InterBase 6

Getting Started

Installation and Migration

Borland/INPRISE

100 Enterprise Way, Scotts Valley, CA 95066   http://www.interbase.com

6E0R0699

Table of Contents

CHAPTER

1 Installation

This chapter gives detailed instructions for installing InterBase products on Windows® 95/98, Windows NT®, and Solaris™.

InterBase on other platforms might install differently due to operating system requirements. Refer to installation notes included in the software package, on the media, or on the InterBase web site (http://www.interbase.com/) for platform-specific installation instructions that supersede the content in this chapter.

Installation on Windows NT and Windows 95/98

It is recommended that you uninstall any previous versions of InterBase that are on the server where you are installing the InterBase 6 product. With this software, as with any software, it is recommended to remove any field test versions prior to installing. If InterBase 5.5 or earlier is present on your server, you *must* uninstall it before proceeding with the Version 6 install.

Before uninstalling, you must first stop the InterBase Server. Click on the Services icon in the Control panel, select InterBase Server, and click on stop.

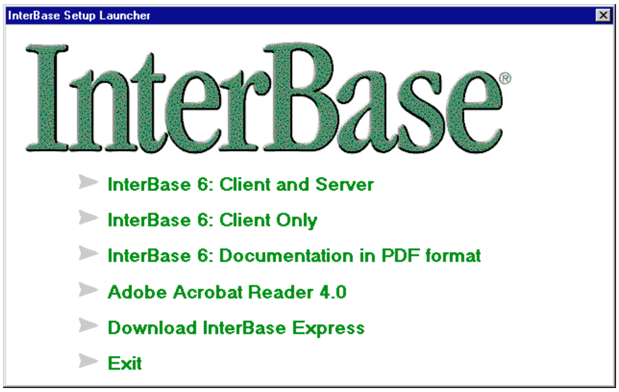
- There must be one and only one copy of **gds32.dll** on the machine where InterBase 6 client or server is running. If this file exists on the server, you must remove it before installing InterBase 6.

- You cannot install InterBase onto a network (mapped) drive.
- You must be logged in as a user with Administrator privileges to install or uninstall InterBase on Windows NT.

Installing InterBase on Windows

Windows system requirements: Windows NT 4.0 with Service Pack 4 or Windows 95/98 with Service Pack 1. Installation of the Client and Server requires approximately 44MB of disk space for a full install that includes InterBase, Adobe Acrobat Reader, InterBase Express and the full document set. Only 11MB is needed to install the InterBase product without the documents, InterBase Express™, or Acrobat Reader.

InterBase Setup Launcher


InterBase Setup Launcher
InterBase®
InterBase 6: Client and Server
InterBase 6: Client Only
InterBase 6: Documentation in PDF format
Adobe Acrobat Reader 4.0
Download InterBase Express
Exit

When you place the InterBase CD-ROM is the drive and close the drive, the InterBase Setup Launcher displays on your screen if you have Autoplay enabled. You can always install by executing **Setup.exe** at the root of the CD-ROM. The Launcher is a menu of different software packages you can install from the CD-ROM. You install only one element at a time. When the install of that element is complete, the menu screen reappears and you can choose the next element or choose to exit. The options are:

- InterBase 6: Client and Server

  The fully functional InterBase Server, which you can run on Windows NT or Windows 95/98. You can select optional components later in the wizard, including database client and server tools, ODBC driver, example database, and client application development files.

- InterBase 6: Client Only

  As above, but without the server component.

- InterBase 6: Documentation in PDF format

  The full InterBase 6 documentation set.

- InterBase 6: InterBase Express

  InterBase Express (IBX) components for Delphi, which access the InterBase API directly. IBX allows you to build InterBase-specific database applications without the overhead of the Borland Database Engine (BDE). These components access InterBase 6 features, including the Services API, Install API, and Licensing API. This IBX is a superset of the IBX that is distributed with Delphi, which accesses only InterBase 5 features. Delphi 5 must be installed before you install this IBX.

- Adobe Acrobat Reader 3.0.1

  Acrobat Reader With Search for using the online InterBase documentation and performing full-text searches on the complete doc set.

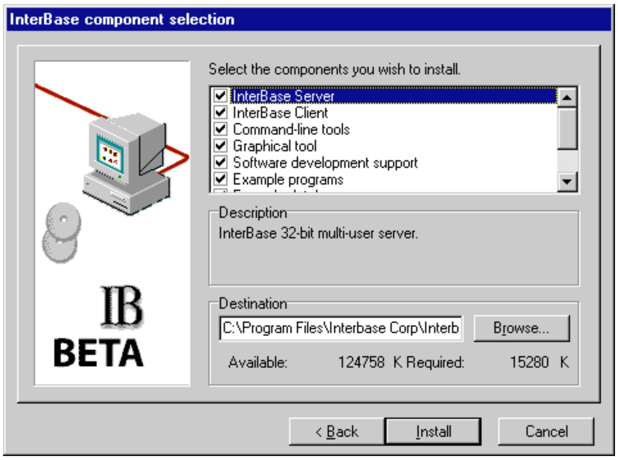
InterBase Client and Server Setup

1. To install the client and server, choose "Install InterBase 6: Client and Server" from the Launcher. This displays a welcome to the InterBase Server installation wizard. Read the screen and then choose Next.
2. Read the **install.txt** document that displays in the Important Installation Information window. Choose Next to proceed to the next step.
3. You must read and accept the terms of the Software License Agreement. Choose Yes to continue the installation or No to exit.
4. In the Software Activation Certificate window, enter a valid Certificate ID and its corresponding Certificate key in the appropriate fields to activate the InterBase server. These numbers are on the Software Activation Certificates that you purchased through your sales representative.

   If you are installing an evaluation copy, type `eval` in the Certificate ID field.

   Note Use the License Registration tool to enable additional functionality by adding other software activation keys.

   Choose Next to display the InterBase Component Selection window.


InterBase component selection

Select the components you wish to install.

- ☑ InterBase Server
- ☑ InterBase Client
- ☑ Command-line tools
- ☑ Graphical tool
- ☑ Software development support
- ☑ Example programs

Description

InterBase 32-bit multi-user server.

Destination

C:\Program Files\Interbase Corp\Interb

Available: 124758 K Required: 15280 K

5. Select components from the list in this window to customize your InterBase installation to your needs. To get more information about a choice, highlight it and read the Description panel.

   Specify a destination directory for installing the software. It must be on a drive local to the machine: it cannot be on a network drive.

   Choose Install. InterBase Setup installs both the InterBase client and your selected server components. It then displays a final window where you can choose to display the Readme file.

6. Choose Finish. The Setup Launcher redisplays so that you can install another element, such as the documentation or Acrobat Reader. The complete InterBase document set with full-text search is available online in Adobe Acrobat format. If you choose Online Documentation as an install component, they are present on your hard drive in the ***ib_install_dir*\doc** directory. If you do not install them, you can still read them from the **\doc** directory of the InterBase CD-ROM.

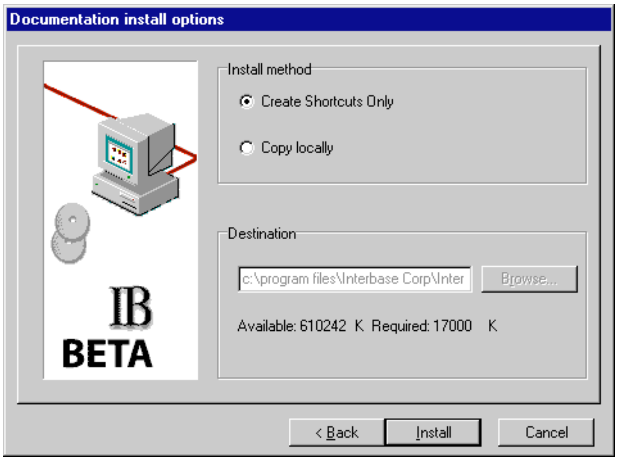
Installing the InterBase client only

To install only the InterBase client, choose "Install InterBase 6: Client Only" from the launcher. The install procedure is identical to that for the Client and Server with one exception: Setup does not ask you for a Certificate ID/key.

