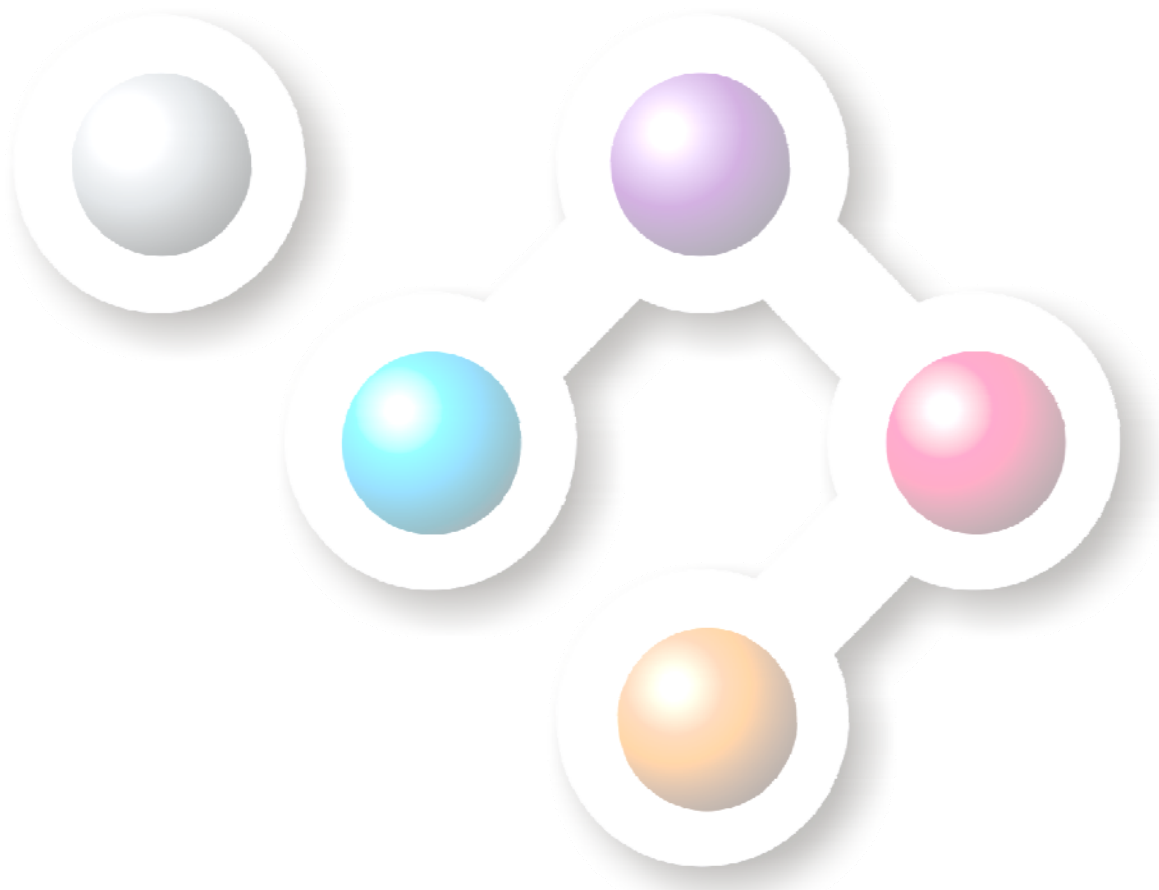


Lycia Development Suite



**Lycia Database Migration Guide for IBM DB2
Versions 5 and 6 – January 2011**



Dynamic Database Interface

Migration Guide for IBM DB2

Versions 5 and 6

January 2011

Part Number: 015-006-135-011

Querix Dynamic Database Guide for IBM DB2

Copyright 2004-2011 Querix (UK) Limited. All rights reserved.

February 2011 Part Number: 015-006-135-011

Published by:

Querix Ltd, 50 The Avenue, Southampton, SO17 1XQ, UK

Publication history:

May 2004:	First edition titled 'Multiple Database Guide'
September 2004:	Second edition, including database
June 2005:	Updated for 'Hydra4GL' version 4.2
July 2005:	Retitled 'Hydra4GL' and Your Database
January 2008:	Dedicated for IBM DB2, updated for versions 4.3 and retitled 'Database Migration Guide for IBM DB2'
December 2008:	v4.31 update
February 2009:	v4.4 updated
January 2011:	v5 and v6 updated

Last Updated:

January 2011

Documentation written by:

Sean Sunderland/Gemma Davis

Notices:

The information contained within this document is subject to change without notice. If you find any problems in the documentation please submit your comments to documentation@querix.com. No part of this document may be reproduced or transmitted, in any form or by any means, electronic or mechanical, for any purpose without the express permission of Querix (UK) Ltd.

Other products or company names used within this document are for identification purposes only, and may be trademarks of their respective owners.

Contents

CONTENTS.....	7
ABOUT THIS GUIDE	1
<i>Who should read this Reference Guide?</i>	1
<i>Conventions used in this book</i>	1
Typefaces and Icons	1
CHAPTER 1	3
INTRODUCTION	3
CHAPTER 2	5
PREREQUISITES FOR CONVERTING TO IBM DB2	5
CHAPTER 3	7
INFORMIX AND IBM DB2 COMPARED	7
<i>General Comparison</i>	7
<i>Isolation Levels in Informix</i>	8
<i>Isolation Levels in IBM DB2</i>	8
LYCIA DATABASE CONNECTIONS.....	9
Environment Settings.....	9
Concepts	9
Common Errors	9
CHAPTER 4	11
CONNECTING TO IBM DB2.....	11
<i>Connecting to IBM DB2</i>	11
Environment Settings.....	11
Concepts	11
Connecting to DB2 on Windows	11
Connecting to a Local DB2 Instance on Unix.....	14
Connecting to a Remote DB2 Instance on Unix.....	14
CHAPTER 5	17
PREPARING FOR THE IBM DB2 CONVERSION.....	17
The Conversion Process	17
Migration of the Database Schema	17
Converting the Database Schema	18
Loading Table Data.....	18
Compiling 4GL Code	19
Application Testing.....	19
The qxexport Schema Extraction Tool	20
CHAPTER 6	21
INFORMIX COMPATIBILITY	21
<i>Unhandled Problems</i>	22

Matches	22
Unsupported SQL.....	23
System Catalogues.....	23
Column Defaults.....	23
<i>Problems Handled Through Emulation</i>	24
Create Temp Table (with NO LOG option)	24
ROWID	24
<i>Problems Fully Handled</i>	25
Legal Parameter Context	25
Function/Operator Naming	25
DDL Syntax	26
Literal Type Mapping.....	26
Join Syntax.....	26
SQL Error Codes	26
<i>Configurable Behaviour</i>	29
APPENDIX A	30
THE LYCIA DYNAMIC SQL TRANSLATOR.....	30
<i>Introduction</i>	30
<i>Concepts</i>	31
Database Drivers	31
Bind Variables	31
Buffer Variables	33
<i>How Dynamic Translation Works</i>	34
SQL interpretation.....	34
SQL Generation.....	34
Datatype Mappings	35
The qx_ \$schema table.....	35
Type Promotion	35
Benefits of Type Promotion.....	36
Areas where Type Promotion is used	36
APPENDIX B	37
DATATYPE MAPPINGS	37
APPENDIX C	41
COMMON CONVERSION ISSUES WHEN MIGRATING TO A DIFFERENT VENDOR'S DATABASE	41
System Catalogues	41
Matches	41
Cursor Names	42
Isolation Levels	42
Stored Procedures	42
INDEX.....	43

About This Guide

Who should read this Reference Guide?

