# logo-sql08.gif

Guide to Migrating from Informix to SQL Server 2008

SQL Server Technical Article

**Writers:** Arthur Alchangian (DB Best Technologies), Galina Shevchenko (DB Best Technologies), Yuri Tumakov (DB Best Technologies), Yuri Rusakov (DB Best Technologies)

**Technical Reviewer:** Dmitry Balin (DB Best Technologies)

**Published:** August 2009

**Applies to:** SQL Server 2008 and SQL Server 2008 R2

**Summary:** This white paper explores challenges that arise when you migrate from an Informix 11 database to SQL Server 2008. It describes the implementation differences of database objects and procedural code between the two platforms. Emulation of system functions is also discussed.

Created by: DB Best Technologies LLC

P.O. Box 7461, Bellevue, WA 98008

Tel.: (408) 202-4567

E-mail: info@dbbest.com

Web: <http://www.dbbest.com>

Copyright

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft and SQL Server are registered trademarks of Microsoft Corporation in the United States and other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

[Introduction 6](#_Toc237661072)

[Migrating Informix Data Types 7](#_Toc237661073)

[Type Mapping 7](#_Toc237661074)

[Data Type Migration Issues 9](#_Toc237661075)

[Time Data Types 9](#_Toc237661076)

[Issue: DATETIME Field Qualifier 9](#_Toc237661077)

[Issue: Serial Data Type Columns 9](#_Toc237661078)

[Issue: INTERVAL Data Type and INTERVAL Field Qualifier 10](#_Toc237661079)

[Complex Data Types 11](#_Toc237661080)

[Issue: Collection Data Types 11](#_Toc237661081)

[Issue: ROW Data Types 11](#_Toc237661082)

[User-Defined Data Types 12](#_Toc237661083)

[Issue: DISTINCT Types 12](#_Toc237661084)

[Issue: OPAQUE Types 12](#_Toc237661085)

[Migration of Table Structure 14](#_Toc237661086)

[CREATE TABLE Statement 14](#_Toc237661087)

[Issue: RAW Keyword in the CREATE TABLE Statement 14](#_Toc237661088)

[Issue: The OPERATIONAL Keyword in the CREATE TABLE Statement 14](#_Toc237661089)

[Issue: The STATIC Keyword in the CREATE TABLE Statement 14](#_Toc237661090)

[Issue: DISTINCT Keyword 14](#_Toc237661091)

[Issue: Constraints Syntax 15](#_Toc237661092)

[Issue: Default Values for the Referenced Column 17](#_Toc237661093)

[Issue: CREATE TABLE Options 18](#_Toc237661094)

[Issue: PUT clause 19](#_Toc237661095)

[Issue: USING Access-Method Clause 19](#_Toc237661096)

[Issue: LOCK MODE Options 19](#_Toc237661097)

[Issue: OF TYPE clause 20](#_Toc237661098)

[Issue: UNDER Clause in a Definition of a Typed Table 21](#_Toc237661099)

[CREATE TEMP TABLE Statement 21](#_Toc237661100)

[Issue: WITH NO LOG Option 21](#_Toc237661101)

[Issue: Temporary Table Options 21](#_Toc237661102)

[Data Manipulation Statements 23](#_Toc237661103)

[SELECT Statement 23](#_Toc237661104)

[FIRST Clause 23](#_Toc237661105)

[FIRST Clause in SELECT UNION 23](#_Toc237661106)

[MIDDLE Clause (XPS only) 24](#_Toc237661107)

[SKIP Clause 24](#_Toc237661108)

[Alias for a Nested Table Expression 25](#_Toc237661109)

[Alias for a Computed Column in Nested Table Expression 26](#_Toc237661110)

[ORDER BY in a Subquery 26](#_Toc237661111)

[Stored Procedures 28](#_Toc237661112)

[SPL Statements 28](#_Toc237661113)

[Issue: CALL Statement 28](#_Toc237661114)

[Issue: CASE Statement 28](#_Toc237661115)

[Issue: CONTINUE Statement 29](#_Toc237661116)

[Issue: DEFINE Statement 30](#_Toc237661117)

[Issue: EXIT Statement 31](#_Toc237661118)

[Issue: FOR Statement 32](#_Toc237661119)

[Issue: FOREACH Statement 33](#_Toc237661120)

[Issue: GOTO Statement 36](#_Toc237661121)

[Issue: IF Statement 37](#_Toc237661122)

[Issue: LET Statement 38](#_Toc237661123)

[Issue: LOOP Statement 39](#_Toc237661124)

[Issue: ON EXCEPTION Statement 41](#_Toc237661125)

[Issue: RAISE EXCEPTION Statement 43](#_Toc237661126)

[Issue: RETURN Statement 44](#_Toc237661127)

[Issue: SYSTEM Statement 45](#_Toc237661128)

[Issue: TRACE Statement 45](#_Toc237661129)

[Issue: WHILE Statement 46](#_Toc237661130)

[Migrating Informix Standard Functions 48](#_Toc237661131)

[Equivalent Functions 48](#_Toc237661132)

[Emulated Functions 48](#_Toc237661133)

[ADD\_MONTHS (date/dtime\_expr, integer) 48](#_Toc237661134)

[ATAN2(numeric-expression-1, numeric-expression-2) 48](#_Toc237661135)

[CEIL (num\_expression) 49](#_Toc237661136)

[CHAR\_LENGTH ( string-expression ) 49](#_Toc237661137)

[CHARACTER\_LENGTH ( string-expression ) 50](#_Toc237661138)