Installing documentation

The complete InterBase document set with full-text search is available online in Adobe Acrobat format. You can install the complete documentation set on your hard drive in the ***ib_install_dir*\doc** directory, or you can create shortcuts to it and browse the **\doc** directory of the InterBase CD-ROM.

1. To install the documentation, choose “Install InterBase 6: Documentation in PDF format” from the Launcher. This displays a welcome to the InterBase Documentation installation wizard. Read the screen and then choose Next.
2. Read the **install.txt** document that displays in the Important Installation Information window. Choose Next to proceed to the next step.
3. You must read and accept the terms of the Software License Agreement. Choose Yes to display the Documentation install options window or No to exit.
4. The Install method panel allows you to create shortcuts to the documentation on your hard drive, or to copy the entire documentation set to your local hard drive. Using shortcuts will save disk space, but you will need to have the CD installed whenever you need to access the documentation.

   
   Documentation install options

   Install method: [✓ Create Shortcuts Only] Copy locally

   Destination: c:\program files\Interbase Corp\Inter

   Available: 610242 K Required: 17000 K

   To copy the documentation to your local hard drive, specify a destination directory for installing the software. It must be on a drive local to the machine: it cannot be on a network drive.

   Choose Install. InterBase Setup installs the documentation.
5. Choose Finish. The Setup Launcher redisplays so that you can install another element, such as the InterBase Express or Acrobat Reader.

Configuring InterBase

You can control many aspects of how InterBase runs, as well as choosing to configure an ODBC driver or adding software activation certificates. See Chapter 4, “Server Configuration” in the *Operations Guide* for information on these topics.

Logging on

To attach to any database, you must have a user name and password. When you first install InterBase, there is one user defined. That user is SYSDBA and the password is *masterkey*. The SYSDBA user has special privileges that override normal SQL security and there are a number of database maintenance tasks that only SYSDBA can perform. You should change the SYSDBA password immediately after installing InterBase.

Use IBConsole or the **gsec** utility to change a password and create additional users. For more information see the IBConsole online help or the *Operations Guide*.

Installing InterBase Express

To install InterBase Express (IBX), simply click on InterBase Express in the Launcher. The IBX component package is installed in Delphi 5.

InterBase 6 field test participants also receive the Delphi 5 field test software, to enable IBX testing.

Uninstalling InterBase on Windows

To uninstall InterBase and InterBase Express, use the **Control Panel | Add/Remove Programs** applet. Choose InterBase or InterBase Documentation, and click the Add/Remove button.

InterBase documentation is removed when the InterBase Client or Server is uninstalled.

- The InterBase Server, Guardian, and InterServer processes must not be running when you uninstall the software. To stop one of these applications, right-click its icon in the Task Tray and selecting Shutdown. To stop one of these services (Windows NT), use the **Control Panel | Services** applet.
- You must be logged in as a user with Administrator privileges to install or uninstall InterBase on Windows NT.
- Uninstall never removes **isc4.gdb** or files created by the server process, including **interbase.log**, ***host*.evn**, ***host*.lck**.
- The Windows InterBase installation allows you to choose between performing a complete install or selecting individual components. If you choose to install components at different times, the uninstall program removes only the components selected for the last install.

- The ODBC driver or the ODBC driver manager must be removed manually; there is no uninstall available through the Windows control panel.

Installation on UNIX

This section provides instructions for installing InterBase 6 on Solaris and HP-UX. For platform-specific instruction on installing InterBase on other UNIX brands, see the InterBase web site (http://www.interbase.com/).

Preparing to install on UNIX

1. Save older databases

   InterBase 6 uses a new On-Disk Structure (ODS), ODS 10. Databases created with InterBase 5 used ODS 9.1. To take advantage of new InterBase 6 features, you must use **gbak** to back up any databases that you intend to use with the Version 6 software. V5 databases can be backed up using V5 **gbak** or with the InterBase 6 **gbak** if you are performing the backup after the install. Once a database has been converted to ODS 10, it cannot be converted back to earlier versions of the ODS. For more information, refer to the Migration chapter.

2. Save customization files

   If you have InterBase 5 installed on the server and you want to preserve the customization files, copy them to a safe place, for example:

   ```
   gbak –b /usr/interbase/isc4.gdb /var/tmp/isc4.gbak
   cp /usr/interbase/isc_license.dat /var/tmp
   cp /usr/interbase/isc_config /var/tmp
   ```

   You can skip this step if you haven't customized these files in a previous installation.

   Note When you use **pkgrm** remove InterBase V5 for Solaris, these files are automatically saved in **/usr/tmp**.

3. Save older versions

   If InterBase 5 is running on your server, shut it down. To save the current version, rename the directory, for example:

   ```
   ibmgr -shut -user sysdba -password password
   mv /usr/interbase /usr/interbase.save
   ```

4. Point to the install directory

If you install into a directory other than **/usr/interbase**, the install program creates a symbolic link from **/usr/interbase** to that directory.

Installing on Solaris

InterBase requires Solaris versions 2.6.*x*. or 7

1. Log in to your database server as **root**.
2. Load the CD-ROM with the InterBase 6 product. Mount the CD-ROM at an appropriate mount point. If you have the volume manager running, this is done for you, and the CD-ROM mounts according to its label. For instance **/cdrom/interbase_sos_V6** for the CD-ROM of InterBase 6 for Solaris.
3. Run **setup.ksh**. **setup.ksh** brings up the following menu:

```
   1.Install InterBase Client and Server software
   2.Install InterBase Client Only software
   3.Install Adobe Acrobat Reader software
   4.Exit
Enter selection.(default 1) [1-4]
```

The following sections describe these options

Installing InterBase Client and Server on Solaris

When you select “Install InterBase Client and Server software” from the menu, the setup script displays the following messages and starts the **pkgadd** utility:

```
Starting InterBase Client and Server Install, please wait...
set NONABI_SCRIPTS=TRUE
export NONABI_SCRIPTS
pkgadd -d CDROM_DIR/Interbase6.0_ClientServerpkg

The following packages are available:
   1  interbase InterBase RDBMS Software
                (sparc) InterBase Version 6.0

Select package(s) you wish to process (or ‘all’ to process
all packages). (default: all) [?,??,q]:
```

1. Press Enter to choose the Install InterBase default.

```
Processing package instance <interbase> from
   </cdrom/interbase_SOS_V6>
```

```
InterBase RDBMS Software
(sparc) InterBase Version 6.0

Enter the absolute pathname of the install directory
   (default /usr) [?,q]
```

2. Press Enter to accept the default or type in the pathname in which to create the **interbase** directory.

```
Using </usr> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with superuser
permission during the process of installing this package.

Do you want to continue with the installation of <interbase> [y,n,?]
```

3. Choose *y*. The installation script must have superuser privilege to modify system files and create symbolic links.

```
Installing InterBase RDBMS Software as <interbase>
## Installing part 1 of 1.
/usr/interbase/install.txt
/usr/interbase/Release_Notes.pdf
. . .
[ verifying class <none> ]
## Executing postinstall script.
. . .
Please enter the license certificate ID:
```

4. Enter the ID from your InterBase Server software activation certificate.

```
Please enter the license certificate key:
```

5. Enter the corresponding KEY from your InterBase Server license ID/key card.

   You can add more license keys at any time .

   At this step, the InterBase install script looks for the InterClient installation script on the CD-ROM, and runs it if possible. See **"Uninstalling InterBase on UNIX" on page 15**.