This reference guide is intended to be used as a template for application developers who plan to modify existing applications or develop new applications to work with IBM DB2.

This document assumes a background in Informix® development environments and Informix databases.


Conventions used in this book

Typefaces and Icons

Constant-width text is used for code examples and fragments, or command line functions.

Constant-width bold is used for emphasis.

Constant-width text is used for code sample variables.

	<p>This icon indicates a suggestion, discusses prerequisites for the action about to be taken, or indicates a warning about the use of available options.</p>
---	---

Significant code fragments are generally reproduced in a monospace font like this:

```
database cms
main
    display "Hello World"
end main
```

Code examples shown in this document may be used within any Querix applications – no special permission is required.

CHAPTER 1

Introduction

Informix development tools, including Informix 4GL, Informix ESQL/C and Informix NewEra, were developed to interact solely with an Informix RDBMS. For this reason, applications developed using these tools are dependent upon the behaviour and personality of an Informix RDBMS.

Querix development tools, such as Lycia offer full compatibility with Informix development tools, whilst also offering the ability to operate seamlessly with a non-Informix RDBMS.

Any existing Informix application sources can be re-used and developed further using the 'Hydrda4GL' Development tools to open up for modern GUI screen interactions and databases such as IBM DB2.

Using Querix's unique dynamic SQL translation capabilities, the application vendor can still continue to develop using Informix style SQL without any re-training or code re-writes whilst being able to exploit Lycia's extended capabilities.

The result is that identical Informix application sources can operate against both Informix and non-Informix RDBMSs without compromising the investment in the original application sources.

Informix® 4GL, -ESQL-C and -NewEra® are development languages from Informix (now owned by IBM®) and for that reason, they could only interact with Informix databases.

CHAPTER 2

Prerequisites for Converting to IBM DB2

In order to use Lycia with IBM DB2, you must have the following:

- IBM DB2 version 8 or later installation (see <http://www.ibm.com/software/data/db2> for downloads and installation instructions)
- A full installation of the Lycia 4.5 application
- An understanding of 4GL programming
- Full access to the application's source code
- A working knowledge of the application

No Informix tools are needed

CHAPTER 3

Informix and IBM DB2 Compared

This chapter is intended to provide a brief overview of the similarities and differences between Informix (OnLine, Dynamic Server and Standard Engine) and IBM DB2.

General Comparison

Subject	Informix	IBM DB2
Concepts	The server is a container unit for multiple databases.	The server instance is a container for multiple databases.
Storage	<p>Informix Standard Engine uses a directory structure for physical data storage.</p> <p>Informix OnLine and Informix Dynamic Server both use a physical data file for the data storage of a server instance. Multiple databases can reside in one data file.</p>	Storage is spanned over tablespaces which can be comprised of 1 or more physical files.
SQL Compliance	SQL-92 (entry level) SQL-99 (partial) Proprietary extensions	SQL-92 (entry level) SQL-99 (partial) Proprietary extensions
Supported Connection Interfaces	Proprietary, ADO, OLEDB, ODBC, JDBC	SQL-CLI, ODBC
User Authentication	Operating System Managed	Operating System Managed, Kerberos Managed
Supported Isolation Levels	Dirty Read Committed Read Cursor Stability Repeatable Read	Uncommitted Read Read Stability Cursor Stability Repeatable Read

Isolation Levels in Informix

Dirty Read	Does not place or honor table locks whilst reading data, and will read uncommitted data.
Committed Read (ANSI)	Places no locks on data being read, but only reads data that has been committed. This is the default isolation level.
Cursor Stability	Whilst reading data, the current row is share-locked. A share locked row cannot be exclusively locked.
Repeatable Read	Acquires a share lock on all rows being read for the duration of a transaction. A share locked row cannot be exclusively locked.

Isolation Levels in IBM DB2

Cursor Stability	The current row cannot be updated by an external process, however no other level of locking is placed on data. Phantom data and Non-repeatable reads may occur.
Repeatable Read	All accessed rows/pages are locked. No phenomena will occur.
Read Stability	Accessed rows may not be modified by external processes. Phantom data may occur.
Uncommitted Read	Minimal level of row locking. Dirty data, Phantom data and Non-repeatable reads may occur.
No Commit	

Lycia Database Connections

Environment Settings

Because Lycia dynamically selects the database connection method at runtime, the same compiled program can be used to connect to different RDBMs, simply by making changes to the process environment.

- LYCIA_DB_DRIVER – The database connection method to use.

The connection method used by Lycia is determined by the environment variable LYCIA_DB_DRIVER. This may take one of the following values (these are not case sensitive):

- Informix – Connect to Informix using the Informix client libraries.
- Informixqx – Connect to Informix using the standalone Querix interface.
- IBM DB2 – Connect to IBM DB2.
- Sserver – Connect to Microsoft SQL Server.
- Db2 – Connect to IBM DB2.
- Odbc – Connect using ODBC (for UNIX / Linux this uses the unixODBC driver manager).
- Iodbc – Connect using the iODBC driver manager (UNIX / Linux only).

If the environment variable LYCIA_DB_DRIVER is unset, the runtime will use the default database connection method specified during installation.

Concepts

The Lycia database interface establishes a connection based on the connection method specified in the environment variable LYCIA_DB_DRIVER and the database name specified in the 4GL/ESQL 'DATABASE' statement.

In addition to accepting the standard Informix database name format in the DATABASE statement (e.g. 'database' or 'database@server') each connection method will also allow vendor specific database name formats. These are covered in detail in the connectivity chapter of the database guide for the RDBMS you are using.

Common Errors

The following error codes may be returned by the database interface.

- -994 – The connection method specified by LYCIA_DB_DRIVER could not be loaded. Check that the LYCIA_DB_DRIVER variable is set correctly and that your shared library search path is set correctly.

- -998 – Evaluation license limit. The database connection method you are using is running in evaluation mode. Evaluation mode limits a database connection to 1500 SQL statements.

