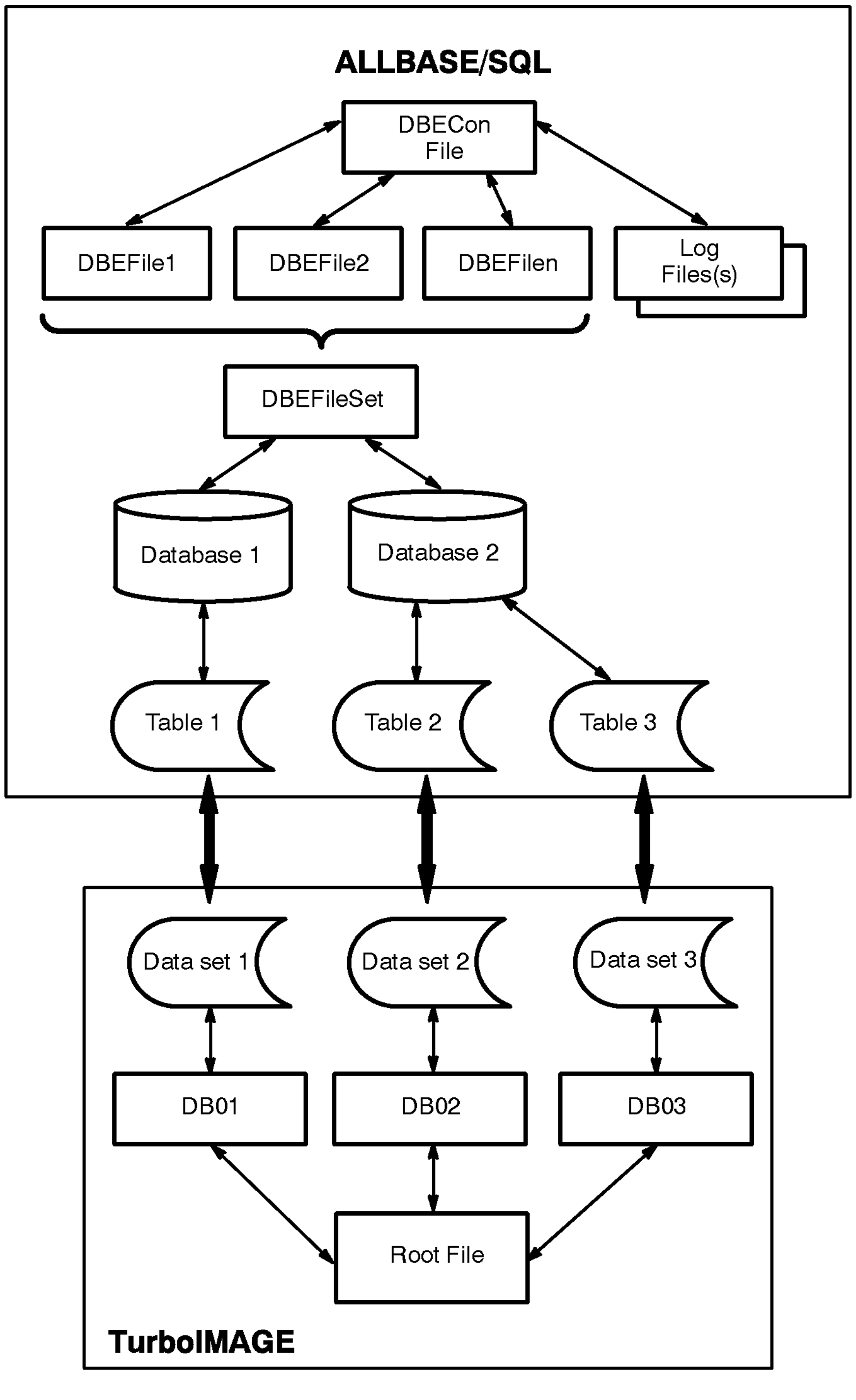


Figure 3-2. IMAGE/SQL Architecture

Creating Databases

In TurboIMAGE/XL, you create a **database** using the following steps:

1. Create a schema. A common way of doing this is to enter the schema into a text file created with an editor.
2. Run DBSCHEMA to generate a root file from the schema.
3. Run DBUTIL to create data sets based on the root file.
4. Create a DBEnvironment.

With TurboIMAGE/XL, a schema is required to define a database. The schema contains definitions of all the data items in your database and describes all the data sets in the database. It also specifies the desired security for the database by defining passwords for specific users. The schema can also be used to create the same database structure in different groups and accounts or on different systems.

To access a TurboIMAGE/XL database with IMAGE/SQL, you need a DBEnvironment. You create a DBEnvironment with a SQL START DBE NEW statement. The statement can be issued interactively (through ISQL) or through an application program. The START DBE NEW statement is as follows:

`START DBE '`*DBEnvironmentName*`' NEW`

The START DBE NEW statement creates a file known as a DBECon file, which is similar to the root file in TurboIMAGE/XL. The DBECon file contains information about startup parameters for the DBEnvironment and its logs. START DBE NEW also creates a structure within the DBEnvironment known as a system catalog, which is a set of information about all the databases in the DBEnvironment.

If you use the START DBE statement and the DBEnvironment does not exist, the system will ask if you want one automatically created. See Chapter 5, "IMAGE/SQL Tasks," for more details and an example.

After the database and DBEnvironment have been created, you must attach the TurboIMAGE/XL database to the DBEnvironment.

Root File versus DBECon File and System Catalog

With TurboIMAGE/XL, the root file, which is generated by DBSCHEMA from the schema, contains security information and definitions of all the data sets in the database. The user executing DBSCHEMA becomes the database creator.

The DBECon file in ALLBASE/SQL is created when you issue the START DBE NEW statement in ISQL. The DBECon file, which has the same name as the DBEnvironment, contains the name of the DBECreator and the names of the logs associated with the DBEnvironment. The DBECon file also contains startup parameters, such as SINGLE or MULTI user mode, but does not contain the names of tables or other database objects; these are stored in the system catalog.

The system catalog contains structural information about database objects and tables, much the way the root file does for standard TurboIMAGE/XL. This catalog serves as an internal schema for accessing data.

When a TurboIMAGE/XL database is attached to a DBEnvironment, data set and item definitions are copied into the system catalog. This provides a mapping to the TurboIMAGE/XL data from the DBEnvironment. The data is not copied; it remains in the TurboIMAGE/XL database.

Naming Conventions

TurboIMAGE/XL data sets and items may contain some characters that are not allowed in IMAGE/SQL. For example, the hyphen is not allowed in IMAGE/SQL names. Therefore, hyphens must be represented in another way in IMAGE/SQL, such as with an underscore (_). Specifically, the characters not allowed in IMAGE/SQL names are: + - * / ? ' % &. For more details, refer to the *HP IMAGE/SQL Administration Guide*.

Tables versus Data Sets

In TurboIMAGE/XL, the access mechanism is the data set, which consists of a set of entries containing an ordered series of data items. A data set is either a detail or a master, and, if it is a master, it is either manual or automatic.

In SQL, a table is an unordered set of rows containing columns. In IMAGE/SQL, a table is synonymous with a data set. Tables are not labeled manual or automatic, master or detail.

Mapping of Data Types

Both TurboIMAGE/XL and IMAGE/SQL have data types that do not map exactly to a type in the other system. However, a satisfactory mapping with appropriate conversions can easily be done for most TurboIMAGE/XL data types.

Basic Mapping

Table 3-2 shows the mapping of the most common data types from TurboIMAGE/XL to IMAGE/SQL:

Table 3-2. Mapping of TurboIMAGE/XL and IMAGE/SQL Data Types

TurboIMAGE/XL Data Type	IMAGE/SQL Data Type	Description
I,J	SMALLINT	16-bit integer
I2,J2	INTEGER	32-bit integer
K1,K2	INTEGER	Requires conversion from binary to integer
P*n*	DECIMAL(*n*-1,0)	Packed decimal
R4	FLOAT	Conversion from HP 3000 real to IEEE real
U(*n*), X(*n*)	CHAR(*n*)	Byte character string
Z*n*	DECIMAL(*n*,0)	Requires conversion from zoned decimal to packed decimal

Compound Items

TurboIMAGE/XL compound items are not compatible with IMAGE/SQL data types because IMAGE/SQL does not accomodate arrays. In IMAGE/SQL, you need to create a separate column description for each member of the compound item.

Default Values

Default values indicate the absence of a value. IMAGE/SQL assigns the missing field an empty string (a blank) for character values or a zero for numeric values.

TurboIMAGE/XL items are always defined in IMAGE/SQL with the NOT NULL and default value specification.

Defining Security

TurboIMAGE/XL and IMAGE/SQL differ markedly in their implementation of security systems.

TurboIMAGE/XL Security

For external security, database users must be valid users in the account where the root file resides or have access to it through system security measures. For internal security, passwords are assigned to numbered classes. These classes can be associated with read and write permissions to items and data sets. When accessing a database, you must specify a password that grants you these permissions. Refer to the *TurboIMAGE/XL Database Management System Reference Manual* for further details on security.

Granting and Revoking IMAGE/SQL Authorities

Your user identification within the DBEnvironment is the same as your system logon.

The creator of the DBEnvironment is called the Database Creator (DBC) and has Database Administrator (DBA) authority. The DBA has the responsibility to GRANT and REVOKE authorities of other users. These authorities relate to the DBEnvironment or to specific tables within it.

Some of these authorities allow users to CONNECT to the DBEnvironment and SELECT, INSERT, and UPDATE against tables. If you are the creator of a table, you have OWNER authority over that table, which lets you perform any operation on it, including granting authorities to other users.

With IMAGE/SQL, by default, the TurboIMAGE/XL creator (or someone who knows the maintenance word to the TurboIMAGE/XL database) and the DBA of the DBEnvironment is set up as the DBC with DBA authority within the DBEnvironment. Additional users can be added and authorities granted.

Defining IMAGE/SQL Groups

In IMAGE/SQL, you can define authorization groups and then grant authorities to them; then you can add users to the groups, at which point they immediately receive the authorities the group possesses. This makes it possible to create an authorization scheme that is independent of any list of particular users and passwords. An authorization group may be a member of another authorization group.

Defining Views in IMAGE/SQL

A different approach to security is possible in IMAGE/SQL through the use of views. For a table that contains some sensitive information and some widely used information, you can create a view that contains only the widely-used information, grant appropriate access on the view to a wide range of users, then restrict the access on the base table to only a few users.

Methods of Accessing Databases

TurboIMAGE/XL and IMAGE/SQL both offer a variety of tools for accessing databases and both provide techniques for **concurrency** control, which regulates access by more than one user at a time.

Interactive Access

TurboIMAGE/XL interactive access is through Query/3000, which lets you find database entries and report on them using the Query command language. In IMAGE/SQL, the interactive interface is known as ISQL, which uses Structured Query Language (SQL) to access the database and display query results.

Programmatic Access

A major difference between TurboIMAGE/XL and IMAGE/SQL is in the programmatic interface. TurboIMAGE/XL provides a set of intrinsics that you use in application programs to open databases, obtain locks, retrieve data, unlock data items and data sets, and close a database.

IMAGE/SQL uses **embedded SQL programming**. You insert standard SQL statements in an application program, then you preprocess the program to convert the SQL statements into valid procedure calls in the language you are using. The converted code is compiled and linked with a library of IMAGE/SQL routines. You can prototype and test your queries in ISQL before using them in an application, thus saving development time. Embedded SQL also includes a set of dynamic statements that let your end users perform queries, inserts, updates, and deletes through your applications.

Locking Mechanisms

In TurboIMAGE/XL, you use the DBLOCK intrinsic in certain access modes to provide **locking** at the database level, the data set level, or the data item level. Locking must be explicitly requested by the user; it is required for concurrent updates. You can request locks conditionally in TurboIMAGE/XL, which means that the call returns if the lock request fails.

In IMAGE/SQL, locking is provided when:

- it is implicitly specified when the data is accessed (such as when inserting, deleting, or updating.) The whole data set is locked until the end of the transaction. For example,

```
BEGIN WORK

SELECT * FROM Turbo.Dataset

DELETE FROM Turbo.Dataset      <-----  LOCK begins
    WHERE Mynumber=25

SELECT * FROM Turbo.Dataset

COMMIT WORK                    <-----  UNLOCK occurs
```

- it is explicitly specified by the LOCK TABLE statement.

Locking is unconditional and applies at the table level; row and page level locking is not supported. Refer to the *ALLBASE/SQL Reference Manual* for more details.

4

A Database Example

This chapter presents a simple database example that will be used in the remaining chapters of this manual. The areas to be covered are:

- Understanding the environment
- Describing the example application
- Describing the logical data entities
- Identifying relationships between entities
- Defining the data sets
- Defining the security scheme
- Defining the TurboIMAGE/XL database
- Creating the TurboIMAGE/XL database

After completing this chapter, you will have a TurboIMAGE/XL database ready to undergo IMAGE/SQL tasks described in Chapter 5, "IMAGE/SQL Tasks."

Understanding the Environment

The IMAGE/SQL environment requires a TurboIMAGE/XL database that may or may not have an existing application to support it. If your site already has TurboIMAGE/XL databases, your design work is already done for you. To describe the concepts of IMAGE/SQL, a TurboIMAGE/XL database must be defined.

The application presented in this chapter is a simple demonstration of some key IMAGE/SQL features. The design is not sophisticated. Database design is the subject of much theoretical discussion and debate, but performance and flexibility usually surface as desired results. The database design presented here may not provide either, but it does demonstrate some key points of IMAGE/SQL.