   Thereafter, the install script returns to **pkgadd**:

```
Installation of <interbase> was successful.
```

```
The following packages are available:
   1  interbase InterBase RDBMS Software
                (sparc) InterBase Version 6.0

Select package(s) you wish to process (or ‘all’ to process
all packages). (default: all) [?,??,q]:
```

6. Choose *q*. There is only one package to install using **pkgadd**.
7. If you saved customization files, restore them now with the ones you backed up before installing InterBase 6. For example:

   ```
   # cp /tmp/isc_config /usr/interbase
   ```

8. If you intend to run the **ibserver** daemon as a user other than root, create an UNIX user account named “interbase” on your machine. Log in as this user before starting the SuperServer with the **ibmgr** utility.
9. Execute the following command to start the InterBase server:

   ```
   # echo “/usr/interbase/bin/ibmgr –start –forever” | su interbase
   ```

   This starts the SuperServer daemon (**ibserver**) and a guardian (**ibguard**) program that keeps track of **ibserver**.
10. Now that the server is running, you can restore the security database with the ones you backed up before installing InterBase 6InterBase 6.

    ```
    # gbak –r /tmp/isc4.gbak jupiter:/usr/interbase/isc4.gdb
    ```

11. To test your existing databases with InterBase 6, use **gbak** to restore the backup files you created before upgrading.

Installing only the client on Solaris

To install only the client on Solaris, select “Install InterBase Client Only software” from the menu. The installation procedure is the same as installing the client and server, except that you are not prompted for a license certificate and key.

Running multiple versions on UNIX

You can run only one version of the InterBase server per host: you cannot run multiple versions of the InterBase server simultaneously. The InterBase 6 server process is called **ibserver**. There is no longer any **gds_inet_server** as there was in the Classic architecture.

If you want to install InterBase 6 but retain the ability to go back to a previous version, you should install InterBase 6 in a location other than **/usr/interbase**, and create a symbolic link from **/usr/interbase** that points to the directory in which you installed InterBase 6.

To switch between versions of InterBase, follow these steps:

1. Shut down the current server and point the symbolic link to a directory that contains the version you want to run. Use `ibmgr -shut` to stop the version 6 server process
2. Run the install script for the version you want to run. This makes the necessary change to the **/etc/inetd.conf** configuration file.

   ```
   cd /usr ; sh ./interbase/install
   ```

3. Use `kill –HUP` to refresh the **inetd** daemon with the new configuration.

   ```
   ps | grep inetd
   kill -HUP process-ID
   ```

4. Start the server in the directory to which the symbolic link points. To start the server daemon, follow the steps in "UNIX daemon" in the *Operations Guide*.

Uninstalling InterBase on UNIX

- On Solaris, use **pkgrm** to remove the InterBase 6 package:

```
# pkgrm interbase
```

CHAPTER

2

InterBase 6 Migration Guide

This chapter describes how to plan and execute a smooth migration from earlier versions of InterBase to InterBase 6. Topics in this chapter include:

- Migration issues: on-disk structure and dialects
- Migration paths
- Delphi and InterBase 6: BDE and IBX

Note See the *Release Notes* for a detailed introduction to new InterBase features.

Introduction

This document outlines the steps necessary to migrate servers, databases and clients and discusses some of the issues that need to be addressed in order to successfully migrate from InterBase 5 to InterBase 6. With careful planning the process of migrating from InterBase 5 to InterBase 6 will go smoothly.

Overview of Migration Process

Server and database migration

In broad overview, migrating servers includes the following steps:

- Backup all databases to be migrated
- Install the InterBase 6 server
- Restore databases to be migrated; at this point, you have dialect 1 databases
- Validate migrated databases
- Migrate databases to SQL dialect 3 (see pages 37 to 45)

Client migration

Migrating clients includes the following general steps:

- Identify the clients that must be upgraded to InterBase 6
- Identify areas in your application which may need upgrading
- Install the InterBase 6 client to each machine that requires it
- Upgrade SQL applications to SQL dialect 3

Migration Issues

This section covers several broad issues, including of client-server compatibility and dialects.

Clients and databases

Applications using an older version of the InterBase client work with the InterBase 6 server and its databases with some restrictions:

- Version 5 clients cannot access dialect 3 columns that are stored as INT64, TIME, or DATE. (DECIMAL and NUMERIC columns with precision greater than 9 are stored as INT64.)
- Version 5 clients cannot display new datatypes in metadata using the SHOW command, or any equivalent.
- Version 5 clients interpret the DATE datatype as TIMESTAMP, since that was the definition of DATE prior to InterBase 6.
- Version 5 clients cannot access any object named with a delimited identifiers.
- Clients that use the Borland Database Engine (BDE) to access an InterBase 6.0 server are not able to access any of the new field type regardless of the version of the InterBase client installed.

IMPORTANT Version 5 clients have one advantage over version 6 clients: If you migrate an older database that uses some of the new version 6 keywords as identifiers to version 6 dialect 1, these older version 5 clients can still access those keyword objects. Version 6 dialect 1 cannot do so. Dialect 3 clients can access these keyword objects if they are delimited in double quotes.

If version 5 clients use any of the new InterBase 6 keywords as object names, the InterBase 6 server permits this without error because it recognizes that these clients were created at a time when these were not keywords.

Example For example, the following statement uses the new keyword word TIME:

```
SELECT TIME FROM atable;
```

This statement, when executed via a pre-InterBase 6 client returns the information as it did in previous versions. If this same query is issued using a version 6 client, an error is returned since TIME is now a reserved word. See page 25 for a list of new keywords.

Servers, databases, and ODS 10

InterBase 6 databases are stored in the ODS 10 data format. This format is not backward-compatible with older formats.

- If you upgrade a server to InterBase 6, you *must* migrate the databases that it accesses to InterBase 6 as well. InterBase 6 servers cannot read or write to InterBase 5 or older databases.
- If you do not upgrade a server, do not migrate the databases that it accesses. InterBase 5 and older servers cannot access InterBase 6 databases.

The following discussions of compatibility between clients and databases all assume that the server version is correct for the database version.

Understanding SQL Dialects

InterBase recognizes different client and database dialects to allow users to move forward with new features that are incompatible with older versions of InterBase. The following features, called transition features, have different meanings based on the dialect used by the client applications:

- Double quotes
- DECIMAL and NUMERIC datatypes with precision greater than 9
- DATE, TIME, and TIMESTAMP datatypes

An InterBase dialect identifies how these areas are interpreted. Both clients and databases have a dialect in InterBase 6 to instruct the server how to interpret the transition features.

Clients and databases each have dialects. Servers do not themselves have a dialect, but they interpret data structures and client requests based on the dialect of each.

Note In this document, “large exact numerics” refers to DECIMAL and NUMERIC datatypes with a precision greater than 9.

Dialect 1

In dialect 1, the InterBase server interprets the transition feature set in the InterBase 5 manner:

- Double quoted strings are string constants. Delimited identifiers are not available.
- The DATE datatype contains all the timestamp information and is interpreted as TIMESTAMP; the name has changed but the meaning has not. Clients expect the entire timestamp to be returned. In dialect 1, DATE and TIMESTAMP are identical.
- The TIME datatype is not available.
- *In databases*: DECIMAL and NUMERIC datatypes with precision greater than 9 are stored and returned as DOUBLE PRECISION, not INT64. *In clients*: The dialect 1 client expects information stored in these datatypes to be returned as double precision. Dialect 1 clients cannot create fields to hold 64-bit integers in a dialect 3 database.

An InterBase 6 server recognizes all the other InterBase 6 features in clients and databases.

Version 6 servers can send warnings to version 6 dialect 1 clients. However, If the server detects that the client is pre-version-6, it does not send warnings, since the client can't understand them.

Dialect 2

Dialect 2 is available only on the client side. It is intended for assessing possible problems in legacy metadata that is being migrated to dialect 3. To determine where the problem spots are when you migrate a database from dialect 1 to dialect 3, you extract the metadata from the database, set **isql** to dialect 2, and then run that metadata file through **isql**. **isql** issues warning and errors whenever it encounters double quotes, DATE datatypes, or large exact numerics in order to alert to places where you might need to change the metadata in order to make a successful migration to dialect 3.

Dialect 3

In dialect 3, the InterBase server interprets the transition feature set in the InterBase 6 SQL 92-compliant manner:

- Double quoted strings are delimited identifiers.
- *In databases*: The DATE datatype contains only date information. *In clients*: The client expects only date information from the DATE datatype.
- The TIME datatype is available and stores only time information.