CHAPTER 4

Connecting To IBM DB2

Connecting to IBM DB2

Environment Settings

- DB2DIR – base installation directory of IBM DB2.
- DB2INSTANCE – The DB2 instance for this connection.
- DBUSER (optional) – Specify the user ID for this connection.
- DBPASS (optional) – Specify the user password for this connection.
- QX_SCHEMA_OWNER (optional) – Specify the default schema for this connection (defaults to 'QUERIX').

Concepts

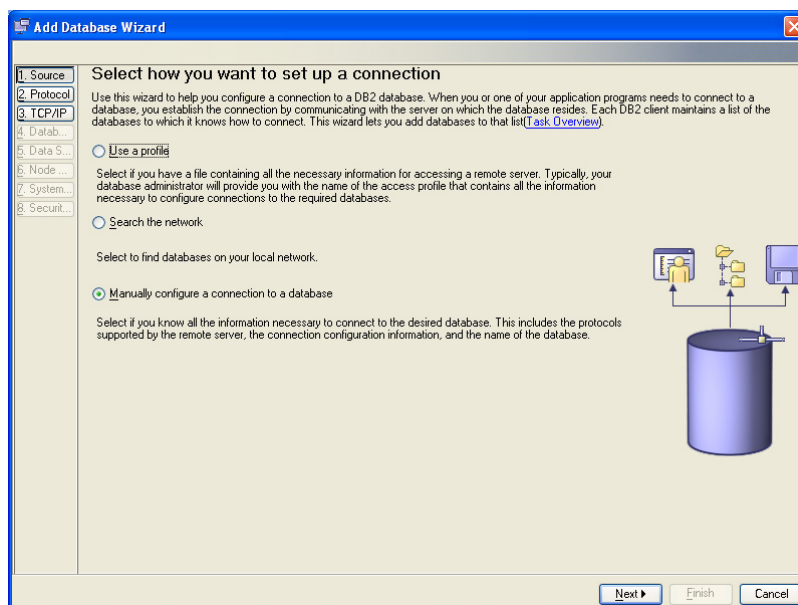
IBM DB2 is similar to Informix in that a server instances contains a collection of databases. For this reason, the most obvious mapping is to create a distinct DB2 database for each Informix database name referenced in the 4GL source code.

Whether connecting to a remote or local DB2 server, the database name you are trying to connect to must be catalogued in the instance specified in the \$DB2INSTANCE environment variable.

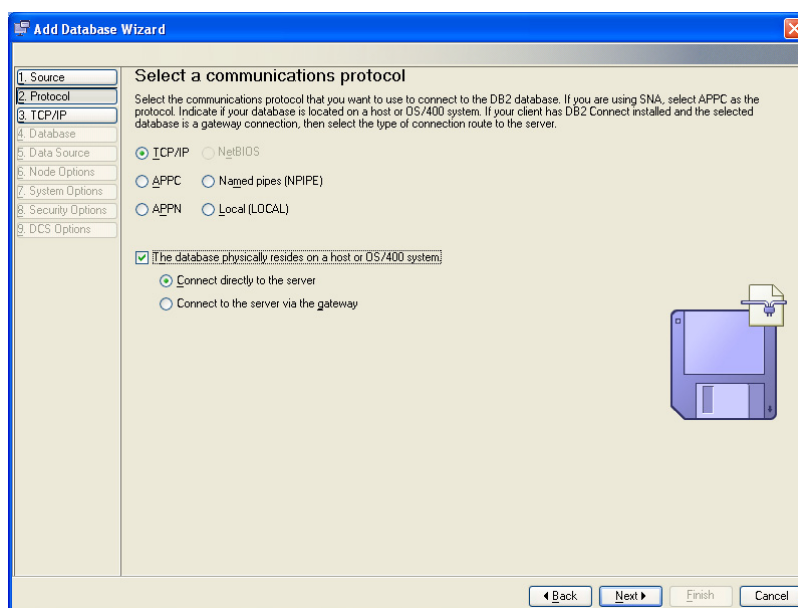
Connecting to DB2 on Windows

To connect to DB2 from Windows, it is first necessary to configure a connection. To do this, run the DB2 Configuration Assistant from the start menu.

For each 'database' name you wish to use in 4GL, it is necessary to configure a database connection with the database name as an alias. In the configuration assistant, select to manually configure a connection to a database.



When prompted for the protocol, specify whether or not you are connecting to a local or remote DB2 server instance. If you are connecting to a remote instance, set the communication protocol to TCP/IP, otherwise accept the server default.

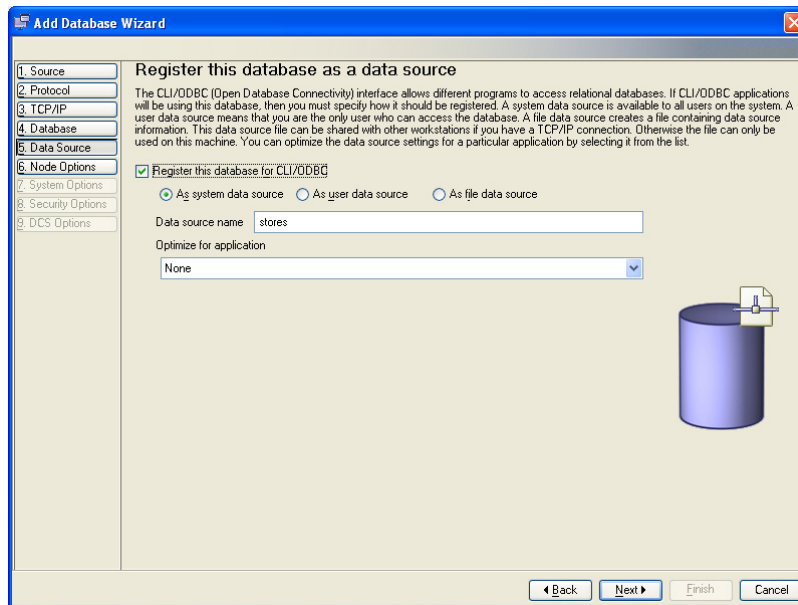


Under the heading 'Data Source' select to register the database for CLI/ODBC. The data source name you select here should match the 'database' name specified in the 4GL DATABASE statement you will be using to access this data source.

