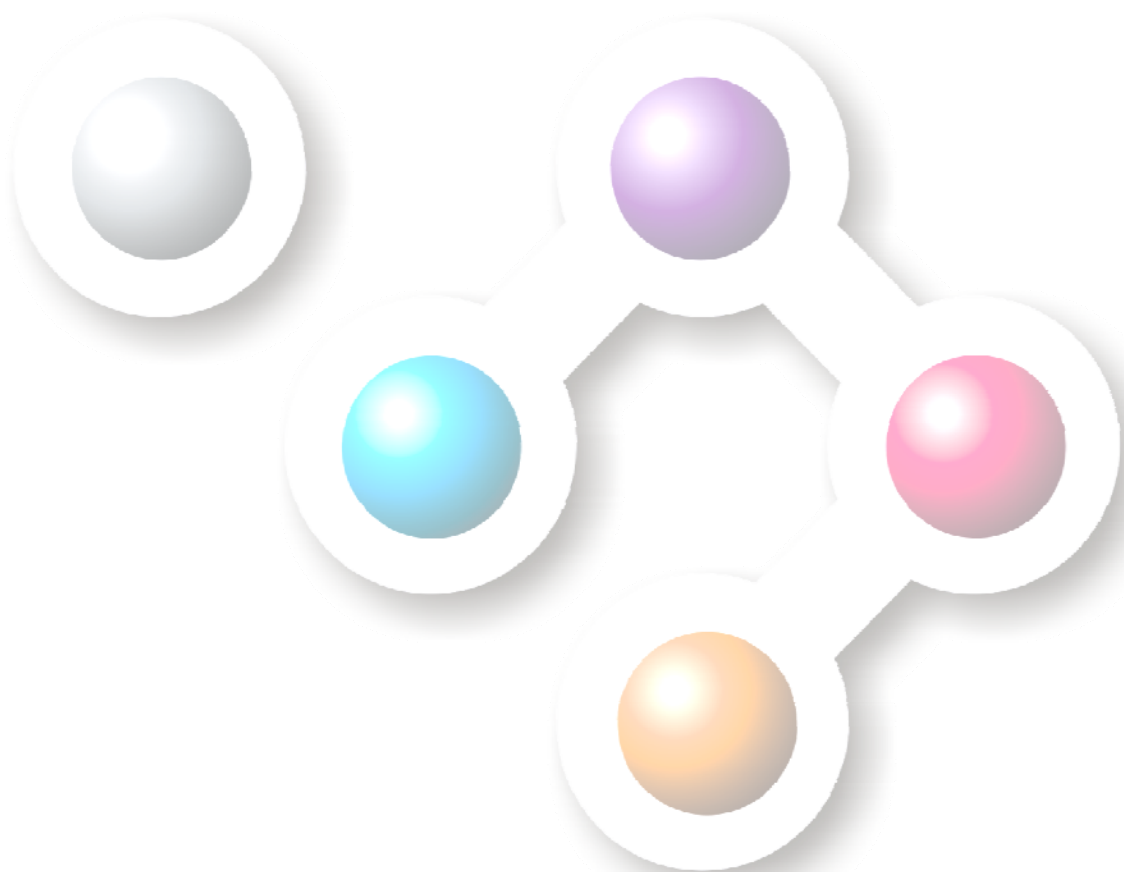


Lycia Development Suite



**Lycia Database Migration Guide for Oracle
Versions 5 and 6 – February 2011**



Dynamic Database Interface

Migration Guide for Oracle

Versions 5 and 6

January 2011

Part Number: 015-006-135-011

Querix Dynamic Database Guide for Oracle

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

January 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 Oracle, updated for versions 4.3 and retitled 'Database Migration Guide for Oracle'
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.