- *In databases*: DECIMAL and NUMERIC datatypes with precision greater than 9 are stored as INT64 if and only if they are in columns that were created in dialect 3. *In clients*: Dialect 3 clients expect DECIMAL and NUMERIC datatypes with precision greater than 9 to be returned as INT64.

  For information about how to migrate older data to INT64 storage, see "Migrating databases to dialect 3" on page 37 and "Migrating NUMERIC and DECIMAL datatypes" on page 42.

  All of the other new InterBase features are available, as they are in both dialects.

Setting the dialect of an **isql** *client*

You can change the dialect of the **isql** client in either of two ways.

- You can start it using the -**sql_dialect** ***n*** switch, where *n* is 1, 2, or 3.
- At any point, you can issue the SET SQL DIALECT *n* statement in **isql** or in SQL scripts.

The following table shows this precedence:

Lowest	Dialect of an attached database
Middle	Command line specification: `isql -sql_dialect n`
Highest	Dialect explicitly specified during the session, for example `SET SQL DIALECT n;`

In InterBase 6, **isql** has the following behavior with regard to dialects:

- If you start **isql** and attach to a database without specifying a dialect, **isql** takes on the dialect of the database.
- If you specify a dialect at the command line when starting **isql**, it retains that dialect after connection unless explicitly changed.
- Whenever you change the dialect during a session using SET SQL DIALECT *n*, **isql** continues to operate in that dialect until explicitly changed.
- When you create a database using **isql**, the database is created with the dialect of the **isql** client; for example, if **isql** has been set to dialect 1, then when you create a database, it is a dialect 1 database.
- If you create a database without first specifying a dialect for the **isql** client or attaching to a database, **isql** creates the database in dialect 3.

  The statements above are true whether you are running **isql** as a command-line utility or are accessing it through IBConsole, InterBase's new interface.

IMPORTANT Any InterBase 6 **isql** client that attaches to a version 5 database resets to dialect 1.

gpre dialect behavior

In InterBase 6, **gpre** takes on the dialect of the database to which it is connected. This enables it to parse pre-6 source files without any change. You can set **gpre** to operate as a client in another dialect by starting it with the **-sql_dialect *n*** switch:

```
gpre -sql_dialect 3
```

The precedence of dialect specification is as follows:

Lowest	Dialect of an attached database
Middle	Command line specification: `gpre -sql_dialect n`
Highest	Dialect explicitly specified within the source, for example `EXEC SQL SET SQL DIALECT n`

New features

Many of the new InterBase 6 features operate without reference to dialect. Other features are dialect-specific. These features are listed below and described in more detail in the *Release Notes*. Many of them are also detailed in the full document set, although that set is not complete for beta.

FEATURES THAT ARE AVAILABLE IN ALL DIALECTS

The following new features are available in both dialects of InterBase 6:

- IBConsole, InterBase's new graphical interface

  IBConsole, InterBase's new graphical user interface, combines the functionality of Server Manager and InterBase Windows ISQL. You now create and maintain databases, configure and maintain servers, and execute interactive SQL from one integrated interface.

- Read-only databases

  You can make InterBase 6 databases be read-only. This permits distribution on read-only media such as CDROMs and reduces the chance of accidental or malicious changes to databases.

- Altering column definitions

  The ALTER COLUMN clause of the ALTER TABLE statement can change a column's name, datatype, or position.

- Altering domain definitions

  ALTER DOMAIN now includes the ability to change the name or datatype of a domain definition.

- The EXTRACT() function

  The new EXTRACT() function extracts information from the new DATE, TIMESTAMP, and TIME datatypes. In dialect 1, you can use it to extract information from the TIMESTAMP datatype. **Note** "DATE" is new in the sense that it has a different meaning in dialect 3 databases than it did previously.

- SQL warnings

  The InterBase API function set now returns warnings and informational messages along with error messages in the status vector.

- The Services API, Install API, and Licensing API

  InterBase now provides three new function libraries. The Services API, which is part of the InterBase client library, provides functions for database maintenance tasks, software activation, requesting information about the configuration of databases and server, and working with user entries in the security database.

- New **gbak** functionality

  In InterBase 6, **gbak** can

  · Back up to multiple files and restore to multiple files

  · Use the -**service** switch to perform server-side backups and restores

  · Set databases to read-only mode when restoring

- InterBase Express™ (IBX)

  IBX provides native Delphi components for InterBase data access, services, and installation.

FEATURES AVAILABLE ONLY IN DIALECT 3 DATABASES

The following features are available only in dialect 3 clients and databases because they conflict with dialect 1 usage.

- Delimited identifiers

Identifiers can now be keywords, contain spaces, be case sensitive, and contain non-ASCII characters. Such identifiers must be delimited by double quotes. String constants must be delimited by single quotes.

- INT64 data storage

  In dialect 3 databases, data stored in DECIMAL and NUMERIC columns is stored as INT64 when the precision is greater than 9. This is true only for columns that are *created* in dialect 3. These same datatypes are stored as DOUBLE PRECISION in dialect 1 and in all older InterBase versions. This change in storage also requires different arithmetic algorithms.

- DATE and TIME datatypes

  In dialect 3, the DATE datatype holds only date information. This is a change from earlier InterBase versions in which it stored the whole timestamp.

  Dialect 3 allows the use of the TIME datatype, which hold only the time portion of the timestamp.

New keywords

InterBase 6 introduces the following new keywords:

COLUMN	SECOND
CURRENT_DATE	TIME
CURRENT_TIME	TIMESTAMP
CURRENT_TIMESTAMP	TYPE
DAY	WEEKDAY
EXTRACT	YEAR
HOUR	YEARDAY
MINUTE	
MONTH	

These keywords are reserved words in all version 6 dialects.

- You cannot create objects in a version 6 dialect 1 database that have any of these keywords as object names (identifiers).
- You can migrate a version 5 database that contains these keywords used as identifiers to version 6 dialect 1 without changing the object names: a column could be named “year”, for instance.
  - Version 5 clients can access these keyword identifiers without error.

  - Version 6 clients *cannot* access keywords that are used as identifiers. In a dialect 1 database, you must change the names so that they are not keywords.
  - If you migrate directly to dialect 3, you can retain the names, but you must delimit them with double quotes. To retain accessibility for older clients, put the names in all upper case. Delimited identifiers are case sensitive.
- Although TIME is a reserved word in version 6 dialect 1, you cannot use it as a datatype because such databases guarantee datatype compatibility with version 5 clients.
- In dialect 3 databases and clients, any reserved word can be used as an identifier as long as it is delimited with double quotes.

Delimited identifiers

To increase compliance with the SQL 92 standard, InterBase 6 introduces *delimited identifiers*. An identifier is the name of any database object; for instance a table, a column, or a trigger. A delimited identifier is an identifier that is enclosed in double quotes. Because the quotes delimit the boundaries of the name, the possibilities for object names are greatly expanded from previous versions of InterBase. Object names can now:

- Be a keyword
- Include non-ASCII characters
- Include spaces
- Be case sensitive

The double-quotes issue

Up to and including version 5, InterBase allowed the use of either single or double quotes around string constants. The concept of delimited identifiers did not exist. InterBase 6 supports delimited identifiers, which means that double quotes must be used *only* for delimited identifiers.

SINGLE QUOTES AND DOUBLE QUOTES: SUMMARY

- In all versions of InterBase, anything delimited by single quotes is treated as a string constant.
- In version 5 and older of InterBase, string constants could be delimited by either double or single quotes. Since there was no concept of delimited identifiers, double quotes were always interpreted as string constants.

- Version 6 dialect 1 is a transition mode that behaves like older versions of InterBase with respect to quote marks: it interprets strings within double quotes as string constants and does not permit delimited identifiers.
- Version 6 dialect 3 uses double quotes only for delimited identifiers. String constants must be delimited by single quotes, never double.
- See Table 2.1 for how to handle quotations marks that occur *within* strings or identifiers.

IMPORTANT In a future release, InterBase will allow only the dialect 3 behavior: double quotes will be interpreted only as delimited identifiers. Only single quotes will delimit string constants.