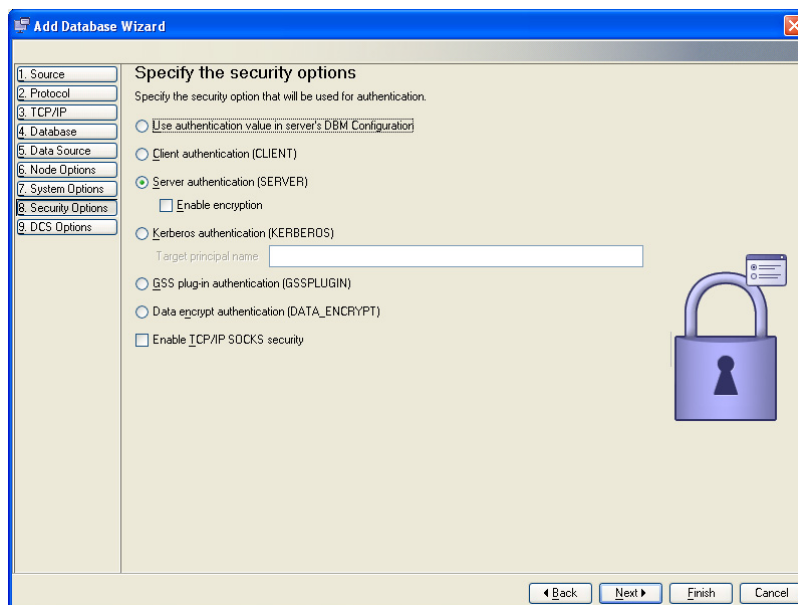
For example, for the 4GL statement

DATABASE stores

The Lycia database interface will try to access a data source named 'stores'.



Under the security settings select the appropriate authentication method for connecting to the server. Typically this will be using server authentication.



Connecting to a Local DB2 Instance on Unix

To connect to a local DB2 instance, it is first necessary to create the database(s) you will be connecting to. The Lycia database interface performs a 1:1 mapping between the database name specified in the 4GL 'DATABASE' statement, and the database name within the DB2 instance.

To create a database, issue the create database statement in the DB2 command processor:

```
db2 => create database <database name>
```

If you intend to use temporary tables within this database, it is necessary to assign a user temporary tablespace with a pagesize of at least 8192. For example

```
db2 => connect to <database name>
```

```
db2 => create user temporary tablespace <tablespace name> pagesize 8192
```

```
db2 => grant use of tablespace <tablespace name> to public
```

For example:

```
db2 => connect to stores
```

```
db2 => create user temporary tablespace usertempl pagesize 8192
```

```
db2 => grant use of tablespace usertempl to public
```

Connecting to a Remote DB2 Instance on Unix

When connecting to a remote instance of DB2 on Unix, the minimum requirement is an install of the db2 connect client. Once the client is installed, and a DB2 client instance has been configured, it is necessary to configure the connection to the remote DB2 instance.

First, log in as the DB2 instance owner, and execute the db2 command processor. For each remote DB2 server you wish to connect to, it is first necessary to register the remote server, using the following command:

```
db2 => catalog tcpip node <node name> remote <hostname> server <service>
```

where 'node name' is a name you wish to assign to this node (this will be used in later commands when defining the database connection), 'hostname' is the name of the host on which the remote DB2 instance resides, and 'service' is the TCP/IP service name/number (by default, 50000). For example

```
db2 => catalog tcpip node jester remote jester.mydomain.com server 50000
```

For each database you wish to reference, we must now issue a 'catalog database' command:

```
db2 => catalog database <database name> [ as <alias> ] at node <node name>  
authentication <authentication type>
```

Where 'database name' is the name of the DB2 database on the remote instance, 'alias' is the optional name by which you wish to reference this database, 'node name' is the node name we defined with the 'catalog tcpip' command, and 'authentication type' is the authentication you wish to use.

For example, to access the database 'stores' on the node 'jester' we defined, we would issue the command:

```
db2 => catalog database stores at node jester authentication server
```


CHAPTER 5

Preparing for the IBM DB2 Conversion

The final goal of the IBM DB2 conversion is to have a set of Informix 4GL sources that work equally with which IBM DB2 and Informix (and any other RDBMS required by the application vendor). While the Lycia dynamic SQL translator will make much of this process transparent to the developer, there are a number of vital steps to undertake to ensure optimal compatibility.

When converting an application to work with an additional RDBMS, a number of key steps need to be taken to ensure a successful result. It is strongly recommended that Querix tools be used for all stages of this process, in order to ensure that the application works consistently against each RDBMS with minimal code modification.

This chapter outlines the base process for converting an application to work with an additional RDBMS.

The Conversion Process

The migration process can be thought of as a four-stage process:

- Migration of the Database Schema
- Table loading
- SQL Syntax issues within 4GL code
- Runtime testing

The overall aim of the process should be to adapt an existing application to work with a new RDBMS in addition to still functioning as expected with the original RDBMS.

Migration of the Database Schema

Lycia will handle schema object conversions for you by means of a 4GL/ESQL-C application. Unlike the native migration tools for your target RDBMS, Lycia will allow you to retain basic Informix datatypes, which are not necessarily handled natively by the RDBMS. These datatypes can include:

- SERIAL
- MONEY
- DATETIME
- INTERVAL

Also included may be various behavioural facets associated with the datatypes in question. Please refer to the RDBMS vendor specific information in chapter 3 for information on datatype support.



Using Lycia tools to perform the database object creation will maximise compatibility of the target RDBMS with Informix behaviour, and minimise later conversion problems.

Converting the Database Schema

The simplest method of assisted database object creation is to convert your Informix database schema into a 4GL source. This can be achieved using the Querix utility 'qxexport'.

To invoke this utility from the command line, you simply need to perform:

```
bash$ qxexport <database_name>
```

This will create a directory named <database_name>_qxexport. Within this directory you will find a file named '<database_name>.4gl' and a subdirectory containing unload data for the database (unless the qxexport utility was invoked with the -t flag).

Having created the 4GL module, it is simply a matter of compiling this code and running it against the target RDBMS in order to create the database objects. For more information on connecting a 4GL application to your vendor's RDBMS, please see appendix 'Connecting To IBM DB2'

It is worth noting that if your target RDBMS does not support the use of reserved words as identifiers, you may encounter a number of table creation errors at this stage (see chapter 'Reserved Words as Identifiers' for information on the quoted identifiers database creation option in IBM DB2). If your RDBMS does not support the use of reserved words as identifiers, this can only be addressed by renaming the conflicting column(s), bearing in mind that these changes should be propagated into table references within the 4GL/ESQL-C code.



The qxexport utility is influenced by the LYCIA_DB_DRIVER environment variable. Ensure that LYCIA_DB_DRIVER is set to the correct RDBMS type before attempting to extract the database schema.

Loading Table Data

Loading of the table data should again be handled by 4GL - this ensures that the data is stored and converted correctly where appropriate. The schema extraction tool (qxexport) will automatically unload table data, and generate LOAD statements in the generated 4GL file for schema creation.

The 4GL LOAD statement can be resource intensive. For this reason, the overall schema creation/loading process may take time depending on the volume of data to be loaded. It may be advisable to perform a trial run without loading the table data to ensure that the schema objects can be created successfully.

Compiling 4GL Code

Once the schema objects have been created, you should now be able to proceed with the compilation of 4GL code against the target RDBMS. Recompiling the code against a new RDBMS is no different to compilation against Informix. Simply ensure that you are able to connect to the target RDBMS instance, and set the LYCIA_DB_DRIVER environment variable accordingly.