CHAPTER 1	3
INTRODUCTION	3
CHAPTER 2	5
PREREQUISITES FOR CONVERTING TO ORACLE.....	5
CHAPTER 3	7
INFORMIX AND ORACLE COMPARED	7
<i>General Comparison</i>	7
<i>Isolation Levels in Informix</i>	8
<i>Isolation Levels in Oracle</i>	8
LYCIA DATABASE CONNECTIONS.....	9
Environment Settings.....	9
Concepts	9
Common Errors	9
CHAPTER 4	11
CONNECTING TO ORACLE	11
<i>Connecting to Oracle</i>	11
Environment Settings.....	11
Concepts	11
Creating an Oracle User	11
Informix Database and Oracle User Mappings	12
Example of database translation	13
Secure Mode	13
Migrating System Authentication	14
Explicit Connections.....	14
CHAPTER 5	15
PREPARING FOR THE ORACLE CONVERSION	15
The Conversion Process	15
Migration of the Database Schema	15
Converting the Database Schema	16
Loading Table Data.....	16
Compiling 4GL Code	17
Application Testing.....	17
The qxexport Schema Extraction Tool	18
CHAPTER 6	19
INFORMIX COMPATIBILITY	19
<i>Unhandled Problems</i>	20
Matches.....	20
Unsupported SQL	21

Restrictions on SQL Union & Outer Join Expressions.....	21
System Catalogues.....	22
Transactions.....	22
Table locking.....	22
DDL Statements.....	22
FOR UPDATE statements	22
Table Locking & Isolation Levels	23
Reserved Words as Identifiers	23
DDL operations on Temporary tables	23
SPL Objects	24
<i>Problems Handled Through Emulation.....</i>	<i>25</i>
WHERE CURRENT OF	25
ROWID	25
Create Temp Table (with NO LOG option)	25
USER Keyword	25
<i>Problems Fully Handled.....</i>	<i>26</i>
Function/Operator Naming	26
DDL Syntax	26
Literal Type Mapping.....	27
Join Syntax.....	27
SQL Error Codes	27
<i>Configurable Behaviour.....</i>	<i>29</i>
APPENDIX A	30
THE 'LYCIA' DYNAMIC SQL TRANSLATOR	30
<i>Introduction.....</i>	<i>30</i>
<i>Concepts</i>	<i>31</i>
Database Drivers	31
Bind Variables	31
Buffer Variables	33
<i>How Dynamic Translation Works.....</i>	<i>34</i>
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 Oracle.

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.

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

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 Oracle.

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 Oracle

In order to use 'Lycia' with Oracle, you must have the following:

- Oracle version 8i or later installation (see <http://www.Oracle.com/> 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 Oracle Compared

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

General Comparison

Subject	Informix	Oracle
Concepts	The server is a container unit for multiple databases.	Semantically, a database is synonymous with an instance. Each database contains multiple users, with each user being synonymous with an Informix database.
Storage	Informix Standard Engine uses a directory structure for physical data storage. 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.	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	Proprietary, ADO, OLEDB, ODBC, JDBC
User Authentication	Operating System Managed	RDBMS managed, Operating system managed
Supported Isolation Levels	Dirty Read Committed Read Cursor Stability Repeatable Read	Serializable Read Committed

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 Oracle

Read Committed	Default Oracle transaction behaviour.
Serializable	DML within the transaction which would cause conflict with a concurrent transaction will fail.

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.
- Oracle – Connect to Oracle.
- Sserver – Connect to Microsoft SQL Server.
- Db2 – Connect to 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 Oracle

Connecting to Oracle

Environment Settings

- ORACLE_HOME – base installation directory of Oracle.
- ORACLE_SID – The default server instance for this connection.
- QXTRANS – The location of the dbtrans file.
- DBTEMP – Directory to use for storing temporary files.


Concepts

Oracle has a largely different concept of user/database to Informix. Under a typical Informix database, a user is able to access objects under a number of different schemas (databases).

This differs in Oracle, in so much as an Oracle user should be treated as a database in its own right. Users, by default, do not have access to objects under another user's tablespace. Given this, the following mappings should be observed:

Informix	Oracle
create database	create user
grant/revoke	N/A in this context

For this reason, 'Lycia' assumes a 1:1 mapping between an Informix database name, and an Oracle user when connecting to Oracle.

	<p>Note: The create database statement under Oracle can have destructive consequences. It does not have the same implications as the identical Informix statement.</p>
---	---

Creating an Oracle User

Installation and configuration of the Oracle RDBMS is beyond the scope of this document. As such it will be assumed that this process has been completed.