DDL AND DML STATEMENTS

When version 6 servers detect that the client is dialect 1, they permit client DML statement to contain double quotes and they correctly handle these as string constants. However, they do not permit double quotes in client DDL statements because that metadata would not be allowed in dialect 3. Version 6 servers all insist that string constants be delimited with single quotes when clients create new metadata.

The DATE, TIME, and TIMESTAMP datatypes

InterBase 6 introduces two new datatypes: TIME and TIMESTAMP. In addition, the meaning of the DATE datatype changes radically in dialect 3.

- InterBase 6, dialect 3 has full support for DATE, TIME, and TIMESTAMP (formerly DATE) datatypes. DATE and TIME store and return only the date and time portion of a timestamp and are available only in a dialect 3 database using a dialect 3 client.
- In dialect 3, DATE and TIME columns require only four bytes of storage, while TIMESTAMP columns require eight bytes.
- In version 5 and earlier, InterBase used the DATE datatype to store the entire timestamp. In version 6 dialect 1, this is still true, but the name of the datatype changes to TIMESTAMP when you migrate a database from version 5 to version 6 dialect 1. The functionality does not change. The DATE and TIMESTAMP datatypes are both available and both mean the same thing in dialect 1: TIMESTAMP.

The following example show the differences between dialect 1 and dialect 3 clients when date information is involved.

Example

```
CREATE TABLE table1 (fld1 DATE, fld2 TIME);
INSERT INTO table1 VALUES (CURRENT_DATE, CURRENT_TIME);
```

Using dialect 1 clients

```
SELECT * FROM table1;

   Statement failed, SQLCODE = -804
   Dynamic SQL Error
   -SQL error code = -804
   -Data type unknown
   -Client SQL dialect 1 does not support reference to TIME datatype

SELECT fld1 FROM table1;

   Statement failed, SQLCODE = -206
   Dynamic SQL Error
   -SQL error code = -206
   -Column unknown
   -FLD1
   -Client SQL dialect 1 does not support reference to DATE datatype
```

Using dialect 3 clients

```
SELECT * FROM table1;

   FLD1          FLD2
   =========== =============
   1999-06-25  11:32:30.0000

SELECT fld1 FROM table1;

   FLD1
   ===========
   1999-06-25
```

Example

```
CREATE TABLE table1 (fld1 TIMESTAMP);
INSERT INTO table1 (fld1) VALUES (CURRENT_TIMESTAMP);
SELECT * FROM table1;
```

In dialect 1:

```
   FLD1
   ===========
   25-JUN-1999
```

In dialect 3:

```
    FLD1
    =========================
    1999-06-25 10:24:35.0000
```

Example `SELECT CAST (fld1 AS CHAR(5)) FROM table1;`

In dialect 1:

```
    ======
    25-JU
```

In dialect 3:

```
    Statement failed, SQLCODE = -802
    arithmetic exception, numeric overflow, or string truncation
```

The EXTRACT operator allows you to return different parts of a TIMESTAMP value. The EXTRACT operator makes no distinction between dialects when formatting or returning the information.

```
SELECT EXTRACT (YEAR FROM timestamp_fld) FROM table_name;
    =======
    1999

SELECT EXTRACT (YEAR FROM timestamp_fld) FROM table_name;
    =======
    1999

SELECT EXTRACT (MONTH FROM timestamp_fld) FROM table_name;
    =======
    6

SELECT EXTRACT (DAY FROM timestamp_fld) FROM table_name;
    =======
    25

SELECT EXTRACT (MINUTE FROM timestamp_fld) FROM table_name;
    =======
    24

SELECT EXTRACT (SECOND FROM timestamp_fld) FROM table_name;
    ============
    35.0000

SELECT EXTRACT (WEEKDAY FROM timestamp_fld) FROM table_name;
    =======
    5
```

```
SELECT EXTRACT (YEARDAY FROM timestamp_fld) FROM table_name;
   =======
   175

SELECT EXTRACT (MONTH FROM timestamp_fld) ||
 '-' || EXTRACT (DAY FROM timestamp_fld) ||
 '-' || EXTRACT (YEAR FROM timestamp_fld) FROM table_name;

   ===================
   6-25-1999
```

CONVERTING TIMESTAMP COLUMNS TO DATE OR TIME

Once you have migrated a database to dialect 3, any columns that previously had the DATE datatype will have the TIMESTAMP datatype. If you want to store that data in a DATE or TIME column, follow these steps:

1. Use ALTER TABLE to create a new column of the desired type.
2. Insert the values from the original column into the new column:

   ```
   UPDATE tablename SET new_field = CAST (old_field AS new_field);
   ```

3. Use ALTER TABLE to drop the original column.
4. Use ALTER TABLE … ALTER COLUMN to rename the new column.

CASTING DATE/TIME DATATYPES

InterBase 6 dialect 3 no longer allows the use of the CAST operator to remove the data portion of a timestamp by casting the timestamp value to a character value. When you cast a TIMESTAMP to a CHAR or VARCHAR in dialect 3, the destination type must be at least 24 characters in length or InterBase will report a string overflow exception. This is required by the SQL3 standard.

In dialect 3, you can use the EXTRACT operator to extract portions of a timestamp. The *Release Notes* for InterBase 6 contain a fairly detailed discussion of casting date/time datatypes and other date/time manipulations.

DECIMAL and NUMERIC datatypes

The following sections highlight some of the changes introduced by InterBase 6 when dealing with numeric values. They need to be considered carefully when migrating your database from dialect 1 to dialect 3. When considering these issues, keep in mind that in order to make use of the new functionality, the statements must be created with a client dialect setting of 3.

In this section, an "exact numeric" refers to any of the following data types: INTEGER, SMALLINT, DECIMAL, NUMERIC. NUMERIC and DECIMAL datatypes that have a precision greater than 9 are called "large exact numerics" in this discussion. Large exact numerics are stored as DOUBLE PRECISION in dialect 1 and as INT64 in columns created in dialect 3.

IMPORTANT When you migrate an exact numeric column to dialect 3 it is still stored as DOUBLE PRECISION. The migration does not change the way the data is stored because INT64 cannot store the whole range that DOUBLE PRECISION can store. There is potential data loss, so InterBase does not permit direct conversion. If you decide that you want your data stored as INT64, you must create a new column and copy the data. Only exact numeric columns that are *created* in dialect 3 are stored as INT64. The details of the process are provided in "Migrating databases to dialect 3" on page 37.

You might or might not want to change exact numeric columns to INT64 when you migrate to dialect 3. See "Do you really need to migrate your NUMERIC and DECIMAL datatypes?" on page 42 for a discussion of issues.

Dialect 3 features and changes include

- Support for 64 bit integers.
- Overflow protection. In dialect 1, if the product of two integers was bigger than 31 bits, the product was returned modulo 2^{32}. In dialect 3, the true result is returned as a 64-bit integer. Further, if the product, sum, difference, or quotient of two exact numeric values is bigger than 63 bits, InterBase issues an arithmetic overflow error message and terminates the operation. (Previous versions sometimes returned the least-significant portion of the true result.). The stored procedure **bignum** below demonstrates this.

Operations involving division return an exact numeric if both operands are exact numerics in dialect 3. When the same operation is performed in dialect 1, the result is a DOUBLE PRECISION.

To obtain a DOUBLE PRECISION quotient of two exact numeric operands in dialect 3, explicitly cast one of the operands to DOUBLE PRECISION before performing the division:

```
CREATE TABLE table 1 (n1 INTEGER, n2 INTEGER);

INSERT INTO table 1 (n1, n2) VALUES (2, 3);

SELECT n1 / n2 FROM table1;

   ======================
                        0
```

Similarly, to obtain a double precision value when averaging an exact numeric column, you must cast the argument to double precision before the average is calculated:

```
SELECT AVG(CAST(int_col AS DOUBLE PRECISION))FROM table1;
```

Please see the *Release Notes* for a description of arithmetic operations with exact numerics in InterBase 6.