The Example Application

Suppose a radio station wants to have a database of classical music recordings for use by the program director and announcers. The station needs this information to plan a schedule of broadcasts, to maintain a log of what is played, and to publish a monthly listener guide. Here is some of the information that will be needed:

- Album title, price, and duration
- Recording company and date recorded
- Medium (CD or cassette)
- Selection titles and duration
- Names of orchestra, conductor, singers, accompanists
- Composer's names and other personal information
- Miscellaneous comments
- Date, time, and announcer for selections played

We will call these items **attributes**. They tend to fall into different categories. Some pertain to albums, some to specific selections, and others relate to when the selection is played. Before the database can be created, the attributes must be organized into logical entities.

Logical Data Entities

To logically order the above information, each attribute must be analyzed in terms of how it will be used. These logical groupings of attributes are called **entities**.

The program director needs to know what albums are available, the selections on each, how long each selection is, and some general information about the style of performance. The announcers need to know which selections are on which albums, how long each selection is, and some information about the composers and artists. The station log will track all music played, which announcer played it, and when the selection started and finished.

The attributes can be divided into the following four categories of information as required by the radio station:

- Album information
- Selection information
- Composer information
- Station log information

These constitute our four data entities.

Next, you need to identify which attributes belong to which entity. Below is a table that shows one way of subdividing the attributes amongst the entities. For each entity, one attribute in the list will uniquely identify an instance of the entity. This attribute is known as a **key**.

Attributes for Four Entities

Album Entity	Selection Entity	Composer Entity	Station Log Entity
Album Name	Selection Title	Name	Selection Title
Medium	Composer Name	Date of Birth	Start Time
Album Cost	Duration	Birthplace	End Time
Recording Company	Performers	Comment	Announcer
Date Recorded	Comment		Comment
Manufacturer's Code			
Comment			

As the design evolves, entities become tables and attributes become columns. Recall that tables and columns are SQL concepts. Using TurboIMAGE/XL terminology, the entities become data sets and the attributes become data items. The key, in TurboIMAGE/XL, is also known as a search item.

Examining the data, you will see some overlapping attributes. These will help you define the relationships between the entities.

Identifying Relationships Between Entities

After subdividing the data by entities, the next step of the design is to identify meaningful relationships between the entities described so far. For each relationship you identify, an attribute or group of attributes must support the relationship. This may mean adding one or more attributes to support the relationship.

What are the relationships among the entities in the sample data? For the Albums and Selections entities, there is a relationship of *content*; that is, each album contains a specific group of selections. For the Composers and Selections entities, the relationship is one of *authorship*; each composer has created one or more selections.

For each relationship defined, linking attributes are added to related entities. In the case of Selections, the link is Selection Name. In the case of Selections and Composers, the link is Composer Name. Where the attribute is missing, it must be added to define the relationship.

After distributing the attributes, the next step is to identify those attributes which will serve as keys to any unique occurrence of an entity. It may be necessary to create attributes to clarify certain ambiguities. For example, two albums could have the same name. This situation is clarified by creating an AlbumCode attribute that will serve as the key to that entity. You can now see a set of relational tables emerging, as follows (* indicates a key column):

Columns and Keys for Four Tables

Albums Table	Selections Table	Composers Table	Station Log Table
*AlbumCode	*AlbumCode	*ComposerName	*AlbumCode
AlbumTitle	*SelectionName	Birth	*SelectionName
Medium	*ComposerName	Death	StartTime
AlbumCost	Duration	Birthplace	EndTime
RecordingCo	Performers	Comment	Announcer
DateRecorded			
MfgCode			
Comment			

Some entities contain a unique occurrence for the key values, other entities will have multiple occurrences for a key value. For example, the composer Beethoven will exist once in the Composers entity. There can be, however, many selections by this composer. Also, a selection, Beethoven's Fifth Symphony, could be on several albums performed by different orchestras.

From a relationship point of view, this is called **referential integrity**, where an occurrence of information in one entity is required to support the existence of information in another entity. To demonstrate referential integrity, suppose the station receives a new album containing Beethoven's greatest symphonies as performed by the San Francisco Symphony Orchestra. This album is entered into the Albums entity. Each selection from the album is added to the Selections entity, which includes the attribute Composer Name. A relationship exists between Selections and Composers based upon this attribute. For the data to be complete, an entry for Beethoven must be added in the Composers entity. Once this is done, referential integrity has been established. Without it, the referential model is incomplete.

Defining the Data Sets

Once you have added the necessary key attributes that support the relationships, you have completed the majority of the work in designing the TurboIMAGE/XL database. Before the database can be completed, paths, a distinction of TurboIMAGE/XL, must be defined.

Paths are fixed relationships between entities that provide fast access to data and define referential integrity. Relationships are defined by designating entities as Master Data Sets and Detail Data Sets, and then linking them with paths.

Note that the Composers and Albums entities each have only one key item, whereas the other entities have more than one. A Master set, by definition, has one key item, whereas a Detail set may have up to sixteen. A Detail set contains one or more entries linked to a Master entry where the key values agree.

For example, Album Code 0019 is assigned to the Beethoven's hits album mentioned before. This information is in the Albums entity. There are two selections on the album:

- Beethoven's Fifth Symphony
- Beethoven's Ninth Symphony

These entries are in the Selections entity.

If a query is made for Album Code 0019 against Albums, one entry is returned. There should only ever be one entry returned. If the same Album Code is requested from Selections, two entries are returned, equal to the number of selections on the album.

Albums, then, is a candidate for being a Master data set and Selections will be a Detail data set. Selections also has a second key, Composer Name, a condition that forces it to be a Detail set.

One other distinction within TurboIMAGE/XL before continuing with our design: Master sets are either Manual Masters or Automatic Masters. Each has a single key, but Automatic Masters can contain no other attributes. Manual Masters can have other attributes per entry.

Also, Automatic Master key values are added and deleted automatically; their function is to provide fast access paths to data. Manual Master key values and their associated attributes must be added and deleted explicitly, thereby supporting the referential integrity concept discussed earlier and providing fast access to data.

The TurboIMAGE/XL database designed so far is shown in Figure 4-1.

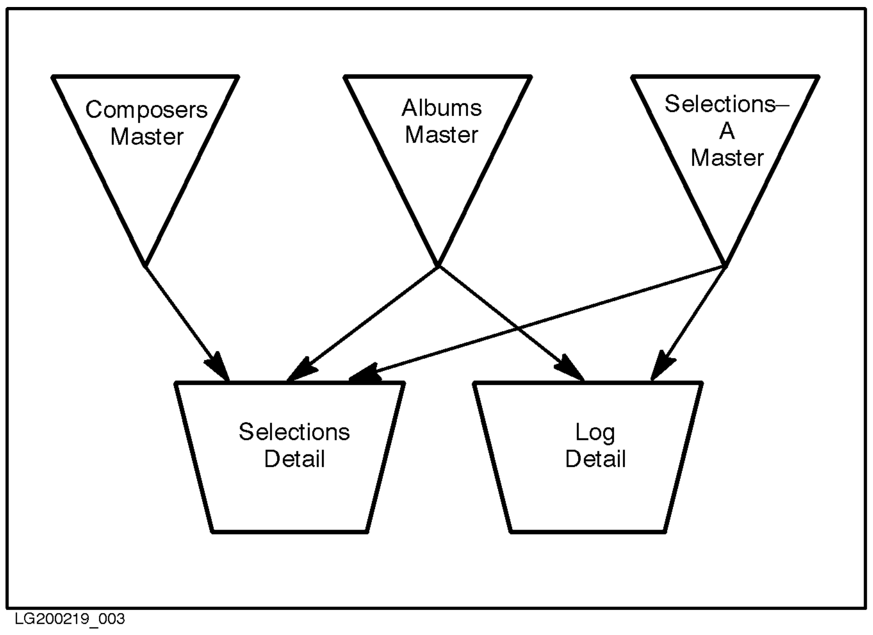


Figure 4-1. Database Design

Composers and Albums are Manual Master data sets, Selections and Log are Detail data sets. Because Selection Name is a key value in both Selections and Log, a supporting Automatic Master was created to supply the keyed access to these two entities.

Before you see the schema to create this database, a final design consideration must be addressed: security.

Defining the Security Scheme

The radio station manager wants to have complete access to all the information in the database. Announcers need to read information about Albums, Selections, Composers, and the Log. The program director adds information about new albums and decides what will be played for publication in the monthly listener guide.

A security scheme will allow database accessors to do only what has been assigned or expected of them and nothing more. The security can be broken down to these three levels:

Manager	Read and Write all entities
Director	Read all entities, Write all but Log
Announcer	Read all entities, Write Log

To complete this security scheme in TurboIMAGE/XL, you will define a password for each of these levels and assign it a numeric value. For simplicity sake, we will use:

```
MGR     10
DIR     20
ANNCR   30
```

Applying these to the data sets results in the following:

```
Albums          Read 10, 20, 30;  Write 10, 20
Selections      Read 10, 20, 30;  Write 10, 20
Composers       Read 10, 20, 30;  Write 10, 20
Log             Read 10, 20, 30;  Write 10, 30
```

Each individual will be assigned one of these passwords to be used when accessing the database. The password will designate the data that can be accessed and the method of access.

The next section will show you the schema for our now complete TurboIMAGE/XL database. In that schema, you will see the syntax for this security schema.

Defining the TurboIMAGE/XL Database

A TurboIMAGE/XL schema is made up of three parts:

- Password
- Item
- Set

The Password Part was discussed in the previous section.

The Item Part contains the list of attributes, called items, in the database. Only a single definition is required, even if the item is used in the database in more than one set. Also included in the Item Part is each item's data type and length. Data types are discussed in the section, "Mapping of Data Types," in Chapter 3 and will not be addressed here.

The Set Part describes the entities, the set type, the security, and the items that make up the entry. Size is also defined, with a capacity value that indicates the maximum number of entries for that entity.

The database schema presented in this section is included with the IMAGE/SQL product and is found in SAMPLEDB.SYS. Practice with this database is in Chapter 6, "Practicing with IMAGE/SQL Using MusicDBE."

The schema for the sample database is as follows:

```
Begin data base Music;
Passwords: 10 MGR;
           20 DIR;
           30 ANNCR;
Items:
  AlbumCode,       I2;
  AlbumCost,       P8;
  AlbumTitle,      X40;
  Announcer,       X40;
  Birth,           X16;
  BirthPlace,      X40;
  Comment,         X80;
  ComposerName,    X16;
  DateRecorded,    X16;
  Death,           X16;
  EndTime,         X16;
  Medium,          X2;
  MfgCode,         X40;
  Performers,      X40;
  RecordingCo,     X10;
  SelectionName,   X40;
  StartTime,       X16;
  Timing,          X16;
```

```
Sets:
Name:  Albums, Manual(10,20,30/10,20);
Entry: AlbumCode (2),
       AlbumTitle,
       Medium,
       AlbumCost,
       RecordingCo,
       DateRecorded,
       MfgCode,
       Comment;
Capacity: 1000;

Name:  Composers, Manual (10,20,30/10,20);
Entry: ComposerName (1),
       Birth,
       Death,
       Birthplace,
       Comment;
Capacity: 1000;

Name:  Selections-A, Automatic;
Entry: SelectionName(2);
Capacity: 1000;

Name:  Selections, Detail (10,20,30/10,20);
Entry: AlbumCode (Albums),
       SelectionName (Selections-A),
       ComposerName (Composers),
       Timing,
       Performers,
       Comment;
Capacity: 1000;

Name:  Log, Detail (10,20,30/10,30);
Entry: AlbumCode (Albums),
       SelectionName (Selections-A),
       StartTime,
       EndTime,
       Announcer;
Capacity: 1000;

End.
```

The next step is to create a root file for this database with DBSCHEMA as follows:

```
        :FILE DBSTEXT = MUSICSCH.SAMPLEDB.SYS
        :RUN DBSCHEMA.PUB.SYS;PARM=3
```

The schema is displayed here, followed by this detail:

```
DATA SET      TYPE  FLD  PT  ENTR  MED  CAPACITY    BLK  BLK  DISC
  NAME              CNT  CT  LGTH  REC              FAC  LGTH SPACE

ALBUMS           M   8    2   98    115  1000        4    461  1008
COMPOSERS        M   5    1   84    95   1000        4    381  752
SELECTIONS-A     A   1    2   20    37   1000        10   371  304
SELECTIONS       D   6    3   98    110  1000        4    441  1008
LOG              D   5    2   58    66   1001        7    463  576
        TOTAL DISC SECTORS INCLUDING ROOT: 3664


NUMBER OF ERROR MESSAGES: 0
ITEM NAME COUNT: 18      DATA SET COUNT: 5
ROOT LENGTH: 916         BUFFER LENGTH: 463     TRAILER LENGTH: 256

ROOT FILE MUSIC  CREATED.
```

Creating the TurboIMAGE/XL Database

The last step to creating the database is to create the sets. This is done with DBUTIL:

```
:RUN DBUTIL.PUB.SYS
HP30391C.04.09 TurboIMAGE/XL:  DBUTIL (C) COPYRIGHT HEWLETT-PACKARD

>>CREATE MUSIC
  Database MUSIC has been CREATED.
>>EXIT
```

Your TurboIMAGE/XL database now exists.

This chapter covers a very small part of creating databases with the TurboIMAGE/XL Database Management System. For more details, refer to the *TurboIMAGE/XL Database Management System Reference Manual*.

You are now ready to begin the tasks in Chapter 5, "IMAGE/SQL Tasks," to access this database with SQL.

5

IMAGE/SQL Tasks

This chapter shows selected IMAGE/SQL tasks. See Chapter 6, "Practicing with IMAGE/SQL Using MusicDBE," for more examples using MusicDBE.