[CONCAT (expr\_1, expr\_2) 50](#_Toc237661139)

[CURRENT 51](#_Toc237661140)

[DECODE (expr, when\_expr, then\_expr, ..., else\_expr) 51](#_Toc237661141)

[LENGTH ( string-expression ) 52](#_Toc237661142)

[LOGN (float\_expression) 53](#_Toc237661143)

[LPAD (source\_string, length, pad\_string) 53](#_Toc237661144)

[LTRIM (source\_string, pad\_string) 54](#_Toc237661145)

[MDY (int month, int day, int year) 55](#_Toc237661146)

[MOD (dividend, divisor) 55](#_Toc237661147)

[NVL (expr1, expr2) 56](#_Toc237661148)

[OCTET\_LENGTH ( string-expression ) 56](#_Toc237661149)

[POW (float\_expression1, float\_expression2) 57](#_Toc237661150)

[RANGE (numeric\_column) 57](#_Toc237661151)

[ROOT (float\_expression1, float\_expression2) 57](#_Toc237661152)

[ROUND (float\_expression1, int\_expression2) 58](#_Toc237661153)

[RPAD (source\_string, length, pad\_string) 58](#_Toc237661154)

[RTRIM (source\_string, pad\_string) 59](#_Toc237661155)

[SQLCODE 60](#_Toc237661156)

[STDEV (numeric\_column) 60](#_Toc237661157)

[SUBSTR ( string-expression, start [, length ] ) 61](#_Toc237661158)

[SUBSTRING( string-expression FROM start [FOR length ] ) 61](#_Toc237661159)

[TODAY 62](#_Toc237661160)

[TRIM ({BOTH|LEADING|TRAILING } pad\_string FROM source\_string) 62](#_Toc237661161)

[TRUNC (float\_expression1, int\_expression2) 63](#_Toc237661162)

[VARIANCE (numeric\_column) 63](#_Toc237661163)

[WEEKDAY (date/dtime\_expr) 64](#_Toc237661164)

[Conclusion 64](#_Toc237661165)

[About DB Best Technologies 64](#_Toc237661166)

# Introduction

This white paper covers issues related to migration of an Informix 11+ database to the Microsoft® SQL Server® 2008 database software. We describe basic steps for this kind of migration and what you must know about converting database objects.

Generally, the following steps are necessary:

1. Decide how you will map Informix databases to SQL Server 2008. You have two main options:
	* Map each Informix database to a separate SQL Server database. For example, you could map the MyDB Informix database to MyDB SQL Server database.
	* Map each Informix database to a single SQL Server database but a separate schema. For example, you could map the MyDB Informix database to InformixDatabases SQL Server database, schema MyDB.
2. Convert database objects: these are tables, table constraints, indexes, views, procedures, functions, and triggers.
3. Map Informix data types to SQL Server data types.
4. Rewrite your views, procedures, and functions according to SQL Server syntax.
5. Change your applications as necessary so that they can connect and work with SQL Server 2008.

After successful database conversion, migrate your data from the old Informix database to the newly created SQL Server 2008 database. For this task you can use SQL Server Integration Services (SSIS).

# Migrating Informix Data Types

This section explains mappings and differences between Informix and SQL Server 2008 data types and specific data type handling, and it provides solutions for problems related to data types.

## Type Mapping

Following are the recommended type mappings for converting built-in data types.

|  |  |  |  |
| --- | --- | --- | --- |
| **Informix type** | **SQL Server 2008 equivalent** | **Conversion remarks** | **Alternative mappings** |
| BOOLEAN | **bit** | f maps to 0, t to 1 | None |
| CHAR(N),CHARACTER(N), LVARCHAR(N, R) | **char(n)** | N is size of string in bytes.Use VARCHAR(max) for size greater than 8000Use NCHAR(N) for multibyte text strings and NVARCHAR(max) for multi-byte text strings with size greater 4000 | **varchar(n),** **nvarchar(n)** |
| NCHAR(N) | **nchar(n)** | N is size of string in bytes. Use NVARCHAR(max) for size greater than 4000 | **nvarchar(n)** |
| VARCHAR(N, R),CHARACTER VARYING(N, R) | **varchar(n)** | N is maximum size of string in bytes.Use NVARCHAR(M) for multi-byte text strings | CHAR(N), NCHAR(N) |
| NVARCHAR(N, R) | **nvarchar(n)** | N is maximum size of string in bytes | **nchar(n)** |
| DECIMAL(P, S), DEC(P, S), NUMERIC(P, S), MONEY(P, S) | **decimal(p, s),****numeric(p, s)** | P is precision, S is scale | None |
| INT, INTEGER | **int, integer** | None | **tinyint, smallint, bigint** |
| INT8, BIGINT | **bigint** | None | None |
| DECIMAL(P), DEC(P), NUMERIC(P) | **float(53), double precision** | P is precision | **real, float(n)** |
| FLOAT, DOUBLE PRECISION (N) | **float(53), double precision** | None | **real, float(n)** |
| SMALLFLOAT, REAL | **real, float(24)** | None | **float(53), double precision** |
| TEXT, CLOB | **nvarchar(max)** | None | None |
| BYTE, BLOB | **varbinary(max)** | None | None |
| DATE | **date** | None | None |
| DATETIME | **datetime2(5)** | None | None |

# Data Type Migration Issues

This section describes data type conversion issues. Each issue is caused by an Informix feature that is not supported in SQL Server.