For example, from the command line:

```
bash$ 4make vclean
```

```
bash$ LYCIA_DB_DRIVER=db2 ; export LYCIA_DB_DRIVER
```

```
bash$ 4make myprogram
```

Application Testing

When working with an RDBMS for the first time, it is necessary to plan for sufficient application testing, especially if potential problems have been identified.

To assist with the testing process, the Lycia database interface provides SQL tracing facilities allowing the developer to view the SQL being issued from the 4GL, and the translated SQL being issued to the RDBMS. This can be especially helpful when trying to track unexpected behaviour in the target RDBMS.

To enable the SQL tracing facilities, it is first necessary to compile the application in debug mode, for example, from the command line:

```
bash$ 4make vclean
```

```
bash$ 4make -D myprogram
```

Once the application is compiled, SQL tracing can be enabled by setting the environment variable QXDEBUG. For general SQL tracing, the appropriate setting for QXDEBUG is 'UuBbSe', for example

```
bash$ QXDEBUG=UuBbSe ; export QXDEBUG
```

Running with SQL tracing enabled will cause the application to log interaction with the database to stderr. When running the program divert the output to a file for later inspection, for example:

```
bash$ ./myprogram 2>sqltrace.out
```

The qxexport Schema Extraction Tool

Querix provides a command line tool named 'qxexport' for the extraction of an Informix database schema and table data (the tool may also be used for extracting schemas from other RDBMSs). Please note that the qxexport tool will only extract the information required for transferring to another RDBMS vendor, and is not recommended as a backup/recovery tool.

The tool generates a LyciaStudio project, including a 4GL program (and table unload data) which can be compiled and run to automatically create and populate tables in the target RDBMS.

```
qxexport [ -fgl | -sql | -xml ] [ -s ] [ -o directory ] <database_name>
```

CHAPTER 6

Informix Compatibility

This chapter discusses scenarios within an Informix application which cannot be handled automatically by the Dynamic SQL Translator. It is broken into 3 sections marking the severity of the problem class. These sections cover the following:

- Problems that are not handled
- Problems that are handled through emulation of Informix behaviour
- Problems that are automatically handled, and are of no overall concern to the developer when working through the Dynamic SQL Translator

Unhandled Problems

Matches

The MATCHES keyword within Informix SQL is a non-ANSI extension to SQL supported only by Informix. The majority of RDBMSs do not have a direct equivalent to this operator, and as such, this should be addressed in advance.

The nearest equivalent (for most uses of MATCHES) is the LIKE operator. Unlike the MATCHES operator, LIKE cannot handle regular expressions, only wildcards.

- The MATCHES wildcard character '*' should be replaced with the LIKE wildcard character '%'.
Note: This replacement is not valid for regular expressions.
- The MATCHES wildcard character '?' should be replaced with the LIKE wildcard character '_'.
Note: This replacement is not valid for regular expressions.
- Care should be taken to escape any literal '%' or '_' in the expression being matched. '*' and '?' will no longer need to be escaped.

MATCHES Expression	LIKE Expression
WHERE table1.col MATCHES 'ABCD*'	WHERE table1.col LIKE 'ABCD%'
WHERE table1.col MATCHES 'ABCD?'	WHERE table1.col LIKE 'ABCD_'
WHERE table1.col MATCHES 'ABCD%*'	WHERE table1.col LIKE 'ABCD\%*'
WHERE table1.col MATCHES 'ABCD_\?'	WHERE table1.col LIKE 'ABCD_?'
WHERE table1.col MATCHES 'ABCD[0-9]*'	No equivalent.
WHERE table1.col MATCHES table2.col	Care should be taken to check the contents of the column being matched before making decisions regarding the conversion of this statement.

Unsupported SQL

The create/drop database statements can be extremely destructive in the context of IBM DB2. For this reason, the Hydra interface will not allow such statements to be executed.

Informix	Non-Informix RDBMS
Create/drop database	Not supported (options and arguments are engine dependent, so will not migrate). Lycia will not pass these statements to the RDBMS.
grant/revoke	Security models differ between Informix and other RDBMS vendors. It makes little sense to try to migrate a grant statement directly. Each instance of code that uses grant/revoke should be addressed individually

System Catalogues

The Informix system catalogues are not available in an IBM DB2 environment. The most commonly used of these are systables, sysuser and sysindexes. IBM DB2 does have equivalents to these catalogues, and as such it is possible to emulate these tables through a series of views.

Column Defaults

Unlike Informix, IBM DB2 does not allow functional defaults for table columns. For this reason, the following values cannot be used as column defaults under DB2:

- TODAY
- CURRENT
- USER

Problems Handled Through Emulation

The following issues are automatically handled via emulation of Informix behaviour. In most cases the emulation will be transparent to the developer/user. The relative problems are detailed with each item.

Create Temp Table (with NO LOG option)

Although temporary tables are fully emulated by Querix, you should be aware that the NO LOG option is disregarded at runtime.

Querix emulates the temporary tables using standard IBM DB2 tables (which are implicitly logged); the resultant behaviour is that of an unlogged table, since it will be implicitly dropped on program termination.

ROWID

SQL Server has no equivalent of ROWID; therefore ROWID emulation is provided.

The implementation method is table dependent. For tables with an existing SERIAL column, this column is returned as the ROWID value. For columns without a SERIAL column, a new hidden column ROWID is added to the table.

Setting the environment variable QXORA_EMULATE_ROWID to a value of 1 will enable the ROWID emulation feature. If ROWID emulation is not required the QXORA_EMULATE_ROWID environment variable should be unset.

When performing an INSERT of a literal value into a serial column, the ROWID for the insert is not correctly returned in `sqlca.sqlerrd[6]`.

ROWID emulation is available for SQL Server.



The ROWID emulation is unable to re-use ROWID values in an integer context. For this reason, it is feasible that an application with a high insert/delete rate may exceed the 2^{32} (approx. 4 billion) value limit. Use of this feature, therefore, should be treated with due consideration.

Problems Fully Handled

The following differences are automatically handled, and will present no problems for the developer/user. This is not an exhaustive list, and covers only the major differences.

Legal Parameter Context

IBM DB2 places a number of constraints for the use of parameter markers ('?') in SQL expressions. This includes:

- Parameters as function arguments
- Parameters as operands

For example, the following expression segment will cause an execution error:

```
WHERE YEAR ( ? ) = 2000
```

The Dynamic SQL Translator will automatically convert such expressions to perform a CAST on the parameter, thus creating legal SQL statements. For example

```
WHERE YEAR ( ? ) = 2000
```

Becomes

```
WHERE YEAR ( CAST ( ? AS DATE ) ) = 2000
```

Function/Operator Naming

The Dynamic SQL Translator will convert all standard Informix functions/operators to their IBM DB2 equivalents where an appropriate match exists. The following translations are performed automatically:

- YEAR()
- MONTH()
- DAY()
- DATE()
- MDY()
- WEEKDAY()
- LENGTH()
- TODAY

- CURRENT
- DATETIME / INTERVAL literals
- Character subscripts/substrings

Note that the expressions passed through to functions may also be modified internally depending on the context within the SQL expression.