To create a user under Oracle, run the following command, and connect as a user with DBA privileges:

sqlplus

Basic parameters are to provide the username and password, and to grant resource and connect privileges. These privileges are sufficient for most purposes, and it is not usually advisable to grant full DBA privileges to an Oracle user.

As an example, administering the following commands in SQL*Plus could be used to create a user with the name querix and the password hello:

```
sqlplus> create user querix identified by querix default tablespace users;  
sqlplus> grant resource to querix;  
sqlplus> grant connect to querix;
```



Note: Unless otherwise specified, user objects will be created in the system tablespace.

For additional user options please refer to your Oracle documentation.

Informix Database and Oracle User Mappings

When connecting to Oracle, the 'Lycia' driver will attempt to translate the Informix database name given in the DATABASE or CONNECT statement to an Oracle user to use for the connection.

For example, your 4GL code may contain a database statement such as the following:

```
DATABASE stores
```

This database name 'stores' needs to be translated (or mapped) to an Oracle user logon string, such as 'scott/tiger' or 'scott/tiger@ora1'. 'Lycia' provides a means of specifying possible mappings in the file 'dbtrans' (located in either /etc or \$QUERIXDIR/etc). These files consist of five fields (separated by spaces or tabs) as follows:

Column	Description
Informix Database	The <i>Informix</i> name of the database.
Username	Username of the user running the Querix-4GL program, this can also be the username of the effective UID of the program.
Group Name	Group of the users running the Querix-4GL program, this can also be the effective group of the program.
Translation	The Oracle "connect" string is used for connecting to a database in the form USER/PASSWORD, or USER/PASSWORD@ORACLE_NET_SERVICE_NAME.
Comments	The remainder of the line can be used to contain comments.

The rows in the dbtrans file are subject to the following rules:

- The symbol "*" in the username, group name or database fields match all possibilities.
- If the user name or group name is preceded by a minus sign, then the field will match the effective user/group id of the running program, instead of the real user/group id (This feature is not available in secure mode, see next section).
- The first entry matching all the specified fields is returned.
- An Oracle user/password string of <null> results in automatic connection failure and returns Informix error code: -406.
- If no entries match the specified database name and the program is not in secure mode, a connection is attempted without any name conversion (although this will most likely fail). If the program is in secure mode and no matches are detected, the connection will fail and returns Informix error code: -406.
- Lines commencing with a hash sign (#) are considered comments, and are therefore ignored.

Furthermore, in the event of there being no matches for the specified database name, the name is passed verbatim to the Oracle database as a connect string. This is true even if the database name would otherwise be a valid oracle username/password combination. This rule is negated in secure mode.

Example of database translation

```
# database user  group translation
# mydb from any user connects to
# myuser/mypass@mydbserver
mydb      *      *      myuser/mypass@mydbserver

# Fred will try to attach to the default database
# as user fred and password fred if he uses testdb
testdb    fred  *      fred/fred

# All other users in the group develop,
# will get database dev/dev
testdb    *      develop    dev/dev

# This rule will never match. Since john is a member of
# group develop, the previous rule will match and apply.
testdb    john  develop    john/john

# All other connection attempts will be denied


*      *      *      <null>
```

Secure Mode

The dbtrans file places oracle user and password information in plain text in a freely readable form. Although in development environments this can be acceptable, in a deployment environment it is much

less desirable. 'Lycia', therefore, provides the facility of a secure mode. If the file /etc/dbtrans exists and is not readable by the executing program, secure mode is invoked.

In secure mode the program uses the external setuid program \$QUERIXDIR/bin/lookup to attempt to determine the correct connection information.

	<p>By default this program is setuid to the Querix software owner, so you should ensure that the software owner has authority to access the file /etc/dbtrans.</p>
---	--

In secure mode, only the file /etc/dbtrans is ever examined for database name translations. If, in secure mode, no match is found for the name given, the connection will be refused

Migrating System Authentication

Oracle 7.x provided the facilities to use operating system authentication in the database connection. This is still supported as a legacy connection method in Oracle 9i.