## Time Data Types

### Issue: DATETIME Field Qualifier

The Informix DATETIME Field Qualifier may be used to specify the largest and smallest unit of time in a DATETIME column or value. Possible qualifiers are:

* YEAR Specifies a year, in the range from A.D. 1 to 9999
* MONTH Specifies a month, in the range from 1 (January) to 12 (December)
* DAY Specifies a day, in the range from 1 to 28, 29, 30, or 31 (depending on the specific month)
* HOUR Specifies an hour, in the range from 0 (midnight) to 23
* MINUTE Specifies a minute, in the range from 0 to 59
* SECOND Specifies a second, in the range from 0 to 59
* FRACTION Specifies a fraction of a second, with up to five decimal places. The default scale is three digits (thousandth of a second).

Example:

DATETIME (2003-9-30 12:30) YEAR TO MINUTE

Solution:

Use corresponding data type to store data, for example:

* DATE for YAER TO DAY
* TIME for HOUR TO FRACTION
* INT for DAY TO DAY

Use DATEPART and DATEDIFF functions when converting these values to DATETIME2 format. In general, apply a user-defined type.

### Issue: Serial Data Type Columns

The SERIAL, SERIAL8, and BIGSERIAL data types store a sequential integer, in the positive range, that is automatically assigned by the database server when a new row is inserted. A table can have no more than one SERIAL column, but it can have a SERIAL column and either a SERIAL8 column or a BIGSERIAL column.

Example:

CREATE TABLE customer

 (

 cust\_id serial,

 . . .

 )

Solution:

Use IDENTITY for columns that would be serial in Informix with corresponding data type (INT or BIGINT). Use SET IDENTITY\_INSERT <*table*> ON to insert custom values in such columns.

CREATE TABLE customer

 (

 cust\_id INT NOT NULL IDENTITY(1, 1),

 . . .

 )

### Issue: INTERVAL Data Type and INTERVAL Field Qualifier

The INTERVAL data type stores a value that represents a span of time. INTERVAL types are divided into two classes: *year-month intervals* and *day-time intervals*. A year-month interval can represent a span of years and months, and a day-time interval can represent a span of days, hours, minutes, seconds, and fractions of a second:

INTERVAL *largest\_qualifier*(*n*) TO *smallest\_qualifier*

Qualifiers are:

* YEAR-MONTH INTERVAL
* YEAR number of years
* MONTH number of months
* DAY-TIME INTERVAL
* DAY number of days
* HOUR number of hours
* MINUTE number of minutes
* SECOND number of seconds
* FRACTION decimal fraction of a second, with up to 5 digits. The default scale is 3 digits (thousandth of a second). To specify a nondefault scale, write FRACTION(*n*), where 1 ≤ *n* ≤ 5.

Example:

INTERVAL (60 01:30) DAY TO MINUTE

Solution:

Use the corresponding integer data type to store values and DATEPART and DATEADD functions to operate. In general case, use a user-defined type.

## Complex Data Types

### Issue: Collection Data Types

A collection data type is a complex type that is made up of one or more elements, all of the same data type.

The SET data type is an unordered collection type that stores unique elements. The elements in a SET have no ordinal position.

The MULTISET data type is a collection type that stores a nonordered set that can include duplicate element values. The elements in a MULTISET have no ordinal position. The LIST data type is a collection type that stores ordered, nonunique elements; that is, it allows duplicate element values. The elements of a LIST have ordinal positions; that is, the list has a first element, a second element, and so on.

Example:

LIST{"blue", "green", "yellow"}

INSERT INTO table1 VALUES (MULTISET{5, 9, 7, 5})

CREATE TABLE tab ( c CHAR(5), s SET(INTEGER NOT NULL) )

Solution:

Write a SQL Server user-defined type.

### Issue: ROW Data Types

A named ROW type is declared by its name. That identifier must be unique within the schema. No two named ROW types can be equal, even if they have identical structures, because they have different names.

An unnamed ROW type is a ROW type that contains fields but has no user-defined name. An unnamed ROW type is defined by its structure. Two unnamed ROW types are equal if they have the same structure (meaning the ordered list of the data types of the fields). If two unnamed ROW types have the same number of fields, and if the order of the data type of each field in one ROW type matches the order of data types of the corresponding fields in the other ROW data type, then the two unnamed ROW data types are equal.

Example:

name\_t (lname CHAR(15), initial CHAR(1), fname CHAR(15))

ROW (dept char(15), rating char(1) name char(15))

Solution:

Write a SQL Server user-defined type.

## User-Defined Data Types

### Issue: DISTINCT Types

A distinct data type has the same internal structure as some other source data type in the database. The source type can be a built-in or extended data type. What distinguishes a distinct type from its source type are support functions that are defined on the distinct type.

Informix example:

CREATE DISTINCT TYPE birthday AS DATE;

Solution:

Use a SQL Server alias data type.

SQL Server example:

CREATE TYPE birthday FROM DATE

### Issue: OPAQUE Types

An opaque data type is a user-defined data type that is fully encapsulated. That is, its internal structure is unknown to the database server. User-defined types that are not DISTINCT types whose source types are built-in types are opaque.

Example:

CREATE OPAQUE TYPE var\_type (INTERNALLENGTH=VARIABLE, MAXLEN=4096);

Solution:

Write a SQL Server user-defined type.

# Migration of Table Structure

This section discusses differences between the syntax of creating permanent and temporary tables in Informix and SQL Server.

## CREATE TABLE Statement

### Issue: RAW Keyword in the CREATE TABLE Statement

In Informix, it is a nonlogging table that does not support referential constraints, primary key constraints, or unique constraints, but that can be indexed and updated. This type of table is used for quickly loading data. SQL Server table organization does not allow division of table types.