Compiled objects

The behavior of a compiled object such as a stored procedure, trigger, check constraint, or default value depends on the dialect setting of the client at the time the object is compiled. Once compiled and validated by the server the object is stored as part of the database and its behavior is constant regardless of the dialect of the client that calls it.

Example Consider the following procedure:

```
CREATE PROCEDURE exact1 (a INTEGER, b INTEGER) RETURNS (c INTEGER)
AS BEGIN
   c = a / b;
   EXIT;
END;
```

When created by a dialect 1 client:

EXECUTE PROCEDURE *exact 1* returns 1 when executed by either a dialect 1 or dialect 3 client.

When created by a dialect 3 client:

EXECUTE PROCEDURE *exact 1* returns 0 when executed by either a dialect 1 or dialect 3 client.

Example Consider the following procedure:

```
CREATE PROCEDURE bignum (a INTEGER, b INTEGER) RETURNS (c NUMERIC(18,0)
AS BEGIN
   c = a * b;
   EXIT;
END;
```

When created by a dialect 1 client:

EXECUTE PROCEDURE *bignum* (65535, 65535) returns –131071.0000 when executed by either a dialect 1 or dialect 3 client.

When created by a dialect 3 client:

EXECUTE PROCEDURE *bignum* (65535, 65535) returns `*ERROR* can't access INT64` when executed by a dialect 1 client.

EXECUTE PROCEDURE *bignum* (65535, 65535) returns 4294836225 when executed by a dialect 3 client.

Generators

InterBase 6 generators return a 64-bit value, and only wrap around after 2^{64} invocations (assuming an increment of 1), rather than after 2^{32} as in InterBase 5. Applications should use an *ISC_INT64* variable to hold the value returned by a generator. A client using dialect 1 receives only the least significant 32 bits of the updated generator value, but the entire 64-bit value is incremented by the engine even when returning a 32-bit value to a client that uses dialect 1. If your database was using an INTEGER field for holding generator values, you need to recreate the field so that it can hold 64-bit integer values.

Miscellaneous issues

- IN clauses have a limit of 1500 elements

  Resolution If you have more than 1500 elements, place the values in a temporary table and use a SELECT subquery in place of the list elements.

- Arithmetic operations on character fields are no longer permitted in client dialect 3

  Resolution Explicitly cast the information before performing arithmetic calculations.

- Using isql to select from a TIMESTAMP column displays all information when client dialect is 3.

  Resolution In versions of InterBase prior to 6.0, the time portion of a timestamp displayed only if SET TIME ON was in effect. In 6.0 client dialect 3, the time portion of the timestamp always displays.

Migrating servers and databases

You can migrate your servers and applications to InterBase 6 at different times. They are separate migrations. Bear the following issues in mind as you plan your migration:

- When you migrate a server to InterBase 6, you must also migrate all the databases on that server.
- Older clients can still access databases that have been migrated to InterBase 6. You must be aware, however, that they cannot access new datatypes or data stored as INT64, and they always handle double quoted material as strings.
- InterBase strongly recommends that you establish a migration testbed to check your migration procedures before migrating production servers and databases. The test bed does not need to be on the same platform as the production clients and servers that you are migrating.

The migration path varies somewhat depending on whether you are replacing an existing server or installing a new server and moving old databases there. Upgrading an existing server costs less in money, but may cost more in time and effort. The server and all the databases that must migrate with it are unavailable during the upgrade. If you have hardware available for a new InterBase 6 server, the migration can be done in parallel, without interrupting service more than very briefly. This option also offers an easier return path if problems arise with the migration.

"In-place" server migration

This section describes the recommended steps for replacing an InterBase 5 server with an InterBase 6 server.

1. Shut down each database before backup to ensure that no transactions are in progress.
2. Back up all databases on the version 5 server. Include **isc4.gdb** if you want to preserve your configured user IDs.

   As a precaution, you should validate your databases before backing up and then restore each database to ensure that the backup file is valid.
3. Shut down the version 5 server. If your current server is a Superserver, you are not required to uninstall the server if you intend to install over it, although uninstalling is always good practice. You cannot have multiple versions of InterBase on the same machine. If your current server is Classic, you *must* uninstall before installing InterBase 6.
4. Install the version 6 server.

Note The install does not overwrite **isc4.gdb** or **isc4.gbk**.

5. Start the new server.
   - On WindowsNT, go to Services in the Control Panel and start the InterBase Guardian.
   - On Windows 9x, run the InterBase Guardian application.
   - On UNIX platforms, issue the following command to start the InterBase Superserver as user "interbase":

     ```
     # echo "/usr/interbase/bin/ibmgr -start -forever" | su interbase
     ```

     Note that InterBase can run only as user "root" or user "interbase" on UNIX.
6. To restore the list of valid users, follow these steps:
   a. Restore **isc4.gbk** to **isc4_old.gdb**
   b. Shut down the server
   c. Copy **isc4_old.gdb** over **isc4.gdb**
   d. Copy **isc4_old.gbk** over **isc4.gbk**
   e. Restart the server
7. Delete each ODS 9 database file. Restore each database from its backup file. This process creates InterBase 6, ODS 10, dialect 1 databases.
8. Perform a full validation of each database.

After performing these steps, you have an InterBase 6 server and InterBase 6, dialect 1 databases. See "About InterBase 6, dialect 1 databases" on page 36 to understand more about these databases. See "Migrating databases to dialect 3" on page 37 for a description of how to migrate databases to dialect 3. See "Migrating clients" on page 45 for an introduction to client migration.

Migrating to a new server

This section describes the recommended steps for installing InterBase 6 as a new server and then migrating databases from a previous InterBase 5 installation. The process differs only slightly from an in-line install.

1. Back up all databases on the version 5 server. Include **isc4.gdb** if you want to preserve your configured user IDs. Shut down the databases before backup to ensure that no transactions are in progress.
2. Install the version 6 server.

3. Start the new version 6 server.
   - On WindowsNT, go to Services in the Control Panel and start the InterBase Guardian.
   - On Windows 9x, run the InterBase Guardian application.
   - On UNIX platforms, issue the following command to start the InterBase Superserver as user “interbase”:

     ```
     # echo "/usr/interbase/bin/ibmgr -start -forever" | su interbase
     ```

     Note that InterBase can run only as user “root” or user “interbase” on UNIX.
4. Copy the database backup files to the new server and restore each database from its backup file. This process creates InterBase 6, ODS 10, dialect 1 databases.

   Save your backup files until your migration to dialect 3 is complete.
5. To restore the list of valid users, follow these steps:
   a. Restore **isc4.gbk** to **isc4_old.gdb**
   b. Shut down the server
   c. Copy **isc4_old.gdb** over **isc4.gdb**
   d. Copy **isc4_old.gbk** over **isc4.gbk**
   e. Restart the server
6. Perform a full validation of each database on the new server.

After performing these steps, you have an InterBase 6 server and InterBase 6, dialect 1 databases. See “About InterBase 6, dialect 1 databases” on page 36 to understand more about these databases. See “Migrating databases to dialect 3” on page 37 for a description of how to migrate databases to dialect 3. See “Migrating clients” on page 45 for an introduction to client migration.

About InterBase 6, dialect 1 databases

When you back up a version 5 database and restore it in InterBase 6, what do you have?

- A version 5 client can access everything in the database with no further changes.
- If there are object names—column or table names, for instance—that include any of the 17 new keywords, you must change these names in order to access these objects with a version 6 dialect 1 client. The new ALTER COLUMN clause of ALTER TABLE makes it easy to implement column name changes.

    · Version 5 clients can still access the columns.
    · Dialect 3 clients can access these columns as long as they delimit them with double quotes.

- The 17 new keywords are reserved words. However, the new datatypes TIME and DATE are not available to use a datatypes. DATE columns have the old meaning—both date and time. The new meaning of DATE—date only—is available only in dialect 3.
- All columns that were previously DATE datatype are now TIMESTAMP datatype. TIMESTAMP contains exactly the information that DATE did in previous versions.
- Exact numeric columns—those that have a DECIMAL or NUMERIC datatype with precision greater than 9—are still stored as DOUBLE PRECISION datatypes. All arithmetic algorithms that worked before on these columns still work as before. It is not possible to store data as INT64 in dialect 1.