The key difference in O/S authentication is that the user/password is not supplied on connection; instead the oracle user to which you connect is based on the effective user id of the connecting process.

```
sqlplus> create user querix identified externally;
sqlplus> grant resource to querix;
sqlplus> grant connect to querix;
```

When a user is created in this manner, you need simply supply a connection string of "/" within either your dbtrans file, or 4GL DATABASE statement. For example:

```
DATABASE "/"
```

Or, in the dbtrans file:

```
# All databases, all users will use O/S authentication.
*      *      *      /
```

Explicit Connections

As well as being able to specify the logical mapping of Informix database name to an Oracle connection, 'Lycia' will also allow you to specify explicit Oracle database connections within 4GL/ ESQL/C. You can provide Oracle connection strings directly as an argument to the 4GL DATABASE or CONNECT statement. For example:

```
DATABASE "scott/tiger@server1"
```

Or

```
LET my_db = scott/tiger@server1
DATABASE my_db
```

CHAPTER 5

Preparing for the Oracle Conversion

The final goal of the Oracle conversion is to have a set of Informix 4GL sources that work equally with which Oracle 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 Oracle'

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 Oracle). 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=oracle ; 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 '%'.
- The MATCHES wildcard character '?' should be replaced with the LIKE wildcard character '_'.
- 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 Oracle. For this reason, the Lycia 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

Restrictions on SQL Union & Outer Join Expressions

Informix tends to allow more freedom in the use of outer joins within SQL syntax than Oracle. Oracle places a number of key restrictions on the use of outer joins, to the extent that:

A table can be, at most, outer joined to one table.

An OUTER table cannot be joined to a sub-query.

SELECT ...

FROM table1, OUTER table2

WHERE table2.a IN (SELECT ... FROM table 3)

The use of the OR operator within the WHERE clause within an outer join expression will result in error. This also applies to Boolean equivalencies of the OR operator.

SELECT ...

FROM table1, OUTER table2

WHERE (table2.a = 1 OR table2.a = 2)

SELECT ...

FROM table1, OUTER table2

WHERE NOT (table2.a <> 1 AND table2.a <> 2)

Although Querix will automatically convert the syntax of outer join and union expressions to the Oracle equivalent syntax, you may still encounter problems with OUTER JOIN expressions for the various restrictions in place. For this reason we recommend that special attention is paid to such SQL expressions.

System Catalogues

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

There are fundamental restrictions with this, however, due to numerous conceptual differences between Informix and Oracle. Principally, the main difference is that the user is constant in an Oracle environment. Therefore, statements such as

```
SELECT USER from systables WHERE tabid = 1
```

will consistently return the same value. As a result, all such references should be addressed.

Transactions

Oracle is implicitly transactional - this means that all statements issued outside of a transaction carry an implicit commit. This can have consequences on a number of fundamental operations:

Table locking

For example, In Informix, a table is locked by issuing a LOCK TABLE statement, and subsequently unlocked by issuing an UNLOCK TABLE statement. Under Oracle, there is no UNLOCK TABLE statement, since the table is automatically unlocked at the close of a transaction (i.e. when a COMMIT/ROLLBACK WORK statement is issued). This has the implication that, unless the table is locked within an explicit transaction (i.e., after an unclosed BEGIN WORK statement), the table will become unlocked at the end of the implicit transaction, which is the LOCK TABLE statement itself.

Therefore, unless a table is locked within an explicit transaction, the LOCK TABLE statement is ineffective.

DDL Statements

In addition to this, all DDL statements carry an implicit transaction commit. This includes statements such as:

- CREATE [TEMP] TABLE
- DROP TABLE

DDL Statements will force a transaction commit, even where the process is in transaction explicitly opened by the process.

FOR UPDATE statements

Using a FOR UPDATE cursor outside of an explicit transaction will consistently cause a 'fetch out of sequence' error due to the cursor being implicitly invalid. For example, the following:

```
DECLARE mycurs CURSOR FOR  
SELECT *
```

```
FROM mytab
FOR UPDATE
```

Will cause an error if the cursor is opened/fetched outside a transaction (this can mean issuing the statement following a COMMIT WORK statement). For this reason, it is important to ensure that all update cursors are completely contained within a transaction, as closing a transaction will implicitly close the cursor, and using the cursor before a transaction has been opened will result in this error.