Solution:

Ignore this keyword.

### Issue: The OPERATIONAL Keyword in the CREATE TABLE Statement

In Informix, it is a logging table that uses light appends; it cannot be restored from archive. The OPERATIONAL keyword is used on tables that are refreshed frequently, because light appends allow the quick addition of many rows. SQL Server table organization does not have such division of table types.

Solution:

Ignore this keyword.

### Issue: The STATIC Keyword in the CREATE TABLE Statement

In Informix, it is a nonlogging table that can contain index and referential constraints but cannot be updated. This type of table is used for read-only operations, because no logging or locking overhead occurs. SQL Server table organization does not have such division of table types.

Solution:

Ignore this keyword.

### Issue: DISTINCT Keyword

In Informix, the DISTINCT keyword requires that a column or set of columns accept only unique data values. The keyword DISTINCT is a synonym for UNIQUE.

Informix example:

CREATE TABLE accounts

(acc\_name CHAR(12),

 acc\_num INT DISTINCT CONSTRAINT acc\_num);

Solution:

Replace the DISTINCT constraint with a UNIQUE constraint.

SQL Server example:

CREATE TABLE accounts

(acc\_name CHAR(12),

 acc\_num INT CONSTRAINT acc\_num UNIQUE );

### Issue: Constraints Syntax

In Informix, the constraints syntax differs from SQL Server.

Informix examples:

Example 1:

CREATE TABLE accounts

 (acc\_name CHAR(12),

 acc\_num INTEGER UNIQUE CONSTRAINT acc\_num);

Example 2:

CREATE TABLE accounts

 (acc\_name CHAR(12),

 acc\_num INTEGER PRIMARY KEY CONSTRAINT acc\_num);

Example 3:

CREATE TABLE order\_items

 (

 order\_id INT,

 line\_item\_id INT not null,

 unit\_price DECIMAL(6,2),

 quantity INT,

 UNIQUE (order\_id,line\_item\_id) CONSTRAINT items\_constr

 );

Solution:

Convert the constraint to SQL Server syntax by moving the type of constraint to the end of the constraint definition.

SQL Server examples:

Example 1:

CREATE TABLE accounts

 (acc\_name CHAR(12),

 acc\_num INT CONSTRAINT acc\_num UNIQUE );

Example 2:

CREATE TABLE accounts

 (acc\_name CHAR(12),

 acc\_num INT CONSTRAINT acc\_num PRIMARY KEY);

Example 3:

CREATE TABLE order\_items

 (

 order\_id INT,

 line\_item\_id INT not null,

 unit\_price DECIMAL(6,2),

 quantity INT,

 CONSTRAINT items\_constr UNIQUE (order\_id,line\_item\_id)

 );

### Issue: [Default Values for the Referenced Column](http://publib.boulder.ibm.com/infocenter/idshelp/v111/topic/com.ibm.sqls.doc/sqls02.htm#ToC_347)

In Informix, if the referenced table is different from the referencing table, you do not need to specify the referenced column; the default column is the primary key column (or columns) of the referenced table.

Informix example:

CREATE TABLE accounts (

 acc\_num INTEGER PRIMARY KEY,

 acc\_type INTEGER,

 acc\_descr CHAR(20));

CREATE TABLE sub\_accounts (

 sub\_acc INTEGER PRIMARY KEY,

 ref\_num INTEGER REFERENCES accounts,

 sub\_descr CHAR(20));

Solution:

Specify the referenced column(s) (which is a primary key column(s)) explicitly.

SQL Server example:

CREATE TABLE accounts (

 acc\_num INT PRIMARY KEY,

 acc\_type INT,

 acc\_descr CHAR(20));

CREATE TABLE sub\_accounts (

 sub\_acc INT PRIMARY KEY,

 ref\_num INT REFERENCES accounts (acc\_num),

 sub\_descr CHAR(20));

### Issue: CREATE TABLE Options

In Informix, there are a number of options that can be defined in the CREATE TABLE statement. These options are concerned with table storage strategies. The SQL Server CREATE TABLE statement does not have such options, because the storage options are defined mainly when the database is created.

Informix example:

CREATE TABLE family

 (

 id\_num INT UNIQUE,

 name CHAR(40),

 nickname CHAR(20),

 mother CHAR(40),

 father CHAR(40)

 )

 IN famdata;

Solution:

Because table storage principles in Informix and SQL Server are different, most of the Informix options cannot be converted to SQL Server, except for IN<dbspace> and the PARTITION BY clause. The Informix IN <dbspace> clause can be replaced with the ON <filegroup> clause in SQL Server. But note that the filegroup should be created in the database in advance. PARTITION BY is the same in Informix and SQL Server.

SQL Server example:

CREATE TABLE family

 (

 id\_num INT UNIQUE,

 name CHAR(40),

 nickname CHAR(20),

 mother CHAR(40),

 father CHAR(40)

 )

 ON famdata\_filegroup;

### Issue: PUT clause

In Informix, the PUT clause is used to specify the storage spaces and characteristics for each column that will contain smart large objects. SQL Server does not have this functionality.

Solution:

 In SQL Server, you only need to specify the corresponding data type (which can store large amount of data). If you need emulate this Informix functionality, write CLR types instead and design functions processing them by means of C#.

### Issue: USING Access-Method Clause

In Informix, the USING Access Method clause can specify an access method. SQL Server does not have this functionality.