DDL Syntax

There are numerous differences in the core DDL syntax between Informix and IBM DB2. This includes such things as:

- ALTER TABLE
- Constraint naming definitions

The dynamic SQL translator automatically converts all DDL from the Informix syntax to the IBM DB2 syntax.

Literal Type Mapping

IBM DB2 does not allow the same level of weak typing as Informix, and as such, literal values are automatically converted to the type expected in the expression. For example, IBM DB2 will not natively allow integer constants in a DATE context.

```
SELECT ...
```

```
WHERE date_column = 32514
```

Is translated as

```
SELECT ...
```

```
WHERE date_column = ?
```

```
-- the INTEGER constant is converted to a DATE parameter
```

Join Syntax

IBM DB2 does not support the Informix Join syntax, instead using the ANSI join syntax for SQL. The Dynamic SQL Translator automatically handles the conversion for all join expressions.

SQL Error Codes

4GL applications are largely dependent upon expected error codes from Informix. IBM DB2 reports errors using a series of native error codes.

For this reason, the Dynamic SQL Translator will convert SQLSTATE values to the Informix equivalent error, and populate the `sqlca.sqlcode` field accordingly.

Should the RDBMS return an error code which does not correspond to a known Informix error code, the database interface will return `sqlca.sqlcode` with a value of -999, and the native (ISAM) error stored in `sqlca.sqlerrd[2]`.

Configurable Behaviour

The following options can be set in the fglprofile to configure the behaviour of the IBM DB2 database interface.

`dbi.db2.rowid = (emulate | native)`

This option determines whether or not ROWID emulation is enabled. This option defaults to 'native' (emulation disabled).

APPENDIX A

The Lycia Dynamic SQL Translator

This chapter discusses the dynamic SQL translation engine, and the work that it performs. This chapter is intended to provide an overview of the capability of the translator.

Although the principles of operation of the translator are common across all RDBMS vendors, the functionality of the translator is sensitive to capabilities and personality of the target RDBMS.


Introduction

The basic function of the Dynamic SQL Translator (DST) is to make a non-Informix database appear as close as possible to an Informix database for the purposes of 4GL and ESQL/C functionality.

The translator takes the users' Informix SQL statements as input, and dynamically translates them to the format required by the target RDBMS.

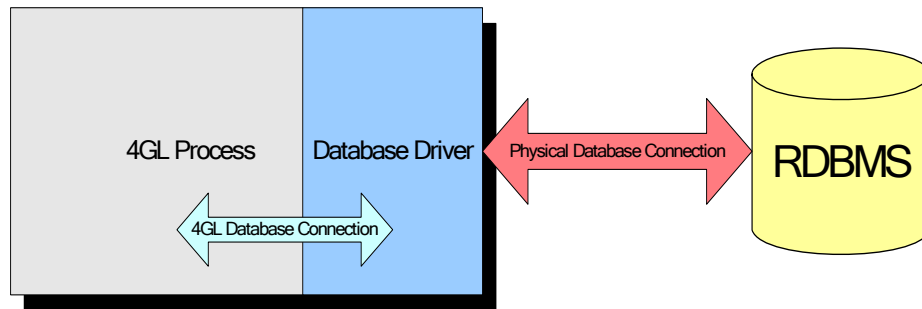
In addition to this, the DST also emulates the behaviour of Informix datatypes, such as SERIAL, MONEY, INTERVAL etc., even where no direct equivalent exists within the target RDBMS.

The overall result is that 4GL and ESQL/C programs can be used with a wide range of RDBMSs with the absolute minimum of code modification.

	<p>The concepts described in this chapter apply only to non-Informix databases. The Dynamic SQL Translator is not used for Informix connections.</p>
---	--

Concepts

Database Drivers



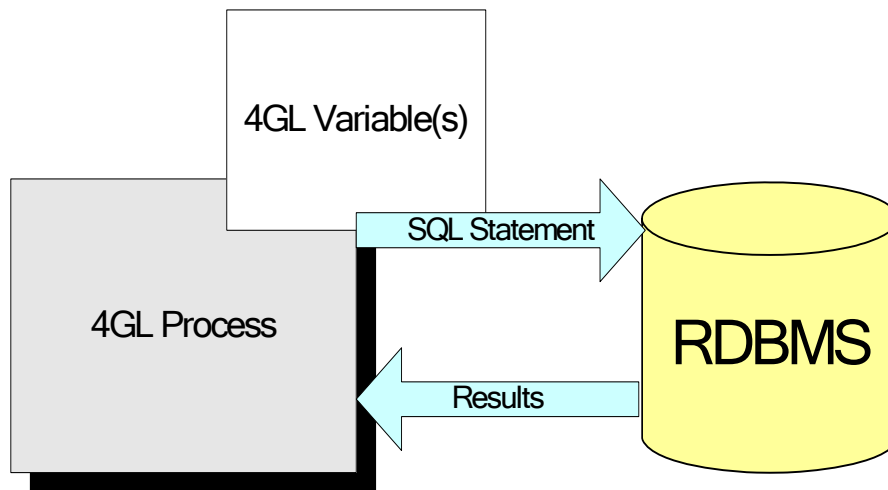
Each database vendor provides a different means of connecting to and communicating with a database. For this reason, it is not possible to use the same communication mechanism with, for example, IBM DB2 as you would with Informix.

Each database, therefore, requires a different layer for communication. Lycia provides such a layer for each database vendor supported, and these layers are referred to as 'Database Drivers'.

Multiple connections to the same database type will all use the same driver.

Bind Variables

A bind variable is a 4GL or ESQL-C variable which is passed as a parameter to a SQL statement.



For example, consider the following 4GL statement:

```
DEFINE my_int INTEGER
```

```
LET my_int = 1
SELECT *
  FROM my_table
  WHERE my_table.column1 = my_int
```

In this example, the 4GL variable 'my_int' is passed as a parameter (or input value) for the SQL select statement. Similarly, the values that are passed to a prepared statement to substitute the placeholder markers ('?') are also bind variables. For example:

```
DEFINE my_int INTEGER
...
PREPARE p1 FROM
  "SELECT * FROM x1 WHERE x1.a = ? AND x1.b = ?"
EXECUTE p1 USING 1, my_int
```

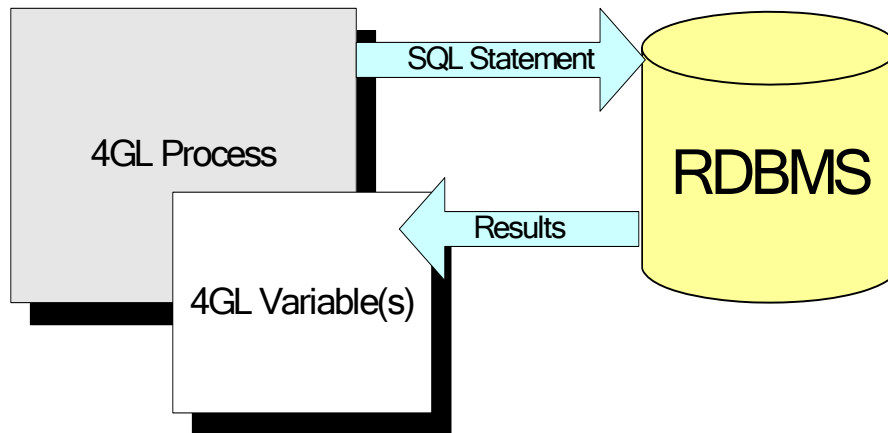
In this example, both the literal value '1' and the 4GL variable 'my_int' are considered bind variables for the SQL statement.