Table Locking & Isolation Levels

Isolation levels are very much database level operations, and there are no direct mappings between Informix and Oracle.

The nearest equivalent Oracle statement is the SET TRANSACTION statement, which will only affect the users' current transaction – it will not affect other users or other transactions. This also has major implications on table locking, since Oracle employs a significantly different table-locking model to Informix

For example, locking a table in exclusive mode under Informix will prevent other users from either altering or viewing the table data, whereas Oracle places a read-lock a table.

Oracle offers the following table locking modes:

- ROW SHARE/SHARE UPDATE - allows concurrent access to the current table. This mode prevents other users from locking the table in exclusive mode.
- ROW EXCLUSIVE/SHARE ROW EXCLUSIVE has a similar effect, but also prevents the table being locked in SHARE mode.

A final point to note about table locking is that tables are locked up until the close of a transaction.

Reserved Words as Identifiers

Oracle will not allow reserved words to be used as identifier names. For this reason, all instances of Oracle reserved words being used as identifier names will need to be addressed, and the name changes reflected within the 4GL source code.

DDL operations on Temporary tables

By default, the dynamic SQL translator will map Informix temporary tables to Oracle global temporary tables. Due to limitations of the Oracle implementation of temporary tables, it is not possible to execute DDL statements on temporary tables that hold data. For example:

```
SELECT mytable.* INTO TEMP my_temp
CREATE INDEX i1 ON my_temp (column1)
```

The 'CREATE INDEX' statement in this case will generate an Oracle error 14452 "attempt to create, alter or drop an index on temporary table already in use".

This issue can be prevented by creating the index before populating the table, or by enabling emulation of temporary tables (emulated temporary tables are enabled by setting 'dbi.oracle.temp = emulate' in the fglprofile).

SPL Objects

Objects written using SPL (such as stored procedures, triggers and functions) cannot be automatically converted to the stored procedure language used by the target RDBMS. For this reason, all such objects must be recreated by hand.

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.

WHERE CURRENT OF

Oracle does not have an equivalent of the Informix WHERE CURRENT OF statement. This behaviour will be emulated automatically if ROWID emulation is enabled. If ROWID emulation is not to be enabled, then all instances of this statement must be re-written.

ROWID

Under Oracle, the ROWID is a binary representation of the physical address of the row – this is represented within 4GL as a VARCHAR(18).

This presents a number of issues in the conversion process. Firstly, `sqlca.sqlerrd[6]` cannot be used, since this will consistently be set to 0 (`sqlca.sqlerrd` is an array of integers, and as such is unable to store an Oracle ROWID value). In addition, all assignments of ROWID into 4GL variables should be done using CHAR(18) data types.

Querix does provide facilities for Informix ROWID emulation, allowing the user to retain integer definitions of ROWID.

Please refer to the section on 'Configurable Behaviour' for more information regarding this.

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 Oracle tables (which are implicitly logged); the resultant behaviour is that of an unlogged table, since it will be implicitly dropped on program termination.

USER Keyword

When connecting to Oracle, the Informix database name generally maps to an Oracle user. For this reason, the USER keyword will consistently return the same value, no matter which user ID the application is running as. It is strongly recommended to substitute all uses of the USER keyword in SQL with the use of the built-in 4GL function `fgl_username()`.

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.

Function/Operator Naming

The Dynamic SQL Translator will convert all standard Informix functions/operators to their Oracle 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 Oracle. This includes such things as:

- ALTER TABLE
- Constraint naming definitions

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

Literal Type Mapping

Oracle 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, Oracle 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

Oracle 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. Oracle 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 Oracle database interface.

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

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

`dbi.oracle.temp = (emulate | native)`

This option determines whether or not the interface will use Oracle temporary tables (available in Oracle 8i or later) or emulates temporary tables. This option defaults to 'native' (using Oracle temporary tables).

`dbi.oracle.long_binding = (true | false)`

This option, when set to true, will cause the runtime to bind all BYTE/TEXT values as Oracle version 7 'LONG' datatypes. The runtime defaults to binding BYTE/TEXT values as Oracle CLOB/BLOB datatypes.