Solution:

Ignore this keyword and use other methods available in SQL Server to manage access to tables.

### Issue: LOCK MODE Options

The LOCK MODE options are used to specify the locking granularity of the table. SQL Server does not have this functionality.

Solution:

In SQL Server, the Database Engine manages granularity of locks automatically. If the number of fine-grained locks is too large, it applies lock escalation. Lock escalation is the process of converting many fine-grain locks into fewer coarse-grain locks.

### Issue: OF TYPE clause

In Informix, the OF TYPE clause is used to create a typed table for an object-relational database. SQL Server does not allow the creation of typed tables.

Informix example:

CREATE ROW TYPE student\_t

 (name VARCHAR(30),

 average REAL,

 birthdate DATETIME YEAR TO DAY)

CREATE TABLE students OF TYPE student\_t

Solution:

Create the table defining the columns from the ROW TYPE definition.

SQL Server example:

CREATE TABLE students(

name VARCHAR(30),

average REAL,

birthdate DATE

)

### Issue: UNDER Clause in a Definition of a Typed Table

When the UNDER clause is used, the ROW type must be derived from the ROW type of the supertable. A type hierarchy must already exist in which the named ROW type of the new table is a subtype of the named ROW type of the supertable. SQL Server does not allow the use of table hierarchies.

Solution:

You can emulate this functionality by means of XML and CLR implementation.

## CREATE TEMP TABLE Statement

### Issue: WITH NO LOG Option

The WITH NO LOG option is used to reduce the overhead of transaction logging. SQL Server syntax for creating temporary tables does not include a comparable option.

Informix example:

CREATE TEMP TABLE tab2 (fname CHAR(15), lname CHAR(15))

 WITH NO LOG;

Solution:

Ignore this option.

SQL Server example:

CREATE TABLE #tab2 (fname CHAR(15), lname CHAR(15))

### Issue: Temporary Table Options

Informix supports the following options for temporary table definition: WITH CRCOLS, storage options, LOCK MODE options, USING access-method clause. SQL Server does not allow the definition of options for temporary tables.

Solution:

Ignore these options.

# Data Manipulation Statements

## SELECT Statement

### FIRST Clause

In Informix, the FIRST clause sets the maximum number of rows that can be retrieved.

Informix example:

select first 10 id, val

 from tbl

 order by id desc

Solution:

In SQL Server, emulate the FIRST clause by using the TOP clause of the SELECT statement.

SQL Server example:

SELECT TOP 10 id, val

 FROM tbl

 ORDER BY id desc

### FIRST Clause in SELECT UNION

In Informix, the FIRST clause sets the maximum number of rows that can be retrieved after a UNION operation is performed.

Informix example:

 select first 4 id, val from tbl

 union ALL

 select id, val from tbl order by id desc

Solution:

In SQL Server, emulate the FIRST clause by applying the TOP clause of the SELECT statement to the result of the union operation.

SQL Server example:

SELECT TOP 4 id, val FROM

( SELECT id, val FROM tbl

 UNION ALL

 SELECT id, val FROM tbl

) AS tmp

 ORDER BY id desc

### MIDDLE Clause (XPS only)

In Informix, the MIDDLE clause sets the maximum number of rows that can be retrieved from the middle of the set of qualifying rows.

Informix example:

select middle 10 id, val

 from tbl

 order by id desc

Solution:

Replace the query with a common table expression and a ROW\_NUMBER ( ) OVER (order\_by\_clause ) construct that returns the sequential number of a row within a result set.

### SKIP Clause

In Informix, the SKIP clause sets the number of rows to skip from top of the set of qualifying rows.

Informix example:

select skip 2 id, val

 from tbl

 order by id desc

Solution:

In SQL Server, emulate the SKIP clause by using the following construction.

SQL Server example:

SELECT id, val FROM

(SELECT id, val, row\_number() OVER ( PARTITION BY id, val ORDER BY (SELECT 1)) AS Tmp$Num FROM tbl

 EXCEPT

 SELECT TOP 2 id, val, row\_number() OVER ( PARTITION BY id, val ORDER BY (SELECT 1)) AS Tmp$Num FROM tbl ORDER BY 1 desc

) AS tmp

ORDER BY 1 desc

### Alias for a Nested Table Expression

In Informix, you do not need to specify an alias for a nested table expression. In SQL Server, you must include an alias for a derived table (subquery).

Informix example:

select val

from (select id, val from tbl)

order by id;

Solution:

In SQL Server, add an alias for the nested table expression.

SQL Server example:

SELECT val

FROM (SELECT id, val FROM tbl) AS c

ORDER BY id;

### Alias for a Computed Column in Nested Table Expression

In Informix, you do not need to specify an alias for a computed column in nested table expression. In SQL Server, you must include an alias name for this column.

Informix example:

select \*

from (select id, val, id||val from tbl)

order by id;

Solution:

In SQL Server, add an alias for a computed column in the nested table expression.

SQL Server example:

SELECT \*

FROM (SELECT id, val, CAST(id AS varchar)+val somename FROM tbl) AS c

ORDER BY id;

### ORDER BY in a Subquery

In Informix, the ORDER BY clause can be used in subqueries. The same ordering that is used in an aliased query should be applied to the result table of the subselect. This functionality of ORDER BY is not supported in SQL Server.

Informix example:

select \*

from ( select id as a , val as b from tbl

 order by b ) as sq

Solution:

In SQL Server, add the ORDER OF clause with a list of fields or aliases from the subquery sort specification. Then either remove the ORDER BY clause from the subquery (see example 1), or add the TOP clause to the subquery (see example 2).