SQL statements that can use bind variables are:

- DELETE
- EXECUTE PROCEDURE
- INSERT
- SELECT
- UPDATE

Buffer Variables

We use the term 'buffer variables' to describe the variables into which an SQL statement returns data. Such variables will typically be found in the 'INTO' clause of a SQL statement.



For example, consider the following 4GL fragment:

```

DEFINE rec1 RECORD LIKE x1.*

DECLARE c1 CURSOR FOR
  SELECT *
  FROM x1

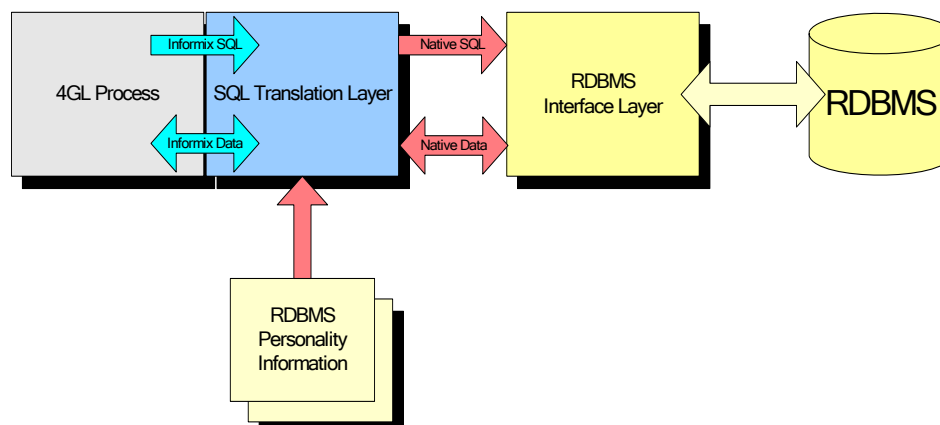
FOREACH c1 INTO rec1.*
  ...
  
```

In this example, each element of the record 'rec1' is a buffer variable, as they are modified with each execution of the FOREACH loop.

SQL Statements that can use buffer variables are:

- EXECUTE PROCEDURE
- SELECT

How Dynamic Translation Works




The translation engine has a number of layers of functionality, and these can be categorised as follows:

- SQL Interpretation
- SQL generation
- Type promotion

SQL interpretation

The first function of the translation layer is to read and understand the SQL statement passed to it. During this process, the translator will evaluate the entities present within an SQL statement.

Once interpreted, the statement is categorised to allow optimal processing of the SQL statement.

	If the translation engine does not fully recognise the SQL statement, it will not attempt to process it. Instead, it will pass the statement to the RDBMS unmodified.
---	---

SQL Generation

The final function of the Dynamic SQL Translator is to recreate the SQL statement in a form that is understood by the target RDBMS. There are numerous aspects to this, but the key areas are:

- Keywords: Many SQL keywords are vendor specific
- Grammatical differences in SQL: Each RDBMS exhibits minor differences in the grammatical structure of SQL
- Functions and Operators: The names and syntax of functions and operators varies between RDBMS vendors.

- Datatypes: Some Informix datatypes are not supported by other RDBMS vendors.

Datatype Mappings

In addition to type promotion in SQL (this will be discussed in the next section), the Dynamic SQL Translator is sensitive to differences in the data types supported by different RDBMS vendors. For this reason, the dynamic SQL translator automatically handles the translation of Informix types to vendor specific types, in such a way that these differences are transparent to a 4GL or ESQL/C process.

The qx__\$schema table

Because interpretation has to be performed for some datatypes, the translator will maintain metadata about such columns in the table qx__\$schema. The qx__\$schema table contains such information as the original (or intended) Informix datatype, and any supplementary information (such as qualifiers for DATETIME types).

Type Promotion

Unlike Informix, a number of RDBMSs are strongly typed. This means that they do not permit comparisons of differing datatypes. For example, where Informix may allow the user to directly compare a CHAR representing a date to a DATE value, IBM DB2 will not.

The SQL translator scans the SQL statement for areas where differing datatypes are being used, and attempts to convert the types where possible.

As an example, consider the following scenario:

```
SELECT *
FROM x1
WHERE x1.date_column = "01/01/2001"
```

In this example, the DST will first look at the relational operator '=':

In evaluating this operator, the translator realises that potential problems exist due to a CHAR value being compared to a DATE column. Problems that potentially exist include:

- A strongly typed RDBMS will not allow such a comparison without an explicit cast operation.
- Localisation of the database server may prevent the literal date value from being correctly interpreted.
- The RDBMS may not be able to efficiently cache semantically equivalent statements that differ only by literal value.

Since it is clear that the column cannot be altered, it is necessary to process the literal value instead.

In this case, the translator will first attempt to convert the literal value to a bind value of type CHAR. In this case, the SQL will now be seen as:

```
SELECT *
FROM x1
```

```
WHERE x1.date_column = ?
```

Finally, due to the context of the CHAR bind, the CHAR bind will be converted to a bind of type DATE, preventing the need for any explicit casting operation.

The result of this process is an SQL statement that will be accepted by a strongly typed database, and also provides more efficient SQL than if the vendor's conversion functions (such as CAST or CONVERT) had been used.

Benefits of Type Promotion

There are a number of beneficial side-effects of the type promotion process. Firstly, the database is able to cache SQL statements better if literal values aren't used. For example, consider the following statements:

```
SELECT * FROM x1 WHERE x1.a = 1  
SELECT * FROM x1 WHERE x1.a = 2  
SELECT * FROM x1 WHERE x1.a = 3
```

The RDBMS will attempt to cache each of these statements. This caching is made more effective by the SQL translator, as in each of these cases, the only statement passed to the RDBMS is:

```
SELECT * FROM x1 WHERE x1.a = ?
```

As a result, the database is able to cache SQL statements much more efficiently.