Migrating databases to dialect 3

There are four major areas of concern when migrating a database from dialect 1 to dialect 3: double quotes, the DATE datatype, large exact numerics (for purposes of this discussion, NUMERIC and DECIMAL datatypes that have a precision greater than 9), and keywords.

The process varies somewhat depending on whether you can create an application to move data from your original database to an empty dialect 3 database. If you do not have access to such a utility, you need to perform an in-place migration of the original database.

Overview

In either method, you begin by extracting the metadata from your database, examining it for problem areas, and fixing the problems.

- If you are performing an in-place migration, you copy corrected SQL statements from the metadata file into a new script file, modify them, and run the script against the original database. Then you set the database to dialect 3. There are some final steps to take in the dialect 3 database to store old data as INT64.
- If you have a utility for moving data from the old database to a newly created empty database, you use the modified metadata file to create a new dialect 3 database and use the utility to transfer data from the old database to the new.

In both cases, you must make changes to the new database to accommodate migrated columns that must be stored as INT64 and column constraints and defaults that originally contained double quotes.

The two methods are described below.

Method one: in-place migration

1. If you have not migrated the database to version 6, dialect 1, do so first. Back up the database again.
2. Extract the metadata from the database using **isql -*x***. If you are migrating legacy databases that contain GDML, see “Migrating older databases” on page 45.
3. Prepare an empty text file to use as a script file. As you fix data structures in the metadata files, you will copy them to this file to create a new script.

   Note You could also proceed by removing unchanged SQL statements from the original metadata file, but this is more likely to result in problems from statements that were left in error. InterBase recommends creating a new script file that contains only the statements that need to be run against the original database.

For the remaining steps, use a text editor to examine and modify the metadata and script files. Place copied statements into the new script file in the same order they occur in the metadata file to avoid dependency errors.

4. Search for each instance of double quotes in the extracted metadata file. These can occur in triggers, stored procedures, views, domains, table column defaults, and constraints. Change each double quote that delimits a string to a single quote. Make a note of any tables that have column-level constraints or column defaults in double quotes.

   Copy each changed statement to your empty script file, but do not copy ALTER TABLE statements whose only double quotes are in column-level constraints or column defaults.

   Important When copying trigger or stored procedure code, be sure to include any associated SET TERM statements.

Quoted quotes If there is any chance that you have single or double quotes *inside* of strings, you must search and replace on a case-by-case basis to avoid inappropriate changes. The handling of quotation marks within strings is as follows:

String:	`In "peg" mode`
Double-quoted:	`"In ""peg"" mode"`
Single-quoted:	`'In "peg" mode'`
String:	`O'Reilly`
Double-quoted:	`"O'Reilly"`
Single-quoted:	`'O''Reilly'`

TABLE 2.1 Handling quotation marks inside of strings

5. In the new script file, search for occurrences of the TIMESTAMP datatype. In most cases, these were DATE datatypes in your pre-6 database. For each one, decide whether you want it to be TIME, TIMESTAMP, or DATE in your dialect 3 database. Change it as needed.
6. Repeat step 5 in the metadata file. Copy each changed statement to your new script file.
7. In the new script file, search for occurrences of reserved words that are used as object names and enclose them in double quotes; that makes them delimited identifiers.
8. Repeat step 7 in the metadata file. Copy each changed statement to your new script file.
9. In each of the two files, search for each instance of a DECIMAL or NUMERIC datatype with a precision greater than 9. Consider whether or not you want data stored in that column or with that domain to be stored as DOUBLE PRECISION or INT64. See “Do you really need to migrate your NUMERIC and DECIMAL datatypes?” on page 42 for a discussion of issues. For occurrences that should be stored as DOUBLE PRECISION, change the datatype to that. Leave occurrences that you want stored as INT64 alone for now. Copy each changed statement that occurs in the metadata file to your new script file.

Perform the following steps in your new script file:

10. Locate each CREATE TRIGGER and CREATE DOMAIN statement and change it to ALTER TRIGGER or ALTER DOMAIN as appropriate.
11. Locate each CREATE VIEW statement. Precede it by a corresponding DROP statement. For example, if you have a CREATE VIEW *foo* statement, put a DROP VIEW *foo* statement right before it, so that when you run this script against your database, each view first gets dropped and then re-created.
12. If you have any ALTER TABLE statements that you copied because they contain named table-level constraints, modify the statement so that it does nothing except drop the named constraint and then add the constraint back with the single quotes.
13. Check that stored procedure statements are ALTER PROCEDURE statements. This should already be the case.
14. At the beginning of the script, put a CONNECT statement that connects to the original database that you are migrating.
15. Make sure your database is backed up and run your script against the database.
16. Use **gfix** to change the database dialect to 3.

    ```
    gfix -sql_dialect 3 database.gdb
    ```

    Note To run **gfix** against a database, you must attach as either the database owner or SYSDBA.
17. At this point, DECIMAL and NUMERIC columns with a precision greater than 9 are still stored as DOUBLE PRECISION. To store the data as INT64, follow the steps in “Migrating NUMERIC and DECIMAL datatypes” on page 42.
18. Validate the database using either IBConsole or **gfix**.

That’s it. You’ve got a dialect 3 database. There is a little more work to do if you want your NUMERIC and DECIMAL columns with a precision of greater than 9 to be stored as INT64. At this point, they are still stored as DOUBLE PRECISION. To decide whether you want to change they way data in these columns is stored, read

In addition, there are some optional steps you can take that are described in the following sections, “Column defaults and column constraints” and “Unnamed table constraints”.

IMPORTANT If you ever extract metadata from the dialect 3 database that you created using the steps above, and if you plan to use that metadata to create a new database, check to see if the extracted metadata contains double quotes delimiting string constants in column defaults, column constraints, or unnamed table constraints. Change any such occurrences to single quotes before using the metadata to create the new database.

Column defaults and column constraints

The steps above permitted you to retain double quoted string constants in column defaults, column constraints, and unnamed table constraints. This is possible because, once created, InterBase stores them in binary form.

Following the steps above creates a dialect 3 database that is fully functional, but if it contains double quoted string constants in column defaults, column constraints, or unnamed column constraints, inconsistencies are visible when you SHOW metadata or extract it. You can choose to resolve these inconsistencies by following these steps:

1. Back up the database.
2. Examine the metadata to detect each occurrence of a column default or column constraint that uses double quotes.
3. For each affected column, use the ALTER COLUMN clause of the ALTER TABLE statement to give the column a temporary name. If column position is likely to be an issue with any of your clients, change the position as well.
4. Create a new column with the desired datatype, giving it the original column name and position.
5. Use UPDATE to copy the data from old column to the new column:

```
UPDATE table_name
   SET new_col = old_col;
```

6. Drop the old column.

Unnamed table constraints

Read the first two paragraphs under “Column defaults and column constraints” on page 41 to understand why you don’t always need to change constraints with double quotes to single-quoted form, and why you might want to change them.

To bring unnamed table constraints that contain double quotes into compliance with the dialect 3 standard, follow these steps:

1. Back up the database.
2. Examine the metadata to detect each occurrence of an unnamed table constraint that uses double quotes.

3. For each occurrence, use SHOW TABLE to see the name that InterBase has assigned to the constraint.
4. Use ALTER TABLE to drop the old constraint, using the name given in the SHOW TABLE output and add a new constraint. For ease in future handling, give the constraint a name.

   If SHOW TABLE shows that InterBase stores the unnamed constraint as “INTEG_2”, then issue the following statement to change the constraint:

```
ALTER TABLE foo
    DROP CONSTRAINT INTEG_2,
    ADD CONSTRAINT new_name
        CHECK (col_name IN ('val1', 'val2', 'val3'));
```

Migrating NUMERIC *and* DECIMAL *datatypes*

If you want NUMERIC and DECIMAL datatypes with a precision greater than 9 to be stored as exact numerics, you must take some extra steps after migrating to dialect 3. The following sections tell you how to decide whether you really need to take these steps and how to perform them if you decide you want the exact numerics.