SQL Server example 1:

SELECT \*

FROM ( SELECT id AS a , val AS b FROM tbl ) AS sq

ORDER BY b

SQL Server example 2:

SELECT \*

FROM ( SELECT TOP (100) PERCENT id AS a , val AS b

 FROM tbl ORDER BY b ) AS sq

ORDER BY b

# Stored Procedures

This section discusses differences between the Informix SPL procedural language and SQL Server Transact-SQL. This includes the creation and calling of stored procedures as well as working with local variables, cursors, and control-of-flow statements.

## SPL Statements

### Issue: CALL Statement

The CALL statement invokes a user-defined routine (UDR). The CALL statement is identical in behavior to the EXECUTE PROCEDURE and EXECUTE FUNCTION statements, but it can only be used from within an SPL routine.

Informix example:

CALL no\_args (10,20);

CALL yes\_args (5) RETURNING i, j, k;

Solution:

Use the EXECUTE statement for procedures. Functions can also be called directly.

SQL Server example:

EXEC no\_args 10,20

EXEC yes\_args 5, @I OUTPUT , @j OUTPUT, @k OUTPUT

### Issue: CASE Statement

The CASE statement is used to create a set of conditional branches within an SPL routine. The CASE statement is a fast alternative to the IF statement.

Informix example:

 CASE i

 WHEN 1 THEN LET j = x;

 WHEN 2 THEN LET k = x;

 WHEN 3 THEN LET l = x;

 WHEN 4 THEN LET m = x;

 ELSE

 LET n = x

END CASE;

Solution:

Use multiple IF…ELSE statements.

SQL Server example:

IF @i = 1 SET @j = @x

 ELSE IF @i = 2 SET @k = @x

 ELSE IF @i = 3 SET @l = @x

 ELSE IF @i = 4 SET @m = @x

 ELSE SET @n = @x

### Issue: CONTINUE Statement

The CONTINUE statement starts the next iteration of the innermost FOR, LOOP, WHILE, or FOREACH loop.

Informix example:

FOR i IN (3 TO 15 STEP 2)

 INSERT INTO testtable1 values(i, null, null);

 IF i = 11

 CONTINUE FOR;

 END IF;

 INSERT INTO testtable2 values(i, null, null);

 END FOR;

Solution:

Use the CONTINUE keyword.

SQL Server example:

DECLARE @i INT = 1

WHILE @i <= 15

 BEGIN

 SET @i += 2

 INSERT testtable1 values(@i, null, null);

 IF @i = 11 CONTINUE;

 INSERT testtable2 values(@i, null, null);

 END

### Issue: DEFINE Statement

The DEFINE statement is used to declare local variables that an SPL routine uses, or to declare global variables that can be shared by several SPL routines.

Informix example:

DEFINE GLOBAL gl\_out CHAR(15); -- global variable

DEFINE word INT DEFAULT 13;

DEFINE b\_day DATE;

Solution:

For local variables, use the DECLARE statement. You can also emulate global variables by storing and reading variable data from custom tables.

SQL Server example:

DECLARE @word INT = 15;

DECLARE @b\_day DATE;

### Issue: EXIT Statement

The EXIT statement transfers control of execution from an iterative statement, causing the innermost loop of the enclosing statement type (FOR, FOREACH, LOOP, or WHILE) to terminate. If no loop label or WHEN condition is specified, execution resumes at the first statement that follows the current FOR, FOREACH, LOOP, or WHILE statement.

Informix example:

DEFINE i,s,j, INT;

FOR j = 1 TO 20

 IF j > 10 THEN

 CONTINUE FOR;

 END IF

 LET i,s = j,0;

 WHILE i > 0

 LET i = i -1;

 IF i = 5 THEN

 EXIT FOR;

 END IF

 END WHILE

 END FOR

Solution:

Use the BREAK keyword in simple cases. For loop labels and nested cycles, use GOTO.

SQL Server example:

DECLARE @i INT, @s INT, @j INT = 0;

WHILE @j < 20

 BEGIN

 SET @j += 1

 IF @j > 10 CONTINUE

 SELECT @i = @j, @s = 0

 WHILE @i > 0

 BEGIN

 SET @i -= 1;

 IF @i = 5 GOTO for\_exit; -- to exit outer loop GOTO is used instead of BREAK

 END

 END

for\_exit:

### Issue: FOR Statement

The FOR statement is used to initiate a controlled (definite) loop with guaranteed termination. The FOR statement uses expressions or range operators to specify a finite number of iterations for a loop.

Informix example:

FOR index\_var IN (12 TO 21 STEP 2)

 -- statement block

END FOR;

Solution:

Use the WHILE keyword with provided logic to handle iterations—the cycle variable must be declared and assigned before the cycle and changed manually during the cycle.

SQL Server example:

DECLARE @index\_var INT = 10;

WHILE @index\_var <= 21

 BEGIN

 SET @index\_var += 2

 -- statement block

 END

### Issue: FOREACH Statement

The FOREACH statement is used to select and manipulate more than one row or collection. A FOREACH loop is the procedural equivalent of using a cursor.