Areas where Type Promotion is used

Type promotion will be attempted for all parts of SQL where datatypes won't necessarily match. This includes:

Parameters (bind variables) to INSERT statements

- Parameters (bind variables) to UPDATE statements
- WHERE clauses
- Output parameters (buffer variables) of SELECT statements

Where bind/buffer variables are type promoted, the promotion is applied to a duplicate variable, with the original 4GL variable remaining unaltered.

APPENDIX B

Datatype Mappings

The following table lists the Informix SQL/4GL datatypes, and the type mapping adopted by the Dynamic SQL Translator. All Informix datatypes are supported under IBM DB2 through the Dynamic SQL Translator. If any behavioural emulation is required, it is stated within the notes for that datatype.

Informix SQL Datatype	Informix 4GL Datatype	IBM DB2 Datatype	Notes
CHAR	CHAR	CHAR/VARCHAR	The DB2 CHAR type is limited to a maximum length of 255. Attempts to create a column larger than 255 will automatically handled using VARCHAR
VARCHAR	VARCHAR	VARCHAR	
LVARCHAR	VARCHAR	VARCHAR	
NCHAR	NCHAR	CHAR	
NVARCHAR	NVARCHAR	VARCHAR	
SMALLINT	SMALLINT	SMALLINT	
INTEGER	INTEGER	INTEGER	
INTEGER8	BIGINT	INTEGER	DB2 doesn't currently support 64-bit integer types
SERIAL	INTEGER	INTEGER	
SERIAL8	BIGINT	INTEGER	DB2 doesn't currently support 64-bit integer types
FLOAT	FLOAT	FLOAT	
SMALLFLOAT	SMALLFLOAT	REAL	
DOUBLE PRECISION	FLOAT	FLOAT	
DECIMAL(p, s)	DECIMAL(p, s)	DECIMAL(p, s)	DB2 Decimal datatypes are limited to a maximum precision of 31

Informix SQL Datatype	Informix 4GL Datatype	IBM DB2 Datatype	Notes
MONEY(p, s)	MONEY(p, s)	DECIMAL(p, s)	DB2 Decimal datatypes are limited to a maximum precision of 31
DATE	DATE	DATE	
DATETIME	DATETIME	TIMESTAMP	
INTERVAL	INTERVAL	NUMBER	
BYTE	BYTE	BLOB	
TEXT	TEXT	CLOB	

The following table lists the mappings adopted by the Dynamic SQL Translator when encountering a datatype within the IBM DB2 database.

IBM DB2	Informix 4GL	Informix SQL	Notes
BLOB	BYTE	BYTE	
CHAR	CHAR	CHAR	
CLOB	TEXT	TEXT	
DATE	DATE	DATE	
DBCLOB	TEXT	TEXT	
DECIMAL	DECIMAL	DECIMAL	
DOUBLE	FLOAT	FLOAT	
GRAPHIC	BYTE	BYTE	
INTEGER	INTEGER	INTEGER	
REAL	SMALLFLOAT	SMALLFLOAT	
SMALLINT	SMALLINT	SMALLINT	
TIME	DATETIME	DATETIME	
TIMESTAMP	DATETIME	DATETIME	
VARCHAR	VARCHAR	VARCHAR	
VARGRAPHIC	BYTE	BYTE	

APPENDIX C

Common Conversion Issues when migrating to a different Vendor's Database

Prior to recompiling the 4GL, a number of operational issues need to be addressed in the code. This is due to a number of fundamental operational differences between Informix and the target RDBMS. It is important to address these issues in advance, as the consequences of one issue can have cascading effects.

System Catalogues

The Informix system catalogues are generally not directly available in the target RDBMSs. Dependent upon the RDBMS in question, system catalogues can be either emulated as a series of views, or references to the catalogue in question automatically converted to the native equivalent. In the majority of cases, it is best not to be dependent upon the system catalogues behaving as you would expect under Informix.


Lycia provides a set of API functions for accessing the native system catalogues. It is recommended that these functions be used in place of direct queries against the table, for two key reasons:

- The true Informix type and characteristics of the column is returned.
- All work concerning the variances in catalogues are automatically handled.

Matches

The MATCHES keyword within Informix SQL is a non-ANSI extension to SQL. Whilst PostgreSQL has a direct equivalent (the '~' and '!~' operators), the majority of RDBMSs do not have a direct equivalent to this operator, and as such, this should be addressed in advance.

The nearest equivalent (for most uses of MATCHES) is the LIKE operator. Unlike the MATCHES operator, LIKE cannot handle regular expressions, only wildcards.

	<p>The MATCHES operator only poses a problem within SQL statements. The operator will pose no problem within general 4GL program logic.</p>
---	---

Owing to the problems associated with the MATCHES operator, wildcard symbols ('*' & '?') entered during a CONSTRUCT statement will automatically converted to LIKE equivalents. Under such circumstances a LIKE operator is generated in place of MATCHES in the generated SQL clauses.

Cursor Names

Due to common restrictions within various RDBMS systems, ALL cursor names are subject to an 18-character length limit. By default, the 4GL compiler will hash all cursor names into an 18 character string. If you disable cursor hashing within the 4GL compiler (by passing the flag '-CH' to fglc), then any cursor whose name is over 18 characters in length will need to be renamed.

Isolation Levels

Isolation levels are RDBMS specific models for handling concurrent transactions. These operations are handled at the server level, and as such, no guarantee can be provided for identical behaviour in similarly named isolation levels between differing RDBMS vendors.

Most RDBMSs will only allow Isolation Levels to be switched between transactions.

Stored Procedures

Stored procedures must be recoded according to the facilities available in the target RDBMS. Whilst we can provide general guidelines for procedure coding, there are no formal procedures for the conversion of SPL

Index

Authentication	7	NCHAR.....	37
Bind Variables	31	NVARCHAR	37
Buffer Variables	33	Preparing	15
Common Conversion	41	Prerequisites	5
compared	7	Problems Fully Handled.....	26
Comparison.....	7	Problems Handled Through Emulation	25
Compiling 4GL code	17	qx__\$schema table	35
Connecting.....	11, 16	qxexport	16
Conversion.....	15	Read Committed (ANSI).....	8
Cursor Names	42	Repeatable Read.....	8
Cursor Stability	8	Reserved Words as Identifiers.....	23
Database Drivers.....	31	schema object	15
Datatype Mappings	35, 37	SERIAL8.....	37
DATETIME.....	38	Serializable (ANSI).....	8
Dirty Read	8	SQL Compliance	7
Dynamic SQL Translator.....	30	SQL Error Codes.....	27
Informix Compatibility	19	SQL Generation	34
Informix datatypes	15	SQL interpretation.....	34
INTEGER8	37	Storage	7
INTERVAL	38	Stored Procedures	42
Isolation Levels.....	7, 8, 42	Supported Connection Interfaces	7
Loading table data	16	System Catalogs	41
Matches.....	41	Translation	34
Migration of the Database Schema	15	Type Promotion.....	35
MONEY	37		