`dbi.oracle.like.default_escape = (true | false)`

This option, when set to true, will automatically add an ESCAPE clause to LIKE statements where no explicit escape character has been set. This option defaults to 'true'.

`dbi.oracle.no_null_order = (true | false)`

This option, when set to true, will automatically append the NULLS FIRST option to ORDER BY statements. By default, Oracle will place NULL values last when ordering an attribute, whereas Informix will place NULL values first. This option defaults to 'true'.

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.

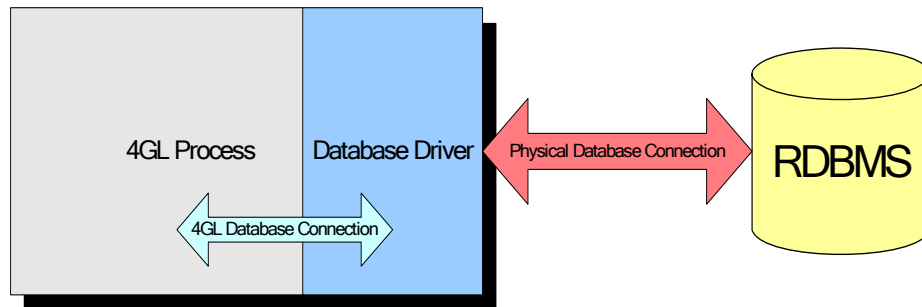
In addition to this, the DST also emulates the behaviour of Informix datatypes, such as SERIAL, MONEY, INTERVAL etc., even where no direct equivalent exists within the target RDBMS.

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



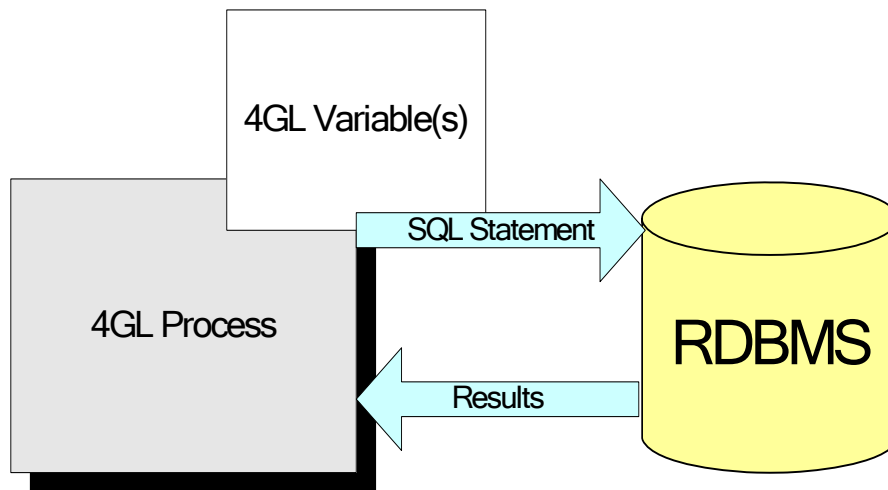
Each database vendor provides a different means of connecting to and communicating with a database. For this reason, it is not possible to use the same communication mechanism with, for example, Oracle as you would with Informix.