This chapter shows the following:

- Setting up a database with ISQL
- Configuring a DBEnvironment
- Attaching a TurboIMAGE/XL database
- Detaching a TurboIMAGE/XL database
- Displaying IMAGE/SQL utility information
- Issuing MPE/iX commands from the IMAGE/SQL utility
- Setting IMAGE/SQL file equations
- Logging IMAGE/SQL utility commands
- Using IMAGE/SQL utility command files
- Accessing TurboIMAGE/XL data with ISQL
- Maintaining the ATCINFO file

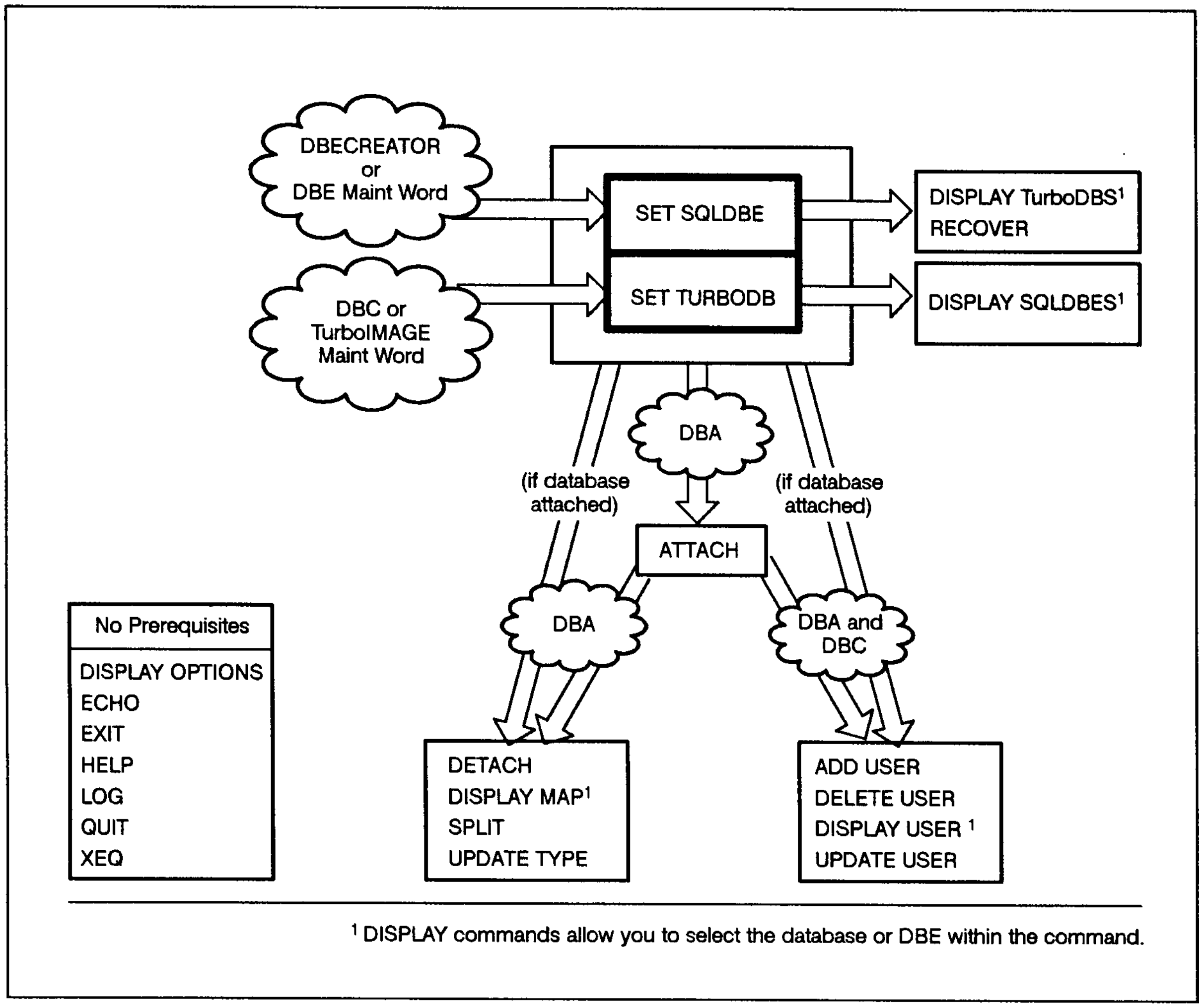
About the Tasks

- Where applicable, each task is divided into three parts:

Getting Ready	describes preparatory steps and gives information you need to know before performing the task.
Performing the Task	describes how to perform the task.
Task Reference	lists information useful when performing the task and provides cross-references to related tasks.

- To perform most tasks, you must be attached to the DBEnvironment. Here are some general guidelines:
  - □ To issue the SET commands, you must be the creator or supply the maintenance word of the TurboIMAGE/XL database or DBEnvironment named in the command.
  - □ For most other commands, you must be a DBA of the DBEnvironment. For commands involving database security, you must be the Database Creator. The maintenance word is not sufficient.

- Figure 5-1 shows the prerequisites needed to issue all IMAGE/SQL utility commands. For complete IMAGE/SQL utility command syntax, refer to the *HP IMAGE/SQL Administration Guide*.



LG200167_1

Figure 5-1. IMAGE/SQL Utility Command Prerequisites

Setting Up a Database with ISQL

This task shows how to set up a new database environment using the IMAGE/SQL utility and ISQL, the interactive component of Hewlett-Packard's SQL products.

The steps to create the database environment are:

- Running ISQL
- Creating a DBEnvironment

The example chosen to illustrate these steps is a database of information about record albums. This database environment will be used again in Chapter 6, "Practicing with IMAGE/SQL Using MusicDBE."

Running ISQL

Before you can create a new DBEnvironment, you must run ISQL.

Performing the Task

Run ISQL by entering the following at your operating system prompt:

```
:RUN ISQL.PUB.SYS (Return)
```

You will then see the ISQL banner as shown in Figure 5-2 (some details may be slightly different on your system):

```
                 IIIIIIII    SSSSSSSS     QQQQQQQ      LL
                    II      SS           QQ     QQ     LL
                    II      SS           QQ     QQ     LL
                    II        SSSSSS     QQ     QQ     LL
                    II              SS   QQ   Q QQ     LL
                    II              SS   QQ     QQ     LL
                 IIIIIIII    SSSSSSS       QQQQQQ QQ   LLLLLLLL


                                         MON, FEB  8, 1993, 10:51 AM
HP36216-02A.F0.29           Interactive SQL/3000           ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992. ALL RIGHTS RESERVED.

isql=>
```

Figure 5-2. ISQL Banner

- Use a semicolon and press (Return) to terminate a command in ISQL. If you forget to use the semicolon, a continuation prompt appears:

  ```
  >
  ```

  Type a semicolon and press (Return), or to terminate, type //.

- Leave ISQL by typing the following:

```
isql=>EXIT; (Return)
```

  If ISQL asks whether or not you want to commit work, you must reply either Y or N and press (Return). Entering Y makes the work you have done permanent.

Creating a DBEnvironment

To create a new DBEnvironment, use the START DBE NEW command at the ISQL prompt. This is explained in the next task.

Task Reference

- For complete information about startup parameters, refer to the "DBEnvironment Configuration and Security" chapter of the *ALLBASE/SQL Database Administration Guide*.
- For more information about log files, refer to the "Backup and Recovery" chapter in the *ALLBASE/SQL Database Administration Guide*.
- For more information on configuring a DBEnvironment, refer to the entry for START DBE NEW in the "SQL Statements" chapter of the *ALLBASE/SQL Reference Manual*. This entry describes all the default configuration values used in creating MUSICDBE.

Configuring a DBEnvironment

This task describes how to configure a DBEnvironment so that you can access your TurboIMAGE/XL database(s) with IMAGE/SQL (see Task Reference).

Getting Ready

- When all of the TurboIMAGE/XL databases to be attached to the DBEnvironment are created by the same user in one group and account, it is convenient to configure the DBEnvironment in this group and account.

  If this is not the case, several other issues should be considered. The following considerations apply if the TurboIMAGE/XL database to be attached exists in a different group and/or account than the DBEnvironment:

  - IMAGE/SQL supports standard MPE/iX security rules. Make sure correct user, group, and account capabilities are in place when you plan to access a TurboIMAGE/XL database from a DBEnvironment in a different account than the database.
  - Make sure maintenance words exist for both database management systems, because IMAGE/SQL utility administrators often need to specify DBEnvironment and TurboIMAGE/XL maintenance words if they are not the creator.
  - Be sure to grant DBA authority to everyone who will be performing IMAGE/SQL utility tasks (see Task Reference), because IMAGE/SQL utility administrators need DBA authority to perform most IMAGE/SQL utility tasks.
- Make sure the TurboIMAGE/XL database and the DBEnvironment have the same native language support (NLS) defined for them.
- When a DBEnvironment is configured, two files are created: DBEFILE0 and DBELOG1. IMAGE/SQL requires that these files be larger than their defaults. Make sure these files are large enough to accommodate IMAGE/SQL. In the example below, a file size of 500 pages is used for each of these files, but you may need to adjust these sizes, depending on the size and number of TurboIMAGE/XL databases you plan to attach (see Task Reference).

Performing the Task

Log on in the same group and account as the TurboIMAGE/XL database(s) and run ISQL. At the ISQL prompt, enter the START DBE NEW command. For example, to configure a DBEnvironment named MusicDBE, enter the following:

```
isql=> START DBE 'MUSICDBE' MULTI NEW
> MAXIMUM TIMEOUT = 10 SECONDS
> DEFAULT TIMEOUT = 5 SECONDS
> DBEFILE0 DBEFILE DBEFILE0 ⇐DBEFile0Definition
> WITH PAGES = 500,         ⇐   .       .
> NAME = 'MusicF0',         ⇐   .       .
> LOG DBEFILE DBELOG1       ⇐DBELog0Definition
> WITH PAGES = 500,         ⇐   .       .
> NAME = 'MusicLog';        ⇐   .       .
isql=> EXIT;
:
```

If you forget the semicolon, ISQL prompts you. At this prompt, enter a semicolon and ISQL will execute the command. The MULTI parameter is necessary if you plan on a multiuser IMAGE/SQL environment.

To avoid deadlock situations when there are more than two connections, you should either set the MAXIMUM TIMEOUT and DEFAULT TIMEOUT options with the START DBE NEW statement or change them with the SQLUtil ALTDBE command.

To set a maintenance word for the newly configured DBEnvironment, use the SETDBEMAINT command of SQLUtil. This utility prompts you for necessary information, as in the following example.

```
:RUN SQLUTIL.PUB.SYS

    THU, APR  8, 1993, 10:26 AM
HP36216-02A.F0.52              DBE Utility/3000            ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992,1993. ALL RIGHTS RESERVED.

>>SETDBEMAINT
Environment Name: MUSICDBE
Current Maintenance Word:
New Maintenance Word:  usr
Retype New Maintenance Word:  usr

Maintenance word changed.
>>EXIT
```

Listing the Newly Created Files

When you display the files in your current group, note that there are three newly created files: MUSICDBE, DBEFILE0, and DBELOG1.

The first file is the DBECon file or DBEnvironment configuration file. This has the same name you assigned to the DBEnvironment in the START DBE command. The DBECon file contains startup parameters for the DBEnvironment.

DBEFILE0 is a file containing the data for the SYSTEM DBEFileSet, which contains the system catalog. (You'll examine the system catalog later.)

DBELOG1 is the log file, which records operations that modify the database.

Once you have configured the DBEnvironment, set the timeout time, and set a maintenance word for it, you are ready to attach TurboIMAGE/XL databases to it, as shown in the next task.

Task Reference

- No information is given here about considerations that may be necessary when configuring a DBEnvironment that contains native ALLBASE/SQL databases. Consult the *ALLBASE/SQL Database Administration Guide* for more guidance in this area.
- The following summaries of IMAGE/SQL commands will get you started. Refer to the *ALLBASE/SQL Reference Manual* for more information.
  - □ The ALLBASE/SQL START DBE NEW statement has the following syntax:

```
START DBE 'DBEnvironmentName' [MULTI] NEW
 ┌DUAL LOG                                  ┐
 │BUFFER = (DataBuffPages,LogBuffPages)     │
 │LANG = LanguageName                       │ |, ... |
 │TRANSACTION = MaxTransactions             │
 │DBEFile0Definition                        │
 └DBELogDefinition                          ┘
```

    Refer to the Performing the Task example for details of the syntax for *DBEFile0Definition* and *DBELogDefinition*.

  - □ Use SQLUtil to set a DBEnvironment maintenance word. To access SQLUtil, issue the following command:

```
:RUN SQLUTIL.PUB.SYS
```

The SQLUtil SETDBEMAINT command is used to set a maintenance word. The syntax for this command is as follows:

```
>> SETDBEMAINT

DBEnvironment Name:  DBEnvironmentName
Current Maintenance Word:  OldMaintenanceWord
New Maintenance Word:  NewMaintenanceWord
Retype New Maintenance Word:  NewMaintenanceWord
>>EXIT
```

When no current maintenance word exists, enter a carriage return at the "Current Maintenance Word:" prompt.

Attaching a TurboIMAGE/XL Database

This task describes how to attach a TurboIMAGE/XL database.

Note

If you are attaching a database with the same name but in a different group and/or account from a database already attached to the DBEnvironment, you must specify an alternative owner name at attach time (see Task Reference). This is because in a mapped table, by default, the owner name is the database name, and duplicate table names are not allowed within the same database. In any case, you cannot attach the same TurboIMAGE/XL database twice to the same DBEnvironment.

Getting Ready

- It is convenient to have the TurboIMAGE/XL database and the DBEnvironment in the same group and account.