DO YOU REALLY NEED TO MIGRATE YOUR NUMERIC AND DECIMAL DATATYPES?

Future versions of InterBase will no longer support dialect 1. It is offered now as a transitional mode. As you migrate you databases to dialect 3, consider the following questions about columns defined with NUMERIC and DECIMAL datatypes:

- Is the precision less than 10? If so, there is no issue. You can migrate without taking any action and there will be no change in the database and no effect on clients.
- For NUMERIC and DECIMAL columns with precision greater than 9, is DOUBLE PRECISION an appropriate way to store your data?
  - In many cases, the answer is “yes.” If you want to continue to store your data as DOUBLE PRECISION, change the datatype of the column to DOUBLE PRECISION either before or after migrating your database to dialect 3. This doesn’t change any functionality in dialect 3, but it brings the declaration into line with the storage mode. In a dialect 3 database, newly-created columns of this type are stored as INT64, but migrated columns are still stored as DOUBLE PRECISION. Changing the declaration avoids confusion.
  - DOUBLE PRECISION may not be appropriate or desirable for financial applications and others that are sensitive to rounding errors. In this case, you need to take steps to migrate your column so that it is stored as INT64 in dialect 3. As you make this decision, remember that INT64 does not store the same range as DOUBLE PRECISION. Check whether you will experience data loss and whether this is acceptable.

MIGRATING NUMERIC AND DECIMAL DATATYPES

Read "Do you really need to migrate your NUMERIC and DECIMAL datatypes?" on page 42 to decide whether you have columns in a dialect 1 database that would be best stored as 64-bit integers in a dialect 3 database. If this is the case, follow these steps for each column:

1. Migrate your database to InterBase 6 as described in "Method one: in-place migration" on page 38.
2. Use the ALTER COLUMN clause of the ALTER DATABASE statement to change the name of each affected column to something different from its original name. If column position is going to be an issue with any of your clients, use ALTER COLUMN to change the positions as well.
3. Create a new column for each one that you are migrating. Use the original column names and if necessary, positions. Declare each one as a DECIMAL or NUMERIC with precision greater than 9.
4. Use UPDATE to copy the data from each old column to its corresponding new column:

```
UPDATE tablename
    SET new_col = old_col;
```

5. Check that your data has been successfully copied to the new columns and drop the old columns.

Method two: migrating to a new database

If you can create a data transfer utility that copies data between databases, the process of migrating a database to dialect 3 is considerably simplified.

Overview Extract the metadata from your database, examine it for problem areas, and fix the problems. Use the modified metadata file to create a new dialect 3 database and use an application to transfer data from the old database to the new.

1. If you have not migrated the database to version 6, dialect 1, do so first. Back up the database again.
2. Extract the metadata from the database using **isql -x**. If you are migrating a database that contains data structures created with GDML, see "Migrating older databases" on page 45.

For the following steps, use a text editor to examine and modify the metadata file.

3. Search for each occurrence of the TIMESTAMP datatype. In most cases, these were DATE datatypes in your pre-6 database. Decide whether you want it to be TIME, TIMESTAMP, or DATE in your dialect 3 database. Change it as needed.
4. Find all instances of reserved words that are used as object names and enclose them in double quotes to make them delimited identifiers.
5. Search for each instance of double quotes in the extracted metadata file. These can occur in triggers, stored procedures, views, domains, exceptions, table column defaults, and constraints. Change each double quote to a single quote.
6. Search for each instance of a DECIMAL or NUMERIC datatype with a precision greater than 9. Consider whether or not you want that data stored as DOUBLE PRECISION or INT64. See "Do you really need to migrate your NUMERIC and DECIMAL datatypes?" on page 42 for a discussion of issues. For occurrences that should be stored as DOUBLE PRECISION, change the datatype to that. Leave occurrences that you want stored as INT64 alone for now.
7. At the beginning of the file, enter SET SQL DIALECT 3. On the next line, uncomment the CREATE DATABASE statement and edit it as necessary to create a new database.
8. Run the metadata file as a script to create a new database.
9. Use your data transfer utility to copy data from the old database to the new dialect 3 database. In the case of a large database, allow significant time for this.
10. Validate the database using **gfix**.
11. At this point, DECIMAL and NUMERIC columns with a precision greater than 9 are still stored as DOUBLE PRECISION. To store the data as INT64, follow the steps in "Migrating NUMERIC and DECIMAL datatypes" on page 42.

▸ *Migrating older databases*

If you have legacy databases in which some data structures were created with GDML, you may need to extract metadata in a slightly different way.

1. Try extracting metadata as described in Step 2 above and examine it to see if all tables and other DDL structures are present. If they are not, delete the metadata file and extract using the **-a** switch instead of the **-x** switch. This extracts objects created in GDML.
2. You may have to change some of the code to SQL form. For example, the following domain definition

   ```
   CREATE DOMAIN NO_INIT_FLAG AS SMALLINT
       ( no_init_flag = 1 or
       no_init_flag = 0 or
       no_init_flag missing);
   ```

   needs to be translated to:

   ```
   CREATE DOMAIN NO_INIT_FLAG AS SMALLINT
       CHECK ( VALUE = 1 OR VALUE = 0 OR VALUE IS NULL );
   ```

3. Some code may be commented out. For example:

   ```
   CREATE TABLE BOILER_PLATE (BOILER_PLATE_NAME NAME,
        DATE DATE,
        CREATED_DATE COMPUTED BY /*  Date */);
   ```

   needs to be changed to:

   ```
   CREATE TABLE BOILER_PLATE (BOILER_PLATE_NAME NAME,
       "DATE" DATE,
        CREATED_DATE COMPUTED BY "DATE");
   ```

Migrating clients

To migrate an older client application to InterBase 6, install the InterBase 6 client onto the platform where the client application resides. An InterBase server then recognizes that client as a version 6 dialect 1 client.

It is good practice to recompile and relink the application and make note of field names, datatype use, and so on in the new application. When you recompile, state the dialect explicitly:

```
SET SQL DIALECT n;
```

IMPORTANT If you have databases that use any of the new version 6 keywords as object identifiers and you are not migrating those databases to dialect 3, you might want to not migrate your version 5 clients. If you migrate them to version 6 dialect 1, you lose the ability to access those keyword columns. See “New keywords” on page 25.

When you recompile an existing **gpre** client, you must recompile it with the **gpre -sql_dialect *n*** switch.

There are several paths that permit you to create dialect 3 clients that access all new InterBase 6 features:

- In Delphi 5, make calls to functions in the new InterBase Express (IBX) package.
  - Since the Delphi 5 beta includes InterBase 5.5, it ships with a version of IBX that does not include calls to the new InterBase 6 Services, Install, and Licensing APIs.
  - If you are a beta tester for both Delphi 5 and interBase 6, install the expanded IBX package that is included with the InterBase 6 beta. This InterBase 6 IBX package permits you to do everything that the Delphi 5 version does, but adds calls to the Services, Install, and Licensing APIs.
- To write embedded SQL applications that address all InterBase 6 dialect 3 functionality, compile them using **gpre -sql_dialect 3**.

Client	How to migrate
Older applications such as InterBase version 5 applications	• Dialect is 1; there is no way to change the dialect • A version 5 client application becomes version 6 dialect 1 client whenever the InterBase 6 client is installed on the machine with the client
ISQL	• Issue the command line option: `-sql_dialect` *n* • Or issue the command `SET SQL DIALECT` *n*`;`

GPRE	• Issue the command line option
	`-sql_dialect` *n*
	• Or issue the command
	`EXEC SQL`
	`SET SQL DIALECT` *n*`;`

BDE	All applications use SQL dialect 1. To access InterBase dialect 3 features from Delphi 5, use the IBX components
InterClient	All applications use SQL dialect 1
Direct API calls	Set the dialect parameter on *isc_dsql_execute_immediate*(), *isc_dsql_exec_immed2*(), *isc_dsql_prepare*() API calls to the desired dialect value: 1 or 3

TABLE 2.2 Migrating clients: summary