Informix example:

CREATE PROCEDURE foreach\_ex()

 DEFINE i, j INT;

 FOREACH SELECT c1 INTO i FROM tab ORDER BY 1

 INSERT INTO tab2 VALUES (i);

 END FOREACH

 FOREACH cur1 FOR SELECT c2, c3 INTO i, j FROM tab

 IF j > 100 THEN

 DELETE FROM tab WHERE CURRENT OF cur1;

 CONTINUE FOREACH;

 END IF

 UPDATE tab SET c2 = c2 + 10 WHERE CURRENT OF cur1;

 END FOREACH

 FOREACH EXECUTE PROCEDURE bar(10,20) INTO i

 INSERT INTO tab2 VALUES (i);

 END FOREACH

END PROCEDURE;

Solution:

Use a cursor with the WHILE cycle to get data. If you are executing a FOREACH routine, use a temporary table or a table variable to hold data through the INSERT…EXEC statement from the routine before you declare the cursor.

SQL Server example:

CREATE PROCEDURE foreach\_ex

AS

 DECLARE @i INT, @j INT;

 DECLARE cur CURSOR FOR SELECT c1 INTO i FROM tab ORDER BY 1

 OPEN cur

 FETCH NEXT FROM cur INTO @i

 WHILE @@FETCH\_STATUS = 0

 BEGIN

 INSERT tab2 VALUES (@i);

 FETCH NEXT FROM cur INTO @i

 END

 CLOSE cur

 DEALLOCATE cur

 DECLARE cur1 CURSOR FOR SELECT c2, c3 FROM tab

 OPEN cur1

 FETCH NEXT FROM cur1 INTO @i, @j

 WHILE @@FETCH\_STATUS = 0

 BEGIN

 IF @j > 100

 BEGIN

 DELETE tab WHERE CURRENT OF cur1;

 FETCH NEXT FROM cur INTO @i, @j

 CONTINUE;

 END

 UPDATE tab SET c2 += 10 WHERE CURRENT OF cur1;

 FETCH NEXT FROM cur INTO @i, @j

 END

 CLOSE cur1

 DEALLOCATE cur1

 DECLARE @Result TABLE (res INT)

 INSERT @Result EXEC bar(10, 20)

 DECLARE cur CURSOR FOR SELECT res FROM @Result ORDER BY 1

 OPEN cur

 FETCH NEXT FROM cur INTO @i

 WHILE @@FETCH\_STATUS = 0

 BEGIN

 INSERT tab2 VALUES (@i);

 FETCH NEXT FROM cur INTO @i

 END

 CLOSE cur

 DEALLOCATE cur

### Issue: GOTO Statement

The GOTO statement is used to transfer control of program execution to the statement that has a specified statement label.

Informix example:

CREATE FUNCTION jump\_back()

 RETURNING INT;

 DEFINE i,j INT;

 ...

 <<back>>

 LET j = j + i

 FOR i IN (1 TO 52 STEP 5)

 IF i < 11 THEN

 LET j = j + 3

 CONTINUE FOR;

 END IF;

 IF j > 100 THEN

 GOTO back

 END IF;

 RETURN j;

 END FOR;

END FUNCTION;

Solution:

Use the GOTO keyword and labels.

SQL Server example:

CREATE FUNCTION jump\_back()

RETURNS INT

AS

BEGIN

 DECLARE @i INT, @j INT;

 ...

 back:

 SET @j += @i

 SET @i = -4

 WHILE @i <= 52

 BEGIN

 SET @i += 5

 IF @i < 11

 BEGIN

 SET @j += 3

 CONTINUE

 END

 IF @j > 100 GOTO back

 RETURN @j

 END

END

### Issue: IF Statement

The IF statement is used to create a logical branch within an SPL routine.

Informix example:

IF i = 1

 THEN LET j = k

END IF

Solution:

Use IF…ELSE keywords.

SQL Server example:

IF @i = 1 SET @j = @k

### Issue: LET Statement

The LET statement is used to assign values to variables or to call a user-defined SPL routine and to assign the returned value or values to SPL variables.

Informix example:

LET a = c + d;

LET a, b = c, d;

LET a, b, c = 1, test\_func(1);

Solution:

Use the SET keyword for simple assignment; for multiple-variable and function assignment, use SELECT.

SQL Server example:

SET @a = @c + @d

SELECT @a = @c, @b = @d

SELECT @a = 1, @b = f.b, @c = f.c FROM dbo.test\_func(1) f

### Issue: LOOP Statement

The LOOP statement is used to define a loop with an indeterminate number of iterations. The LOOP statement is an iterative statement that resembles the FOR and WHILE statements. Like FOR and WHILE, the LOOP statement can have an optional loop label. It can include the CONTINUE statement to specify another iteration and the EXIT statement to terminate execution of the loop.

Besides resembling FOR and WHILE in its functionality, the LOOP statement can use the syntax of FOR or WHILE that precedes the statement block.

Informix example:

--simple loop

LOOP

LET i = i + 1;

 EXIT WHEN i = 4;

END LOOP;

-- FOR loop

FOR i IN (1 TO 5) LOOP

 EXIT WHEN i = 5;

 END LOOP;

-- WHILE loop

WHILE (i < 6) LOOP

 LET i = i + 1;

 IF i = 5 THEN EXIT;

 ELSE

 CONTINUE;

 END IF

END LOOP;

-- Labeled loop

<<voort>>

 LOOP

 LET x = x+1;

 <<endo>>

 WHILE ( i < 10 ) LOOP

 LET x = x+1;

 EXIT endo WHEN x = 7;

 EXIT voort WHEN x > 9;

 END LOOP endo;

 LET x = x+1;

 END LOOP voort;

Solution:

Use the WHILE statement. For loop labels and nested cycles, you can also use GOTO.