  If this is not the case, there are several issues to consider. The following considerations apply if the TurboIMAGE/XL database(s) to be attached exist(s) in a different group and/or account than the DBEnvironment:

  - IMAGE/SQL supports standard MPE/iX security rules. Correct user, group, and account capabilities must be in place to use IMAGE/SQL to access a TurboIMAGE/XL database from a DBEnvironment in a different account than the database.
  - IMAGE/SQL utility administrators often need to specify DBEnvironment and TurboIMAGE/XL maintenance words as a part of the SET command if they are not the creator. Because of this, it is recommended that maintenance words exist for both database management systems.
  - IMAGE/SQL utility administrators need DBA authority to perform most IMAGE/SQL utility tasks. Be sure to grant DBA authority to everyone who will be performing IMAGE/SQL utility tasks (see Task Reference).
- The TurboIMAGE/XL database and the DBEnvironment must have the same native language support (NLS) defined for them.
- Be sure that all processes accessing the DBEnvironment are shutdown before using the ATTACH command. The ATTACH command using the IMAGE/SQL utility requires exclusive access to the DBEnvironment.
- The ATTACH command requires that the appropriate SET SQLDBE and SET TURBODB commands have been issued. To check the status of these commands, use the DISPLAY OPTIONS command:

```
:RUN IMAGESQL.PUB.SYS

HP36385B X.F0.12               IMAGE/SQL Utility   THU, APR  8, 1993, 10:30 AM
(C) COPYRIGHT HEWLETT-PACKARD COMPANY 1993

>>DISPLAY OPTIONS
Current base    :
Current SQLDBE  :
Echo            : ON
Command Logging : ON
Log File        : ATCLOG.PUB.TURBONM
```

- Because there is no current database or DBEnvironment, issue the SET command as follows:

```
>>SET SQLDBE MUSICDBE.PUB.TURBONM
>>SET TURBODB MUSIC.PUB.TURBONM
>>
```

- Another DISPLAY OPTIONS command confirms that there is now a current database and DBEnvironment:

```
>>DISPLAY OPTIONS
Current base    : MUSIC.PUB.TURBONM
Current SQLDBE  : MUSICDBE.PUB.TURBONM
Echo            : ON
Command Logging : ON
Log File        : ATCLOG.PUB.TURBONM
```

Performing the Task

Once you are sure the appropriate SET command have been specified and that the correct MPE/iX security is in place, issue the IMAGE/SQL utility ATTACH command.

```
>>ATTACH
Split 1 compound source fields (ATCWARN 32063).
Mapped 15 source table/source field names (ATCWARN 32062).
Mapped 1 incompatible source types (ATCWARN 32061).
>>
```

Messages issued at attach time inform you if any mapping has been done (see Task Reference). The MUSIC database is now a logical part of the MusicDBE DBEnvironment. Although the data remains in the TurboIMAGE/XL database, it can now be selected from mapped tables just as it would be selected from ALLBASE/SQL tables.

Task Reference

- By default, the IMAGE/SQL utility uses the TurboIMAGE/XL database name as the owner name. You must specify an alternative owner name if you are attaching a TurboIMAGE/XL database with the same name as one already attached. To do this, use the WITH OWNER= parameter of the ATTACH command.
- To specify a maintenance word, use the MAINT= parameter of the SET TURBODB or SET SQLDBE command. Refer to the *HP IMAGE/SQL Administration Guide* for details about these commands and their parameters.
- Use the DISPLAY MAP command to see detailed database mapping information.
- When a TurboIMAGE/XL database is attached to a DBEnvironment, IMAGE/SQL performs the following tasks:
  - Makes a table entry in the system catalog of the DBEnvironment for each corresponding source data set.
  - Creates a column definition for each field in the source data set. (The NOT NULL option is used because null values are not defined in TurboIMAGE/XL.)
  - Produces default mapping information which maps TurboIMAGE/XL data sets to SQL tables and stores this information in the ATCINFO file. Specifically, mapping is done in the following areas:

- Data item and data set names

  Some characters allowed in TurboIMAGE/XL names (specifically: + – * / ? ' % &) are not valid SQL names. Therefore, whenever IMAGE/SQL utility encounters such a character in a TurboIMAGE/XL name, it converts it to an underscore (_).

- Data types

  TurboIMAGE/XL data types are mapped to SQL data types. When inexact or imprecise mapping is necessary, an 'I' appears in the NOTES section of the DISPLAY MAP display. When a compound field is split into separate mapped columns, an S appears in the NOTES section of the display.

- User security

  Initially, only the TurboIMAGE/XL database creator (DBC) is defined as a user in the DBEnvironment. For other users to access the attached database, the DBC must add users with IMAGE/SQL utility's ADD USER command.

- Once the database is attached, the DBC must add any additional IMAGE/SQL users.
- It is desirable to update data types and split mapped columns before IMAGE/SQL users access the attached database. For details, refer to the *HP IMAGE/SQL Administration Guide*. This is because whenever a mapped column is split or updated, any user-created views containing these mapped columns are dropped.

Detaching a TurboIMAGE/XL Database

This task describes how to detach a TurboIMAGE/XL database from a DBEnvironment.

Getting Ready

Before detaching a TurboIMAGE/XL database, do the following:

- Obtain exclusive access to the DBEnvironment.
- Make sure the appropriate SET SQLDBE and SET TURBODB commands have been issued. To check the status of these commands, use the DISPLAY OPTIONS command:

```
>>DISPLAY OPTIONS
Current base    :
Current SQLDBE  :
Echo            : ON
Command Logging : ON
Log File        : ATCLOG.PUB.TURBONM
>>
```

- If necessary, issue the SET command. For example:

```
>>SET SQLDBE MUSICDBE.PUB.TURBONM
>>SET TURBODB MUSIC.PUB.TURBONM
>>
```

Performing the Task

Once you have issued the appropriate SET commands, you are ready to detach the database from the DBEnvironment. For example:

```
>>DETACH
>>
```

This command detaches the MUSIC database from the MusicDBE DBEnvironment. If MUSIC is the only database attached to MusicDBE, the ATCINFO file will be deleted. If MusicDBE is the only DBEnvironment to which MUSIC is attached, the MUSICTC file will also be deleted.

In addition, all mapped table definitions are removed from the DBEnvironment system catalog. All IMAGE/SQL views and user-created views based on IMAGE/SQL tables and views are dropped.

Task Reference

- There are several reasons to detach a TurboIMAGE/XL database from a DBEnvironment:
  - If the database is to be restructured. (This includes restructuring with DBChange and other third-party restructuring tools.)
  - If ALLBASE/SQL is to be migrated back to a version that does not support IMAGE/SQL.
  - If you wish to reset all mapping information to default values.
- All mapped tables, all IMAGE/SQL views based on these tables, and all user-created views based on IMAGE/SQL views and tables are dropped when the database is detached.
- The ATCINFO file is purged when it no longer contains mapping information about any databases.
- The *DBaseName*TC file is deleted when the database is no longer attached to any DBEnvironments.

Displaying IMAGE/SQL Utility Information

This task describes how to display IMAGE/SQL information using several IMAGE/SQL utility commands. These commands include:

DISPLAY OPTIONS	Displays the options in effect for your current IMAGE/SQL utility session.
DISPLAY TURBODB	Displays all the TurboIMAGE/XL databases associated with a specific DBEnvironment.
DISPLAY MAP	Displays the current data type mapping information for a specific TurboIMAGE/XL database.
DISPLAY SQLDBE	Displays all the DBEnvironments associated with a specific TurboIMAGE/XL database.
DISPLAY USER	Displays the current information about users in a specific TurboIMAGE/XL database.
HELP	Provides the syntax of IMAGE/SQL utility commands.

Getting Ready

- The DISPLAY command provide two options: Either issue SET commands before displaying information, or specify a particular TurboIMAGE/XL database or DBEnvironment as part of the DISPLAY command.
- For DISPLAY SQLDBE and DISPLAY TURBODB to display useful information, at least one database should be attached.
- For DISPLAY MAP AND DISPLAY USER to display useful information, a database should be attached.

Performing the Task

Examples using the DISPLAY command are shown on the following pages.

Example 1: Displaying Database Information

To see all the TurboIMAGE/XL databases and mapped tables currently associated with MusicDBE, use the DISPLAY TURBODB command.

```
:RUN IMAGESQL.PUB.SYS
>>DISPLAY TURBODB TABLES FOR MUSICDBE.PUB.TURBONM

DBEnvironment    : MUSICDBE.PUB.TURBONM

TURBOIMAGE/XL DATABASE   OWNER           MAPPED TABLE
----------------------   -----           ------------

MUSIC.PUB.TURBONM        MUSIC           ALBUMS
                                         COMPOSERS
                                         SELECTIONS_A
                                         SELECTIONS
                                         LOG

Total Databases : 1
>>
```

Any databases attached to the DBEnvironment and their associated mapped tables are displayed.

Example 2: Displaying Database Mapping Information

The DISPLAY MAP command shows how TurboIMAGE/XL data sets are mapped to ALLBASE/SQL tables. In the following example, information about the mapped table ALBUMS is displayed.

```
>>DISPLAY MAP ALBUMS

ATTACHED BASES   : MUSIC.PUB.TURBONM
DBEnvironment    : MUSICDBE.PUB.TURBONM
Owner Name       : MUSIC

MAPPED(SOURCE)   SOURCE             MAPPED          SOURCE     MAPPED
   TABLE         FIELD              COLUMN           TYPE       TYPE       NOTES
------------ ---------------- -------------------- ------- -------------- -----

ALBUMS (ALBUMS)
             ALBUMCODE        ALBUMCODE            I2      INTEGER
             ALBUMTITLE       ALBUMTITLE           X40     CHAR(40)
             MEDIUM           MEDIUM               X2      CHAR(2)
             ALBUMCOST        ALBUMCOST            P8      DECIMAL(7,0)
             RECORDINGCO      RECORDINGCO          X10     CHAR(10)
             DATERECORDED     DATERECORDED         X16     CHAR(16)
             MFGCODE          MFGCODE              X40     CHAR(40)
             COMMENT          COMMENT              X80     CHAR(80)
```

Example 3: Displaying DBEnvironment Information

To see the ALLBASE/SQL DBEnvironment currently associated with MusicDBE, use the DISPLAY SQLDBE command.

```
>>DISPLAY SQLDBE

ATTACHED BASES   : MUSIC.PUB.TURBONM

DBENVIRONMENT
-------------
MUSICDBE.PUB.TURBONM
```

Example 4: Displaying User Information

To see the current user information associated with MusicDBE, use the DISPLAY USER command.

```
>>DISPLAY USER

ATTACHED BASES   : MUSIC.PUB.TURBONM
DBEnvironment    : MUSICDBE.PUB.TURBONM

USER LOGON         DBOPEN MODE  USER PASSWORD  USER CLASS
----------         -----------  -------------  ----------

MGR@TURBONM        5            ;              64
```

Issuing MPE/iX Commands from the IMAGE/SQL Utility

This task describes how to issue MPE/iX commands from the IMAGE/SQL utility.

Performing the Task

To issue an MPE/iX command, at the IMAGE/SQL utility prompt enter a colon (:) and the name of the MPE/iX command you wish to issue.

For example, to issue the LISTF command:

```
>>:LISTF

FILENAME

ATCINFO     DBEFILE0    DBELOG1     DBUSA       DBUSA01     DBUSA02
DBUSA03     DBUSASCH    DBUTIL      MUSIC       MUSICDBE    MUSICTC

>>
```

The IMAGE/SQL utility prompt returns after the MPE/iX command is executed.

Setting IMAGE/SQL File Equations

This task describes how to set IMAGE/SQL utility file equations for two files: ATCINFO and ATCLOG.

Performing the Task

Setting a File Equation for ATCINFO

An ATCINFO file equation can only be set before the file is created (before any TurboIMAGE/XL databases are attached to the DBEnvironment; see Task Reference). Only the file name can be specified. Other parameters of the FILE command will not be valid at attach time.

For example:

```
:FILE ATCINFO = SONGSCR
:
```

Setting a File Equation for ATCLOG

A file equation for ATCLOG can be issued before beginning an IMAGE/SQL utility session.

For example:

```
:FILE ATCLOG = LOGA1; SAVE
:
```

In the above example, when the logging option is on, IMAGE/SQL Utility commands are logged to the permanent file, LOGA1.

A file equation for ATCLOG can also be set or reset from within the IMAGE/SQL utility.

For example:

```
>>:FILE ATCLOG = LOGA2; SAVE
>>
```

Now when the logging option is on, IMAGE/SQL utility commands are logged to the permanent file, LOGA2 (see Task Reference).

ATCINFO Reference

- ATCINFO is a permanent privileged file containing mapping information about data set and field names, data types, and user security. Its formal file designator is ATCINFO. The default name of the ATCINFO file is *DBEnvironment-name*CR.
- One ATCINFO file exists for each DBEnvironment. It is created in the same group and account as the DBEnvironment and is considered part of the DBEnvironment.
- The ATCINFO file equation can only be used to specify a different file name. It cannot be used to override other ATCINFO file characteristics.

ATCLOG Reference

- ATCLOG is a temporary unnumbered ASCII file to which all IMAGE/SQL utility commands are written when IMAGE/SQL utility logging is on. If this file does not already exist, it is created. If it already exists as a temporary or permanent file, log file records are appended to it. Its formal file designator is ATCLOG.
- When an IMAGE/SQL utility session begins, logging is on and the ATCLOG file is created (if it does not already exist) and opened in the user's logon group and account. By default, it is a temporary file named ATCLOG, but you may wish to set a file equation for this file.
- If you want to save the temporary file, either specify SAVE as a part of the FILE command or save the file before ending your current MPE/iX session.
- The FILE command can be used to override default ATCLOG file attributes, such as size and file domain. However, the log file must remain an unnumbered ASCII file.

Logging IMAGE/SQL Utility Commands

This task describes how to log IMAGE/SQL utility commands and how to save frequently used command sequences in different log files so they can easily be reissued in batch or interactive mode.