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
```

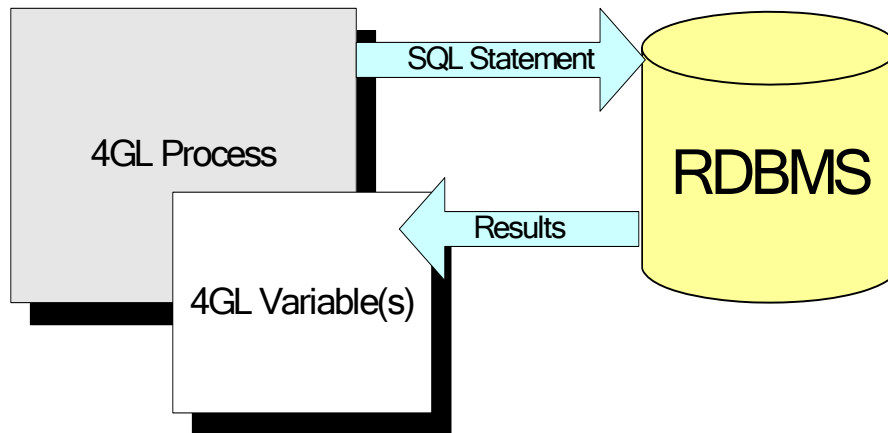
In this example, both the literal value '1' and the 4GL variable 'my_int' are considered bind variables for the SQL statement.

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

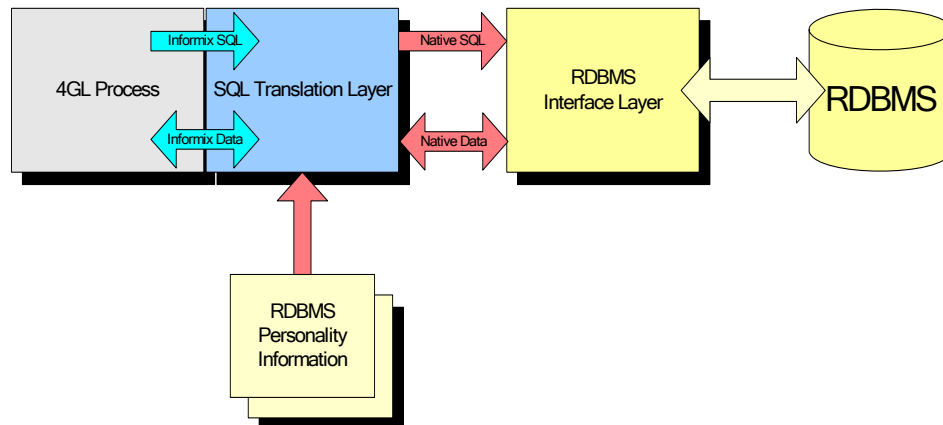
FOREACH c1 INTO rec1.*
  ...
  
```

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, Oracle 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 Oracle 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	Oracle Datatype	Notes
CHAR	CHAR	CHAR/VARCHAR2	The Oracle CHAR type has a maximum length of 2000. The VARCHAR2 type is used for columns of a length greater than 2000. Please note the VARCHAR2 has a maximum length of 4000.
VARCHAR	VARCHAR	VARCHAR2	
LVARCHAR	VARCHAR	VARCHAR2	
NCHAR	NCHAR	NCHAR	
NVARCHAR	NVARCHAR	NVARCHAR2	
SMALLINT	SMALLINT	NUMBER(5)	
INTEGER	INTEGER	NUMBER(10)	
INTEGER8	INTEGER	NUMBER(20)	
SERIAL	INTEGER	NUMBER(10)	
SERIAL8	INTEGER	NUMBER(20)	
FLOAT	FLOAT	FLOAT	
SMALLFLOAT	SMALLFLOAT	FLOAT	
DOUBLE PRECISION	FLOAT	FLOAT	
DECIMAL(p, s)	DECIMAL(p, s)	NUMBER(p, s)	
MONEY(p, s)	MONEY(p, s)	DECIMAL(p, s)	

Informix SQL Datatype	Informix 4GL Datatype	Oracle Datatype	Notes
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 Oracle database.

Oracle	Informix 4GL	Informix SQL	Notes
VARCHAR2	VARCHAR	VARCHAR	
NVARCHAR2	NVARCHAR	NVARCHAR	
NUMBER(p, s)	DECIMAL(p, s)	DECIMAL(p, s)	
LONG	TEXT	TEXT	
LONG RAW	BYTE	BYTE	
ROWID	VARCHAR(18)	VARCHAR(18)	
CHAR	CHAR	CHAR	
NCHAR	NCHAR	NCHAR	
CLOB	TEXT	TEXT	
BLOB	BYTE	BYTE	
DATE	DATE	DATE	
TIMESTAMP	DATETIME	DATETIME	
INTERVAL YEAR(n) TO MONTH	INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH	
INTERVAL DAY(n) TO SECOND(f)	INTERVAL	INTERVAL	
FLOAT	FLOAT	FLOAT	

Oracle	Informix 4GL	Informix SQL	Notes
DOUBLE PRECISION	FLOAT	FLOAT	
REAL	SMALLFLOAT	SMALLFLOAT	

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		