Performing the Task

By default, IMAGE/SQL utility commands are logged to an ASCII file, ATCLOG, which you can read and edit. You can change log files within IMAGE/SQL utility by issuing a file equation for ATCLOG. For example, to rename and save the IMAGE/SQL utility log file as a permanent file, set a file equation for the ATCLOG file before the IMAGE/SQL utility is run:

```
:FILE ATCLOG = DOATTACH; SAVE
:
```

When the IMAGE/SQL utility is run, the DISPLAY OPTIONS command confirms that commands are being logged to DOATTACH.

```
>>DISPLAY OPTIONS
Current base       :
Current SQLDBE     :
Echo               : ON
Command Logging    : ON
Log File           : DOATTACH.PUB.TURBONM

*** Database is not attached.
>>
```

The MUSIC database is then attached to the MusicDBE DBEnvironment:

```
>>SET TURBODB MUSIC.PUB.TURBONM
>>SET SQLDBE MUSICDBE.PUB.TURBONM
>>ATTACH
>>
```

After the attach, the contents of the log file DOATTACH look like this:

```
DISPLAY OPTIONS
SET TURBODB MUSIC.PUB.TURBONM
SET SQLDBE MUSICDBE.PUB.TURBONM
ATTACH
```

To close this log file and write to a new one, issue another MPE/iX FILE command from within IMAGE/SQL utility. For example:

```
>>:FILE ATCLOG=UPDATYPE;SAVE
>>
```

The DISPLAY OPTIONS command confirms that commands are now being logged to UPDATYPE:

```
>>DISPLAY OPTIONS
Set Turbodb      : MUSIC.PUB.TURBONM
Set Sqldbe       : MUSICDBE.PUB.TURBONM
Echo             : ON
Command Logging  : ON
Log File         : UPDATYPE.PUB.TURBONM

*** Database is attached.
```

Any IMAGE/SQL utility commands now issued are written to UPDATYPE. For example, when the following command is issued,

```
>> UPDATE TYPE IN ALBUMS.DATERECORDED TO CHAR(20)
Updated information in table ALBUMS.
>>
```

the log file, UPDATYPE, contains the following commands:

```
DISPLAY OPTIONS
UPDATE TYPE IN ALBUMS.DATERECORDED TO CHAR(20)
```

Using this technique, you can create several files containing often-used IMAGE/SQL utility commands. The commands in these files can then be executed, using the XEQ command (see Task Reference).

Log files can be edited with a text editor. (Be sure to keep the edited file unnumbered or it cannot be used as a command file.)

Note

Many editors automatically insert an end-of-file marker at the end of the text in an edited file. This prevents the file from having log records appended to it. For this reason, if the original file is to be used as a log file, you should keep the edited file under a different name.

Task Reference

- The section “Setting IMAGE/SQL File Equations” in this chapter contains more information about issuing IMAGE/SQL file equations.
- The section “Using IMAGE/SQL Utility Command Files” in this chapter contains more information about using the IMAGE/SQL utility XEQ command interactively or in batch to execute IMAGE/SQL utility commands.
- When an IMAGE/SQL utility session begins, logging is on. The temporary file, ATCLOG, is created (if it does not already exist) and opened in the user’s logon group and account. If the log file already exists as a temporary or permanent file, log records are appended to it.
- If you want to save the temporary file, either save the file before ending your current MPE/iX session or specify ;SAVE as a part of the FILE equation.
- The FILE command can be used to override default ATCLOG file attributes, such as size and file domain. However, the log file must remain an unnumbered ASCII file.

Using IMAGE/SQL Utility Command Files

This task describes how to execute an IMAGE/SQL utility command file in interactive or batch mode.

Getting Ready

An IMAGE/SQL utility command file can be created with a text editor or can be produced as part of the logging process (see Task Reference). If you create or edit this file with an editor, note that it must be kept unnumbered.

In the following examples, the file UPDATYPE contains the following commands:

```
   SET TURBODB MUSIC.PUB.TURBONM
   SET SQLDBE MUSICDBE.PUB.TURBONM
   UPDATE TYPE IN ALBUMS.DATERECORDED to CHAR(20)
   EXIT
```

Note that the commands in this command file assume the TurboIMAGE/XL database is already attached to the DBEnvironment.

If you wish to see commands and comments as the command file is executed, make sure the ECHO option is on (see Task Reference).

Performing the Task

The XEQ command allows you to specify a file containing IMAGE/SQL utility commands as its parameter. To interactively execute commands listed in a command file, run the IMAGE/SQL utility by typing RUN IMAGESQL.PUB.SYS and then issue the XEQ command. Note that in this example ECHO is off.

```
:RUN IMAGESQL.PUB.SYS

HP36385 B.F0.10           IMAGE/SQL Utility        FRI, DEC 18, 1992, 11:30 AM
(C) COPYRIGHT HEWLETT-PACKARD COMPANY 1992

>>XEQ UPDATYPE
Updated information in table ALBUMS.
:
```

The MUSIC database, previously attached to the MusicDBE DBEnvironment, and data type mapping information has been updated for the ALBUMS.DATERECORDED column of the ALBUMS table.

You can also issue XEQ commands in batch mode. The following job stream file contains XEQ commands that execute the commands in the UPDATYPE file.

```
!job JIMAGESQL,USER2/KEVIN.ATC/MGR,PUB/ALL
!comment**************************************************************
!comment*      This job executes an IMAGE/SQL command file.
!comment**************************************************************
!
!tell USER2.ATC;   /-->Start JIMAGESQL for MUSIC
!
!run IMAGESQL.PUB.SYS
!
!comment**************************************************************
!comment*      The UPDATYPE command file contains commands that SET
!comment*      the MUSIC database and the MusicDBE DBEnvironment.
!comment*      It then specifies alternative data type mapping for source
!comment*      data set fields and exits the IMAGE/SQL Utility.
!comment**************************************************************
!
XEQ UPDATYPE
!
!tell USER2.ATC;   /-->End JIMAGESQL for MUSIC
!
!eoj
```

Note that in batch mode, if an error occurs, the job terminates. The remaining commands are flushed.

Task Reference

- An IMAGE/SQL command file is an unnumbered file containing a list of IMAGE/SQL commands. If commands span more than one line, use an ampersand (&) to continue the command to the next line.
- The section "Logging IMAGE/SQL Utility Commands" shows how to use the IMAGE/SQL utility logging facility to create and save files containing often-issued IMAGE/SQL utility commands.
- For the syntax of the ECHO command, refer to the *HP IMAGE/SQL Administration Guide*.

Accessing TurboIMAGE/XL Data with ISQL

This task briefly describes how IMAGE/SQL users access TurboIMAGE/XL data with ISQL.

Getting Ready

To successfully select TurboIMAGE/XL data with ISQL, IMAGE/SQL users need to know the following:

- How to use the IMAGE/SQL interface provided for them. The examples in this manual use ISQL, which also requires familiarity with the SQL SELECT statement.
- The names of the mapped tables and/or views to which they have access (see Task Reference).
- Which columns map to TurboIMAGE/XL search items. Under certain circumstances, using these mapped columns when selecting data can improve performance (see Task Reference).
- Which data is of type FLOAT. When selecting this data, users should specify a range of values rather than a particular number. This is necessary because some precision is lost when converting to FLOAT.

Performing the Task

Because data has not yet been added to the MUSIC database, using a SELECT statement at this point would not demonstrate its use. If you want to see SELECT statements using the MUSIC database, see the section "Selecting Data" in Chapter 6, "Practicing with IMAGE/SQL Using MusicDBE."

Task Reference

- The structure of the TurboIMAGE/XL database cannot be changed with SQL statements. Therefore, SQL statements that alter the structure of the database are not available to IMAGE/SQL users.
- When users have access to the entire data entry, they can select data from the table itself. If they do not have access to the entire data entry, they must select data from a view of the table created for them by IMAGE/SQL. Table names are of the form *OwnerName.MappedTableName*. View names are of the form *OwnerName.MappedTableName_V UserClass#*.
- For examples of using the SELECT statement on the MUSIC database, refer to Chapter 6, "Practicing with IMAGE/SQL Using MusicDBE."
- For detailed information about using the SELECT statement, refer to the *ALLBASE/SQL Reference Manual*.

Maintaining the ATCINFO File

This task describes how to maintain the ATCINFO file.

Getting Ready

Maintenance for the ATCINFO file (*DBEnvironment-name*CR) may be necessary in either of the following situations:

- If a crash occurs while the ATCINFO file is being modified. This may be the case if a crash takes place when the IMAGE/SQL administrator is in the midst of an IMAGE/SQL utility command that updates the ATCINFO file. When this occurs, the RECOVER command can be used to reconstruct the ATCINFO file.
- If the ATCINFO file contains too much free space. This may be the case if multiple databases are attached and then some are detached from the DBEnvironment. When this occurs, the RECOVER statment can be used to compact the data in the ATCINFO file and return the free space to the file system.

Performing the Task

To recover the ATCINFO file associated with the MusicDBE DBEnvironment, issue the following commands:

```
:RUN IMAGESQL.PUB.SYS
>>SET SQLDBE MUSICDBE
>>RECOVER
Checking physical file consistency and recovering free space.
Deleting unused mapped table entries.
Checking external cross references.
>>
```

Task Reference

- For more information on the RECOVER command, refer to the *HP IMAGE/SQL Administration Guide.*

6

Practicing with IMAGE/SQL Using MusicDBE

IMAGE/SQL software includes scripts and files for creating a sample database and DBEnvironment. This chapter shows you how to execute the script and create the TurboIMAGE/XL MUSIC database and MusicDBE DBEnvironment. From there, you can examine the database and insert and select data. After completing this chapter, you should be more comfortable with the IMAGE/SQL environment.

The topics covered in this chapter are:

- Setting up MUSIC and MusicDBE
- Examining the database
- Adding data
- Selecting data
- Modifying data
- Deleting data

Setting Up MUSIC and MusicDBE

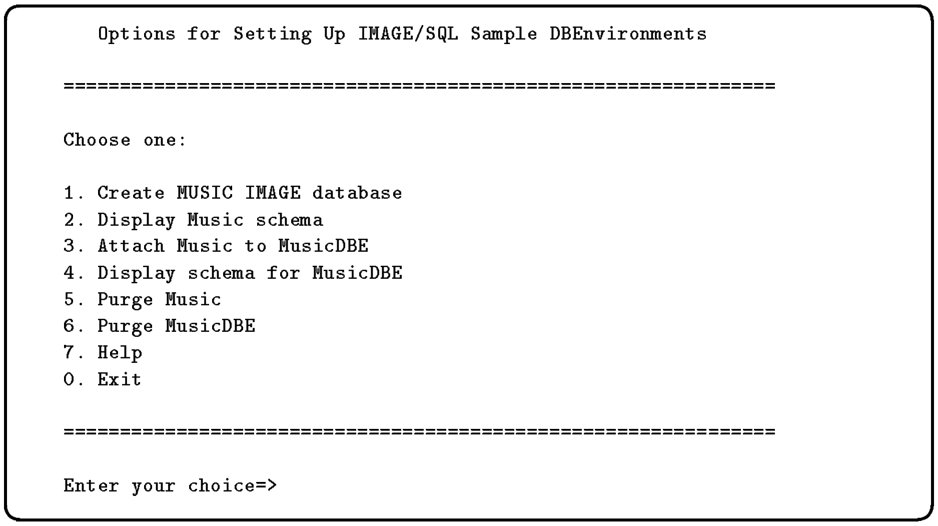
This section shows you the steps to set up the database and DBEnvironment.

Before beginning, change into the group and account where you want to create MUSIC and MusicDBE. For clarity, use an empty group, if possible. This chapter uses the group PUB and the account ACCOUNT.

A script is provided with the IMAGE/SQL product that simplifies the database creation and purging processes. To execute the script, type:

```
:RUN IMSQL.SAMPLEDB.SYS  Return
```

A menu like the one in Figure 6-1 appears on your screen (some details may differ on your system).



```
    Options for Setting Up IMAGE/SQL Sample DBEnvironments

================================================================

Choose one:

1. Create MUSIC IMAGE database
2. Display Music schema
3. Attach Music to MusicDBE
4. Display schema for MusicDBE
5. Purge Music
6. Purge MusicDBE
7. Help
0. Exit

================================================================

Enter your choice=>
```

Figure 6-1. SQLSetup Menu

The options are described in Table 6-1.

Table 6-1. Menu Options

Option	Description
1	Creates the TurboIMAGE/XL MUSIC database in your current group and account.
2	Displays the MUSIC schema.
3	Uses IMAGE/SQL utility to attach MUSIC to MusicDBE.
4	Displays catalog information about MusicDBE.
5	Purges MUSIC. You must be the Creator or System Manager to use this option.
6	Purges MusicDBE. You must be the DBECreator or System Manager to use this option.
7	Displays the current help screen.
0	Exists IMSQL.

From the above menu, select Option 1 to create the TurboIMAGE/XL MUSIC database in your group and account.

Next, select Option 3. This will execute IMAGESQL.PUB.SYS, set the TURBODB parameter to MUSIC, and set the SQLDBE parameter to the non-existent MusicDBE. The DBE will be automatically created for you at this time. This option then attaches MUSIC to MusicDBE.

When you return to the menu, select Option 0 to exit.

After completing Options 1 and 3, list the files in the current group and account. You will find the following files in your group:

```
FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
    SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

MUSIC     PRIV    128W  FB          8          8   1        16  1  1
MUSIC01   PRIV    512W  FB        250        250   1      1008  1  4
MUSIC02   PRIV    384W  FB        250        250   1       752  1  4
MUSIC03   PRIV    384W  FB        100        100   1       304  1  2
MUSIC04   PRIV    512W  FB        250        250   1      1008  1  4
MUSIC05   PRIV    512W  FB        143        143   1       576  1  3
MUSICDBE  PRIV   2048W  FB          3         11   1        48  1  *
MUSICDCR  PRIV   2048W  FB        128     131072   1       256  1  *
MUSICDFL  PRIV   2048W  FB       1000       1000   1     16000  1 32
MUSICDLG  PRIV    256W  FB       1000       1000   1      2000  1 32
MUSICTC   PRIV     28W  FB          1       1023   4       208  1  *
```

The files are all PRIV files, which means you cannot purge them without special system authority. The DBUTIL and SQLUtil utilities provide this authority.

Examining The Database

In this section, you will examine the objects that were created within MusicDBE—tables, views, and authority structures. Information about all these objects is in the system catalog, which is automatically created by IMAGE/SQL when the DBEnvironment is created and configured.

Run ISQL, then connect to MusicDBE using this CONNECT command:

```
isql=>CONNECT TO 'musicdbe'; Return
```

Now examine the system catalog by creating queries on the system views. Use the following query exactly as shown to look at all the tables and views created by the attach function. Type the query exactly as shown because the owner name is case-sensitive.

```
isql=>SELECT NAME, OWNER, Return
>DBEFILESET, TYPE Return
>FROM SYSTEM.TABLE Return
>WHERE OWNER = 'MUSIC'; Return
```

The result table is shown below.

```
select name, owner, dbefileset, type from system.table where owner = 'MUSIC
--------------------+--------------------+--------------------+------
NAME                |OWNER               |DBEFILESET          |TYPE
--------------------+--------------------+--------------------+------
ALBUMS              |MUSIC               |SYSTEM              |     0
COMPOSERS           |MUSIC               |SYSTEM              |     0
LOG                 |MUSIC               |SYSTEM              |     0
SELECTIONS          |MUSIC               |SYSTEM              |     0
SELECTIONS_A        |MUSIC               |SYSTEM              |     0
SELECTIONS_A_VO     |MUSIC               |SYSTEM              |     1
---------------------------------------------------------------------------
Number of rows selected is 6
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >e
```

Figure 6-2. Information on Tables and Views

Each table is listed in the NAME column. The OWNER column specifies the database to which the table belongs. If you are not the table owner (that is, if it is part of the TurboIMAGE/XL database), you must prefix the table name with its owner name whenever you refer to it. In this example, a reference to the ALBUMS table would be MUSIC.ALBUMS.

The DBEFILESET column contains the name of the DBEFileSet an entry has been associated with. IMAGE/SQL tables are always associated with the SYSTEM DBEFileSet.

The TYPE column indicates whether the entry is a table or a view. Entries with type 0 are tables and entries with type 1 are views.

Examining Table Descriptions

You can see individual table descriptions by using the INFO command, which returns the column definition of a table. Use the following command for the Albums table:

```
isql=>INFO music.albums; (Return)
```

The output from this ISQL command is shown below:

```
Column Name          Data Type (length)     Nulls Allowed      Language
--------------------------------------------------------------------------
ALBUMCODE            Integer                   NO
ALBUMTITLE           Char      (   40)         NO              NATIVE-3000
MEDIUM               Char      (    2)         NO              NATIVE-3000
ALBUMCOST            Decimal   (    7,    0)   NO
RECORDINGCO          Char      (   10)         NO              NATIVE-3000
DATERECORDED         Char      (   16)         NO              NATIVE-3000
MFGCODE              Char      (   40)         NO              NATIVE-3000
COMMENT              Char      (   80)         NO              NATIVE-3000
```

Figure 6-3. Column Definitions of a Table

The Column Name column lists the names of all the columns in the table.

The Data Type column shows the SQL data type for each column and its size (in parentheses).

The third column, Nulls Allowed, indicates whether or not null values are permitted in this column. IMAGE/SQL tables do not allow null values, so this column will always contain NO.

The Language column indicates which language is applicable for the column if it is a character type. This corresponds with the Language value in the MUSIC root file.

Examining the Authority Structure

An authority structure consists of many elements. Some of these elements are shown below:

- Group definitions
- Table authorization for select, insert, update, and delete operations on tables
- **Column authorization** for permission to update specific columns

Use the following query to examine each **authorization group** in MusicDBE and their members:

```
isql=>SELECT * FROM SYSTEM.GROUP; (Return)
```

The query result is shown below:

```
select * from system.group;
--------------------+--------------------+--------------------+-----------
USERID              |GROUPID             |OWNER               |NMEMBERS
--------------------+--------------------+--------------------+-----------
MUSIC_0             |MUSIC_0             |MGR@ACCOUNT         |          0
MUSIC_64            |MUSIC_64            |MGR@ACCOUNT         |          0



-------------------------------------------------------------------------
Number of rows selected is 2
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] > e
```

Figure 6-4. Groups in the System Catalog

Figure 6-4 shows two UserID values, MUSIC_0 and MUSIC_64, that correspond with TurboIMAGE/XL password levels of 0 and 64. Level 0 is assigned to items and sets that do not have levels assigned to them. Users accessing the database without a password are given this level. Level 64 is the creator level. Notice that the OWNER for both is MGR@ACCOUNT, the database creator.

Passwords in TurboIMAGE/XL are case-sensitive. Also, you need to allow these users mode 1 access to TurboIMAGE/XL to permit database updates.

The information above is set up by the ATTACH command. There are, however, three levels of security defined in the TurboIMAGE/XL database: MGR, DIR, and ANNCR. To use these levels, you must add additional users with the IMAGE/SQL utility as shown below:

```
:RUN IMAGESQL.PUB.SYS Return

HP36385B X.F0.13              IMAGE/SQL Utility   SAT, JUN  5, 1993,  6:23 PM
(C) COPYRIGHT HEWLETT-PACKARD COMPANY 1993

>>SET TURBODB music  Return
>>SET SQLDBE musicdbe  Return
>>UPDATE USER MGR@ACCOUNT TO PASS = MGR, MODE=1 Return
Warning: command containing a password has been logged (ATCWARN 32069).
ALLBASE/SQL group MUSIC_10 created.
View MUSIC.ALBUMS_V10 created.
View MUSIC.COMPOSERS_V10 created.
View MUSIC.SELECTIONS_A_V10 created.
View MUSIC.SELECTIONS_V10 created.
View MUSIC.LOG_V10 created.

>>ADD USER DIR@ACCOUNT WITH PASS = DIR, MODE=1 Return
Warning: command containing a password has been logged (ATCWARN 32069).
ALLBASE/SQL group MUSIC_20 created.
View MUSIC.ALBUMS_V20 created.
View MUSIC.COMPOSERS_V20 created.
View MUSIC.SELECTIONS_A_V20 created.
View MUSIC.SELECTIONS_V20 created.
View MUSIC.LOG_V20 created.

>>ADD USER ANNCR@ACCOUNT WITH PASS = ANNCR, MODE=1 Return
Warning: command containing a password has been logged (ATCWARN 32069).
ALLBASE/SQL group MUSIC_30 created.
View MUSIC.ALBUMS_V30 created.
View MUSIC.COMPOSERS_V30 created.
View MUSIC.SELECTIONS_A_V30 created.
View MUSIC.SELECTIONS_V30 created.
View MUSIC.LOG_V30 created.
```

You can view these changes and additions with the DISPLAY USERS command:

```
>>DISPLAY USERS Return
```

```
ATTACHED BASES   : MUSIC.PUB.ACCOUNT
DBEnvironment    : MUSICDBE.PUB.ACCOUNT

USER LOGON         DBOPEN MODE  USER PASSWORD  USER CLASS
----------         -----------  -------------  ----------

MGR@ACCOUNT        1            MGR            10
DIR@ACCOUNT        1            DIR            20
ANNCR@ACCOUNT      1            ANNCR          30
```

Figure 6-5. Changes to Levels of Security

Re-invoke ISQL and enter the following command to look at the UserID information:

```
isql=>SELECT * FROM system.group; Return
```

```
select * from system.group;
--------------------+--------------------+--------------------+-----------
USERID              |GROUPID             |OWNER               |NMEMBERS
--------------------+--------------------+--------------------+-----------
MUSIC_0             |MUSIC_0             |MGR@ACCOUNT         |          0
MUSIC_64            |MUSIC_64            |MGR@ACCOUNT         |          0
MUSIC_10            |MUSIC_10            |MGR@ACCOUNT         |          0
MUSIC_20            |MUSIC_20            |MGR@ACCOUNT         |          1
DIR@ACCOUNT         |MUSIC_20            |MGR@ACCOUNT         |          0
MUSIC_30            |MUSIC_30            |MGR@ACCOUNT         |          1
ANNCR@ACCOUNT       |MUSIC_30            |MGR@ACCOUNT         |          0





---------------------------------------------------------------------------
Number of rows selected is 7
```

Figure 6-6. UserID Information

New UserID values now exist for DIR and ANNCR. However, no UserID exists for MGR because MGR is the OWNER of these GroupIDs and UserIDs. Now that all these UserIDs are established, you can access the database using any of these MPE user logins.

Adding Data to the Database

The next step is to add data to the database. Because no applications exist for this database, your choice for adding data is with the interactive tools that come with the product. For standard access to TurboIMAGE/XL, your tool is QUERY. For SQL access to IMAGE/SQL, your tool is ISQL. You can add data using the SQL INSERT statement or ISQL LOAD command.

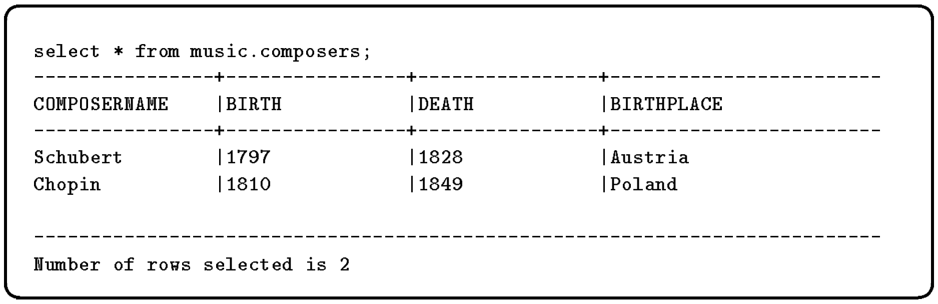
In Chapter 4, "A Database Example", the concept of referential integrity was presented. That concept is further demonstrated here. If you refer to the database design in Chapter 4, selection table entries are dependent upon corresponding values in the Composers and Albums tables. To maintain referential integrity, you will add data in the following order:

1. Composers and Albums
2. Selections

Using the INSERT Statement

You can use ISQL to CONNECT and then INSERT data into a table. The following example adds rows to the COMPOSERS table with the INSERT statement, then displays the table by reading all the data with the SELECT statement:

```
:RUN ISQL.PUB.SYS  Return

isql=>CONNECT TO 'musicdbe'; Return
isql=>INSERT INTO music.composers Return
>values ('Schubert','1797','1828','Austria','Died at age 31');Return

Number of rows processed is 1
isql=>INSERT INTO music.composers Return
>values ('Chopin','1810','1849','Poland','Died at age 39');Return

Number of rows processed is 1
isql=>COMMIT WORK; Return
isql=>SELECT * FROM music.composers; Return
```



```
select * from music.composers;
----------------+----------------+----------------+------------------------
COMPOSERNAME    |BIRTH           |DEATH           |BIRTHPLACE
----------------+----------------+----------------+------------------------
Schubert        |1797            |1828            |Austria
Chopin          |1810            |1849            |Poland

---------------------------------------------------------------------------
Number of rows selected is 2
```

Figure 6-7. INSERT Command Display

The COMMIT WORK statement is necessary because SQL processes statements in units known as a **transaction**. When you issue the first SQL statement in a sequence, a transaction begins, and that transaction continues until you do a COMMIT WORK or ROLLBACK WORK. The use of transactions guarantees the consistency of data within the DBEnvironment.

Using the LOAD Command

In addition to the INSERT statement, you can use the LOAD command to add data with ISQL. The LOAD command has two options: EXTERNAL and INTERNAL.

The EXTERNAL Option

The EXTERNAL option of the LOAD command includes layout information for the data items. The following example shows the EXTERNAL option. The first number after the column name is the starting column position; the second number indicates the length of the field.

```
isql=>LOAD FROM EXTERNAL (Return)
>albums.sampledb.sys TO music.albums (Return)
>AlbumCode 1 4 (Return)
>AlbumTitle 13 40 (Return)
>Medium 53 2 (Return)
>AlbumCost 55 6 (Return)
>RecordingCo 61 10 (Return)
>DateRecorded 71 10 (Return)
>MfgCode 89 40 (Return)
>Comment 137 80 (Return)
>end; (Return)

Load depending on value in input record (Y/N)> n (Return)
Command in progress.
Number of rows read is 9
Number of rows processed is 9
COMMIT WORK to save to DBEnvironment.

isql=>COMMIT WORK; (Return)
isql=>SELECT * FROM music.albums; (Return)
```

```
select * from music.albums;
-----------+----------------------------------------+------+------------+--
ALBUMCODE  |ALBUMTITLE                              |MEDIUM|ALBUMCOST   |RE
-----------+----------------------------------------+------+------------+--
       2002|Famous Bel Canto Arias                  |ca    |       22.00|dg
       2003|Concertos for Diverse Instruments       |cd    |       19.00|rc
       2004|Symphonies and Chamber Works III        |cd    |       29.00|rc
       2005|Nielsen Symphonies 4 & 5                |ca    |       13.00|llo
       2006|Lenontyne Price: a Christmas Offering   |ca    |       13.00|lo
       2007|Sergei Rachmaninov Symphony No. 2       |cd    |       19.00|me
       2008|Franz Schubert: Lieder                  |cd    |       14.00|at
       2009|Beethoven Quartets No. 2 and No. 4      |cd    |       18.00|de
       2010|Chopin Recital: Ivo Pogorelich          |cd    |       16.00|dg

---------------------------------------------------------------------------
Number of rows selected is 9
```

Figure 6-8. Result of the EXTERNAL Option

The INTERNAL Option

The INTERNAL option of the LOAD command works with a special type of file that is generated by the UNLOAD INTERNAL command. Additional information is placed in the file describing the file layout. This command is simple to use because of this additional information. For example:

```
isql=>LOAD FROM INTERNAL Return
>hits.sampledb.sys to music.selections; Return

Command in progress.
Number of rows read is 18
Number of rows processed is 18
COMMIT WORK to save to DBEnvironment.

isql=>COMMIT WORK; Return

isql=>SELECT * FROM music.selections; Return
```

```
select * from music.selections;
-----------+----------------------------------------+----------------+-----
ALBUMCODE  |SELECTIONNAME                           |COMPOSERNAME    |TIMIN
-----------+----------------------------------------+----------------+-----
       2008|Der Saenger                             |Schubert        |
       2008|Fruehlingslied                          |Schubert        |
       2008|Fruehlingslaube                         |Schubert        |
       2008|Vor Meiner Wiege                        |Schubert        |
       2008|Drang in die Ferne                      |Schubert        |
       2008|Der Musensohn                           |Schubert        |
       2008|Viola                                   |Schubert        |
       2008|Vergissmeinnicht                        |Schubert        |
       2010|Klaviersonate Nr. 2 - 1                 |Chopin          |
       2010|Klaviersonate Nr. 2 - 2                 |Chopin          |
       2010|Klaviersonate Nr. 2 - 3                 |Chopin          |
       2010|Klaviersonate Nr. 2 - 4                 |Chopin          |
       2010|Prelude cis-moll op 45                  |Chopin          |
       2010|Scherzo cis-moll op 39                  |Chopin          |
       2010|Nocture Es-dur op 55 No 2               |Chopin          |
       2010|Etude F-dur op 10. No 8                 |Chopin          |
       2010|Etude As-dur op 10. No 10               |Chopin          |
       2010|Etude gis-moll op 25. No 6              |Chopin          |
---------------------------------------------------------------------------
Number of rows selected is 18
```

Figure 6-9. Result of the INTERNAL Option

You have now successfully inserted data into COMPOSERS, ALBUMS, and SELECTIONS tables.

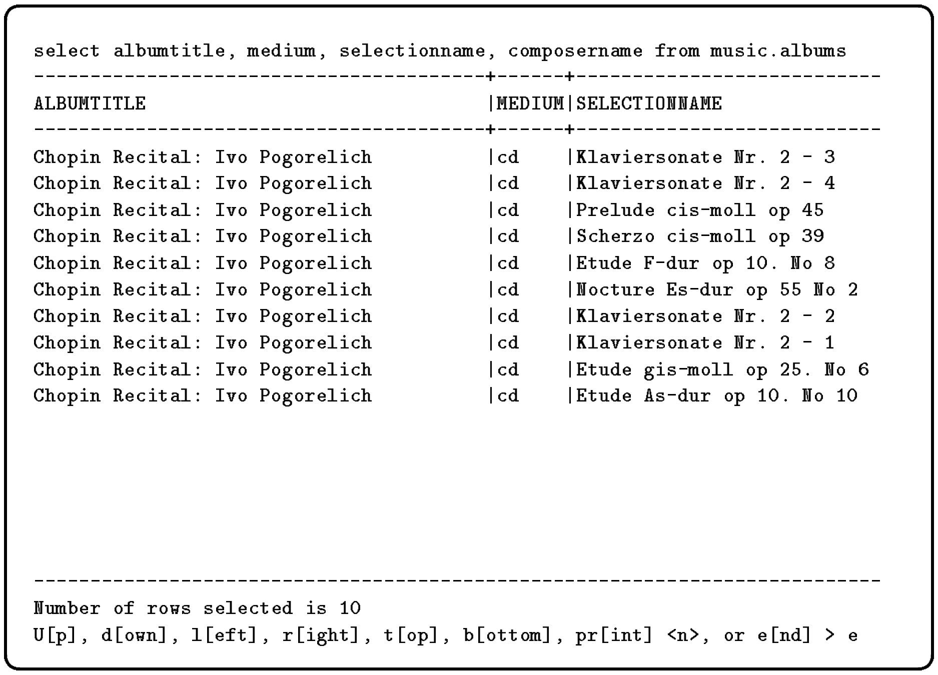
The result of the above SELECT shows selections for Albumcodes 2008 and 2010 corresponding with music by Schubert and Chopin, as indicated by the ComposerName values. It was possible to add these selections only because the Albums and Composers tables already contain these values. This is referential integrity. Referential integrity exists between these three tables based upon these column values.

Selecting Data

You can now select and combine various columns of information by joining tables based on common column values. For example, suppose you wanted to see all album titles, selections, composer, and the medium. This data is found in two tables, but can be joined by albumcode, as follows:

```
isql=>SELECT albumtitle, medium, selectionname, composername Return
>FROM music.albums, music.selection Return
>WHERE music.albums.albumcode = music.selections.albumcode Return
>AND music.albums.albumcost > 15.00; Return
```

This produces the following list:



```
select albumtitle, medium, selectionname, composername from music.albums
--------------------------------------+------+---------------------------
ALBUMTITLE                            |MEDIUM|SELECTIONNAME
--------------------------------------+------+---------------------------
Chopin Recital: Ivo Pogorelich        |cd    |Klaviersonate Nr. 2 - 3
Chopin Recital: Ivo Pogorelich        |cd    |Klaviersonate Nr. 2 - 4
Chopin Recital: Ivo Pogorelich        |cd    |Prelude cis-moll op 45
Chopin Recital: Ivo Pogorelich        |cd    |Scherzo cis-moll op 39
Chopin Recital: Ivo Pogorelich        |cd    |Etude F-dur op 10. No 8
Chopin Recital: Ivo Pogorelich        |cd    |Nocture Es-dur op 55 No 2
Chopin Recital: Ivo Pogorelich        |cd    |Klaviersonate Nr. 2 - 2
Chopin Recital: Ivo Pogorelich        |cd    |Klaviersonate Nr. 2 - 1
Chopin Recital: Ivo Pogorelich        |cd    |Etude gis-moll op 25. No 6
Chopin Recital: Ivo Pogorelich        |cd    |Etude As-dur op 10. No 10







-------------------------------------------------------------------------
Number of rows selected is 10
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] > e
```

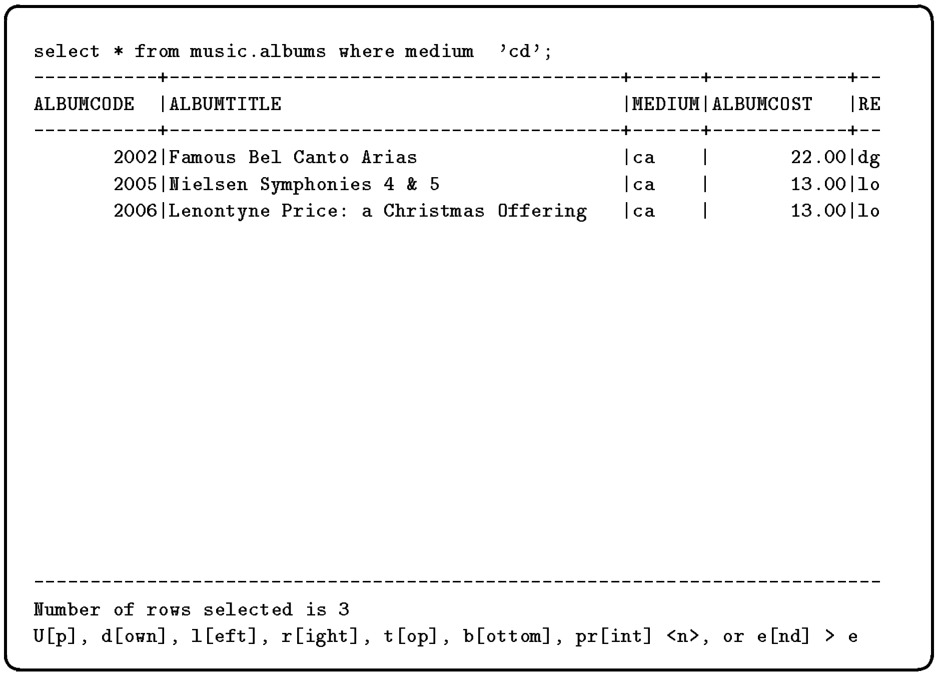
Figure 6-10. JOINED List

AlbumTitle and Medium come from the Albums table, SelectionName and ComposerName are from the Selections table. The WHERE clause defines the join criteria: albumcode is matched from each table. We have further qualified the query by asking only for albums costing more than $15.00.

Modifying Data

The UPDATE command changes column values. A quick look at some of our albums with the following query shows three albums not on 'cd':

```
isql=>SELECT * FROM music.albums WHERE medium <> 'cd';(Return)
```

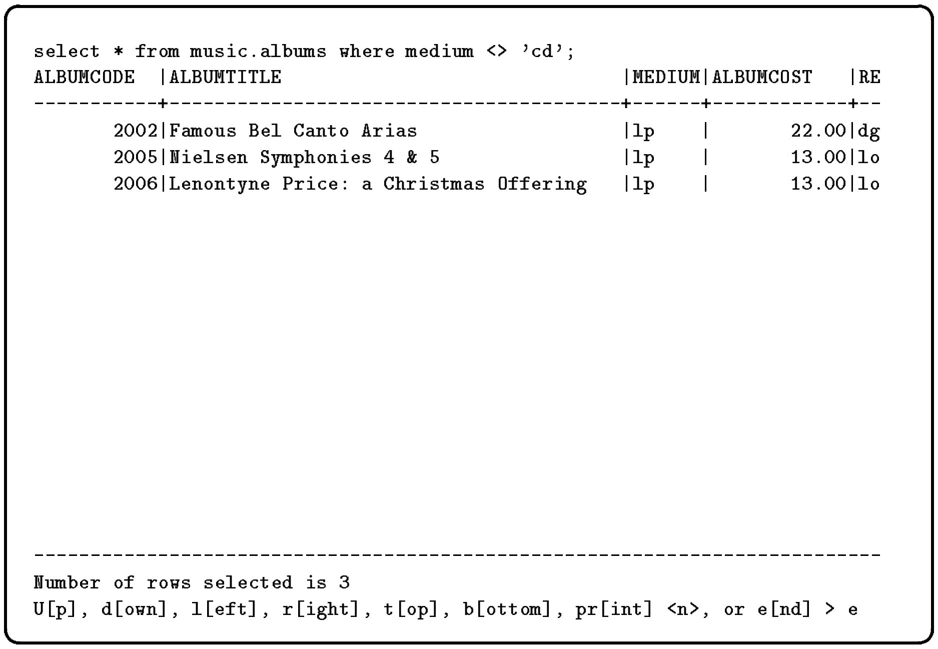


```
select * from music.albums where medium  'cd';
-----------+---------------------------------------+------+------------+--
ALBUMCODE  |ALBUMTITLE                             |MEDIUM|ALBUMCOST   |RE
-----------+---------------------------------------+------+------------+--
       2002|Famous Bel Canto Arias                 |ca    |       22.00|dg
       2005|Nielsen Symphonies 4 & 5               |ca    |       13.00|lo
       2006|Lenontyne Price: a Christmas Offering  |ca    |       13.00|lo









---------------------------------------------------------------------------
Number of rows selected is 3
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] > e
```

Figure 6-11. Result of the Query

To change the column values of these albums to 'lp' for 'Long Play' records, use the following:

```
isql=>UPDATE music.albums SET medium = 'lp' WHERE medium = 'ca'; Return

Number of rows processed is 3
isql=>COMMIT WORK; Return
isql=>SELECT * FROM music.albums WHERE medium <>'cd'; Return
```



```
select * from music.albums where medium <> 'cd';
ALBUMCODE  |ALBUMTITLE                              |MEDIUM|ALBUMCOST   |RE
-----------+----------------------------------------+------+------------+--
       2002|Famous Bel Canto Arias                  |lp    |       22.00|dg
       2005|Nielsen Symphonies 4 & 5                |lp    |       13.00|lo
       2006|Lenontyne Price: a Christmas Offering   |lp    |       13.00|lo










--------------------------------------------------------------------------
Number of rows selected is 3
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] > e
```

Figure 6-12. Result of Modifying Data

Deleting Data

Like the UPDATE command, the DELETE command is also very simple to use. The following command deletes entries with the albumcode of 2002:

```
isql=>DELETE FROM music.albums WHERE albumcode = 2002; (Return)
```

Conclusion

In this chapter, you learned to use these simple SQL statements in an IMAGE/SQL environment:

- INSERT
- LOAD
- SELECT
- UPDATE
- DELETE

The INSERT, SELECT, UPDATE, and DELETE statements have many other options and offer a great deal of flexibility in the IMAGE/SQL environment. Refer to the *ALLBASE/SQL Reference Manual* for more information on these statements.

Making TurboIMAGE/XL data work in a relational way is easy to set up and simple to work with. This new SQL access opens TurboIMAGE/XL to many, never before available, 4GL SQL tools. These third-party tools, including many designed specifically for the client/server environment, can now be used with IMAGE/SQL. In some cases, no knowledge of SQL is required, making these tools even more easy to use.

A

Tools For IMAGE/SQL

This appendix categorizes the available tools and lists some of the tools IMAGE/SQL users can use.

Note This appendix is for your convenience only; Hewlett-Packard does not endorse any particular product or company.

The tools available to IMAGE/SQL users function primarily in a client/server environment using Microsoft Windows. Most of the tools use HP PC API for ALLBASE/SQL and IMAGE/SQL. The tools fall into three categories, as summarized in Table A-1.

Table A-1. Categories of Tools

Category	Description
Decision Support	These tools support business decisions through information analysis, reporting, and graphical representation. These tools also offer data formatting for importing to spreadsheet products. End-user knowledge of SQL is not a prerequisite for using these tools.
Executive Information System (EIS)	These tools offer exception management capability, where information is monitored within user-defined limits and exceptions are highlighted with color and graphics. Other features include trend analysis and information drill-down (digging out the detailed elements of summarized information.) Some knowledge of SQL is necessary.
Application Development	These tools reside on either the PC or the server. They provide the ability to create a windows-like interface that allows programmers fast development of critical end-user applications. Knowledge of SQL is necessary.

The next sections lists some of the available tools.

Decision Support Tools

- *Impromptu* by Cognos
- *NewWave Access* by Hewlett-Packard
- *Q + E* by Pioneer Software
- *Quest* by Gupta

Executive Information System (EIS) Tools

- *Forest & Trees* by Trinzic

Application Development Tools on the PC

- *PowerBuilder* by Powersoft
- *SQLWindows* by Gupta

Application Development Tools on the Server

- *ALLBASE Toolset* by Hewlett-Packard
- *Focus* by Information Builders
- *JAM* by JYACC
- *Powerhouse* by Cognos
- *Speedware* by Speedware
- *Transact* by Hewlett-Packard
- *Uniface* by Uniface

Glossary

ATC
The acronym, ATC, stands for ALLBASE/Turbo CONNECT. The terms Turbo CONNECT and ALLBASE/Turbo CONNECT are synonymous with IMAGE/SQL. The acronym, ATC, appears in error messages for IMAGE/SQL (for example, ATCERR or ATCWARN).

ATCINFO
A permanent privileged file containing mapping information about data types and user security. By default, it is named *DBEnvironment-name*CR. If you want to set a file equation for this file, you must do so before attaching any TurboIMAGE/XL databases.

ATCLOG
A temporary unnumbered ASCII file. If IMAGE/SQL utility logging is on (the default), all IMAGE/SQL utility commands are written to this file. If it does not already exist, it is created. If it exists, log records are appended to it. By default, it is name ATCLOG. However, you can set a file equation to change the name of this file.

ATCUtil
Another name for the IMAGE/SQL utility program.

Attributes
A characteristic of a data element considered during database design. As you organize your data, you arrange it into categories that possess similar attributes. The categories are known as entities.

Authority
Permission to access specific objects for specific purposes within a DBEnvironment.

Authorization Group
See Group.

Base Table
Table upon which a view is based.

Check Constraint
An integrity constraint that enforces a condition that must not be false for the columns of a table. Any value you attempt to insert into a column that has a check constraint defined on it must either satisfy the condition or be NULL.

Class
Special category of IMAGE/SQL owner that is neither a particular DBEUserID nor a group. You do not explicitly create a class; you create it implicitly by creating objects owned by it. A class does not have members like a group. Objects owned by classes can be dropped or modified only by a DBA. A class does not have a password associated with it.

Column
Vertical division within a database table. Analogous to an item in a TurboIMAGE/XL dataset.

Column Authorization
Permission to update a specific column within a table.

Column List
One or more columns specified as part of a query result.

Concurrency
The ability of multiple users to access the same database files simultaneously. Concurrency is regulated by locking, which controls the degree of concurrent access permitted—from exclusive read or write access to shared read with concurrent updates.

Constraint
A condition placed upon a column or table that requires values in the column or table to meet certain conditions before a row can be inserted or deleted. Two types supported by IMAGE/SQL are unique constraints and referential constraints.

Data Analysis
Study of raw data before building a database. Concerns the kind of data that is to be stored and how the data is to be used.

Database
A structured arrangement of data elements designed for the easy selection of information. In IMAGE/SQL, a database is a collection of tables and views having the same ownership in a DBEnvironment. A DBEnvironment may contain several databases.

Database Administrator (DBA)
The individual with DBA authority who creates and maintains objects in a DBEnvironment. DBA authority permits the use of certain restricted SQL and SQLUtil commands or options and also confers co-ownership of all the objects in a DBEnvironment.

Database Design
The creation of a specific arrangement of data in tables or data sets with an appropriate security structure.

Data Control Language
The set of SQL statements that control access to data. This includes the ADD, REMOVE, GRANT, and REVOKE statements, as well as the statements to create, manage, and drop authorization groups. Also known as DCL.

Data Manipulation
The process of accessing data within a database.

Data Manipulation Language
The set of SQL statements that access data. This includes the actions of selecting data, inserting rows, updating columns, and deleting rows. Also known as DML.

Data Type
A kind of data that can be stored in database tables. Valid types are CHARACTER, VARCHAR, INTEGER, DECIMAL, FLOAT, DATE, TIME, DATETIME, INTERVAL, BINARY, and VARBINARY. LONG varieties of BINARY and VARBINARY are also available. See Chapter 3 for a maping of these SQL data types to TurboIMAGE data types.

DBA Authority
The most powerful authority within an IMAGE/SQL DBEnvironment. Includes the authority to grant or revoke all authorities for other users. DBA authority implies co-ownership of all objects within the DBEnvironment. The creator of the DBEnvironment is automatically a DBA.

DBC (Database Creator)
The creator of the TurboIMAGE/XL database. You must be either the database creator or give the TurboIMAGE/XL database maintenance word to attach a database to a DBEnvironment. Commands that add users, or display or modify user information can only be executed by the DBC.

DBECon File
DBEnvironment Configuration File. This contains startup parameters for the DBEnvironment. The contents of this file are initially determined at the time you issue the START DBE NEW statement. You can modify some of these parameters using SQLUtil, and you can override some of them with the START DBE statement.

DBECreator
The individual who issues the START DBE NEW statement. See DBC.

DBEFileSets

Logical grouping of DBEFiles.

DBEnvironment

A collection of files containing one or more databases. Files include the DBECon file, which holds startup parameters and log file names; DBEFile0, which contains the system catalog; and log files. A DBenvironment may also contain additional DBEFiles for table and index data. The DBEnvironment is the maximum scope of a transaction within IMAGE/SQL.

DBEFiles

Operating system files that hold DBEnvironment data. DBEFiles have both physical names (operating system names) and logical names by which the files are known internally to the IMAGE/SQL system catalog.

DBEUserID

A login name and account name joined with the character '@'. One type of owner of database objects.

Embedded SQL Programming

An application program incorporating SQL statements for programmatic access to IMAGE/SQL databases.

Entities

Basic subdivision of data elements in database design. Each entity is a thing or event about which information is kept in the database. For each entity, there is at least one attribute that uniquely identifies a data element as belonging to the entity.

Explicit Locking

Locking of tables in transactions by the use of the LOCK TABLE statement.

Expression

Specifies a value. The most common sources of values are columns in a table or host variables in an application program. Expressions are used to identify columns or rows or to define new values for columns.

Group

Authorization group. Membership in a group is used to confer common ownership or common authorization for other objects in the DBE. You create a group explicitly using the CREATE GROUP statement, then you add users to it. You can then grant authorizations to the group or revoke authorizations from the group. You can also use the group name for the ownership of database objects.

Logging

The use of log files to record operations that modify database files.

IMAGE/SQL Database

A TurboIMAGE/XL database attached to an SQL DBEnvironment.

Implicit Locking

Locking of tables in transactions according to table type and isolation level. For example, PRIVATE tables are locked exclusively for all access; PUBLIC tables are locked exclusively only for write operations.

Integrity Constraint

A constraint placed on the columns of a table to ensure that a database contains only valid data. Two types are the referential constraint and the unique constraint .

Isolation Level

The degree of separation enforced between the transactions of different users. There are four levels: Repeatable Read (RR), Cursor Stability (CS), Read Committed (RC), and Read Uncommitted (RU). You specify an isolation level in the BEGIN WORK statement.

ISQL

The interactive interface to Hewlett-Packard's relational database products. ISQL is the tool you use for queries as well as for loading and unloading data and other database administration tasks.

Joining

A query that accesses data from two or more relational tables at a time. A join column is a column that occurs in both tables of a join (often it is a key column) and contains similar values in both tables.

Key

One or more columns on which an index, hash structure, or integrity constraint are based.

Locking

A technique for concurrency control through which Hewlett-Packard's relational databases restrict access to data by one individual when the data is being used by another. Locks are of three types: shared, exclusive, or shared with intent to become exclusive. Lock type is determined by the type of table being accessed and by the kind of operation the user is performing. Locks are released when a transaction ends with a COMMIT WORK statement.

See also Explicit Locking and Implicit Locking.

Logging

The use of log files to record operations that modify database files. Logging is of two kinds: nonarchive logging , and archive logging . Both kinds permit you to roll back incomplete transactions following a system failure. This maintains data integrity by backing out changes to the database that were not committed. Only archive logging allows you to roll forward from an earlier version of a DBEnvironment by reapplying all committed transactions up to a specific recovery time.

Object

A structure created and stored in a DBEnvironment. The most common objects are tables, views, and groups.

Owner

A DBEUserID, a group name, or a class name. Ownership applies to database objects such as tables, views, indexes, and authorization groups. The owner may drop the object or transfer it to some other owner.

Predicate

Part of query syntax that specifies a subset of rows to be returned in the query result. Predicates are introduced by the keyword WHERE, so they are sometimes called WHERE clauses.

Predicates let you specify a range of values. The comparison predicate lets you compare a column value with a constant or host variable ; the LIKE predicate lets you compare a column value with a portion of a character string; the BETWEEN predicate specifies a range of values for a comparison. Special predicates of various kinds let you search for rows in more complex ways, including the use of subqueries.

Preprocessor

Converts programs with embedded SQL statements into modified source files for input to a compiler in one of several programming languages: C, COBOL, FORTRAN, and Pascal.

Primary Key

A column in a table defined so as to permit reference by foreign keys in other tables. A primary key also enforces uniqueness within the column.

Procedure

A sequence of SQL statements that are stored in a DBEnvironment and applied as a group either through rules or through execution by specific users. Together with rules, procedures let you define generalized constraints within a database to implement the relationships in the database design.

Projection
Relational operation that extracts a subset of columns from one or more tables.

Query
Request for information from database tables. A typical example is a SELECT statement.

Query Language
A set of operators, expressions, and commands that let you manipulate a database. The query language of IMAGE/SQL and ALLBASE/SQL is SQL.

Query Result
The rows retrieved by a SELECT statement. Query results are also known as result tables.

Read Committed (RC)
An isolation level that guarantees only that data you read in a transaction has been committed by some earlier transaction; that is, it is not currently in the process of update by some other transaction at the time you are reading it. In practical terms, this means that another transaction can update or delete the same row before your transaction is over. However, concurrency is greatly improved.

Read Uncommitted (RU)
An isolation level that enforces no separation between your transaction and those of others because no locks are obtained for reads. This level permits dirty reads; that is, reading data from the data buffers that has not and may never be written to the database at all.

Referential Integrity
An integrity constraint that enforces a relationship between the rows of two tables (a referenced table and a referencing table). Any value you attempt to insert into a table having a referential constraint (a referencing table) must either be NULL or be the same as a value in the referenced table.

Relation
See Table.

Relational Operations
Ways of extracting data from relational tables. The three primary relational operations are selection, projection, and joining.

Relationship
The meaningful interaction of entities in database design. Relationships may be one-to-one, one-to-many, or many-to-many.

Repeatable Read (RR)

An isolation level that enforces the highest level of separation between the transactions of different users. This level guarantees that when you re-read any data you have read previously in the same transaction, the value seen in the second read will be the same as the value seen in the first read. In practical terms, this means that other users may not update any data you have read at this isolation level until you COMMIT WORK.

Result Table

See Query Result.

Row

Horizontal division within a database table. Analogous to a record in a file.

Rule

A database object that ties the execution of a procedure to specific kinds of data manipulation performed on a database table. Together with procedures, rules let you define generalized constraints within a database to implement the relationships in the database design.

Schema

A complete SQL database definition.

Also, an ISQL command file containing commands to create a DBEnvironment and the objects within it.

Also, a TurboIMAGE/XL database definition that is the input to the TurboIMAGE/XL DBSCHEMA program.

Selection

Relational operation that extracts a subset of rows from one or more tables.

Serial Scan

A method of reading sequentially from the start of a table until the row is found. Also called table or relation scan. This is the default scan method used to access rows in a table when indexes do not exist.

SQL

See Structured Query Language.

SQLGEN

A utility program for database administrators that generates the SQL commands necessary to re-create all or part of a DBEnvironment. The output from SQLGEN is a command file (sometimes called a schema) that can be used as input to ISQL in re-creating database objects.

SQLUtil

A utility program for database administrators that assists with DBEnvironment maintenance, backup, and recovery. SQLUtil also lets you modify the startup parameters for a DBEnvironment.

Structured Query Language

A standard query language syntax defined by ANSI standards in the United States and X/OPEN standards in Europe. The relational database query language used by IMAGE/SQL and ALLBASE/SQL.

Subquery

A query within another query. An example is a subquery embedded in the predicate of another query. The result of the inner query is used to evaluate the outer query.

SYSTEM

A DBEFileSet created when you issue the START DBE NEW statement. The DBEFile known as DBEFile0 is associated with SYSTEM, which is the DBEFileSet containing the system catalog. You can add DBEFiles to SYSTEM as you would to any other DBEFileSet.

Also, a special user associated with the system views in the system catalog.

System Catalog

A system-maintained database of tables and views owned by the special user SYSTEM and containing information about all the objects in the DBEnvironment. Differs from the DBECon file, which contains startup parameters, not object definitions.

System Table

See System View.

System View

A component view within the system catalog. You can issue queries on the views in the system catalog just as you would on ordinary database tables to display information about the DBEnvironment.

Table

Basic unit of data storage in a relational database. Also known as a relation. Tables consist of rows and columns. A result table is a query result displayed in tabular form.

Table Authority

Permission to use specific SQL statements on particular tables. There are several kinds of TABLE authority: SELECT, INSERT, DELETE, and UPDATE. SELECT, INSERT, and DELETE let you operate on rows or sets of rows in a table; UPDATE lets you modify specific rows or columns in a table.

Transaction

A unit of work. Also, a unit of DBEnvironment logging and recovery. A transaction is started with a BEGIN WORK statement and is ended by a COMMIT WORK statement. The BEGIN WORK statement may be implicitly issued by IMAGE/SQL if no other transaction is current when an SQL statement is executed.

Unique Constraint

An integrity constraint that requires that no two rows in a table have the same values in a specified column or columns.

View

A table derived by placing a "window" over one or more tables. The derivation of a view is a SELECT statement. View names are governed by the same rules as table names.

Index