SQL Server example:

--simple loop

WHILE 1 = 1

 BEGIN

 SET @i += 1

 IF @i = 4 BREAK

 END

-- FOR loop

DECLARE @i INT = 0

WHILE @i <= 5

 BEGIN

 SET @i += 1

 IF @i = 5 BREAK

 END

-- WHILE loop

WHILE (@i < 6)

 BEGIN

 SET @i += 1;

 IF @i = 5 BREAK

 ELSE CONTINUE

 END

-- Labeled loop

voort:

WHILE 1 = 1

 BEGIN

 SET @x += 1

 endo:

 WHILE @i < 10

 BEGIN

 SET @x += 1

 IF @x = 7 GOTO endo

 IF @x > 9 GOTO voort

 END

 SET @x += 1

 END

### Issue: ON EXCEPTION Statement

The ON EXCEPTION statement is used to specify actions to be taken for any error, or for a list of one or more specified errors, during execution of a statement block. The ON EXCEPTION statement, together with the RAISE EXCEPTION statement, provides an error-trapping and error-recovery mechanism for SPL. ON EXCEPTION can specify the errors that you want to trap as the SPL routine executes, and it specifies the action to take if the error occurs within the statement block. ON EXCEPTION can specify an error number list in the IN clause, or it can include no IN clause. If the IN clause is omitted, all errors are trapped.

Informix example:

CREATE PROCEDURE ex\_test()

 DEFINE error\_num INT;

 ...

 ON EXCEPTION SET error\_num

 -- action C

 END EXCEPTION

 ON EXCEPTION IN (-300)

 -- action B

 END EXCEPTION

 ON EXCEPTION IN (-210, -211, -212) SET error\_num

 -- action A

 END EXCEPTION

--test block

Solution:

Use the TRY…CATCH statement.

SQL Server example:

CREATE PROCEDURE ex\_test

AS

 DECLARE @error\_num INT

 BEGIN TRY

 --test block

 END TRY

 BEGIN CATCH

 DECLARE @error\_number INT

 SET @error\_number = ERROR\_NUMBER()

 IF @error\_number IN (50002, 50003, 50004)

 BEGIN

 SET @error\_num = @error\_number

 -- action A

 END

 ELSE

 IF @error\_number= 50001

 BEGIN

 -- action B

 END

 ELSE

 BEGIN

 -- action C

 END

 END CATCH

### Issue: RAISE EXCEPTION Statement

The RAISE EXCEPTION statement is used to simulate an error or to generate an error with a custom message. The special error number -746 enables you to produce a customized message.

Informix example:

RAISE EXCEPTION -208, 0;

RAISE EXCEPTION -746, 0, 'Some error';

Solution:

Use the RAISERROR() function. SQL Server error codes differ from Informix codes; if you want to use an error code rather than build the message dynamically, you must provide the number of an equivalent message in the **sys.messages** catalog view, or if the message does not exist there, you can add a custom error code and message view using the **sp\_addmessage** system stored procedure.

SQL Server example:

RAISERROR (50000, 10, 1);

RAISERROR ('Some error', 16, 1);

### Issue: RETURN Statement

The RETURN statement is used to specify what values (if any) the SPL function returns to the calling context.

Informix example:

CREATE FUNCTION fn\_return (stockno INT) RETURNING CHAR (15);

 DEFINE des CHAR(15);

 SELECT descript INTO des FROM stock

 WHERE stocknum = stockno;

 RETURN des;

END FUNCTION;

Solution:

Use the RETURN statement.

SQL Server example:

CREATE FUNCTION fn\_return (@stockno INT)

RETURNS CHAR(15)

AS

BEGIN

 DECLARE @des CHAR(15)

 SELECT @des = descript FROM stock

 WHERE stocknum = @stockno

 RETURN @des

END

### Issue: SYSTEM Statement

The SYSTEM statement is used to issue an operating-system command from within an SPL routine.

Solution:

Use the **xp\_cmdshell** extended stored procedure. It is disabled by default and can be enabled and disabled by using the Policy-Based Management or by executing the **sp\_configure** system procedure. The operating-system command must be converted manually depending on the operating system.

### Issue: TRACE Statement

The TRACE statement is used to control the generation of debugging output.

Informix example:

CREATE PROCEDURE tracing ()

 DEFINE i INT;

BEGIN

 ON EXCEPTION IN (1)

 END EXCEPTION; -- do nothing

 SET DEBUG FILE TO '/tmp/mytrace.trace';

 TRACE OFF;

 TRACE 'Forloop starts';

 FOR i IN (1 TO 1000)

 BEGIN

 TRACE 'FOREACH starts';

 FOREACH SELECT...INTO a FROM t

 IF <some condition> THEN

 RAISE EXCEPTION 1 -- emergency exit

 END IF

 END FOREACH -- return some value

 END

 END FOR -- do something

END;

END PROCEDURE

Solution:

Use SQL Server Profiler to see the trace of executing statements. In procedures, the PRINT statement can be used, but the results of PRINT statements are only shown in the Messages window in SQL Server Management Studio; they cannot be saved to a file.

### Issue: WHILE Statement

The WHILE statement is used to establish a loop with variable end conditions.

Informix example:

 DEFINE i INT;

 LET i = 1;

 WHILE i < 10

 INSERT INTO tab\_2 VALUES (i);

 LET i = i + 1;

 END WHILE;

Solution:

Use a WHILE statement.

SQL Server example:

DECLARE @i INT

SET @i = 1

WHILE @i < 10

 BEGIN

 INSERT tab\_2 VALUES (@i)

 SET @i+ = 1

 END

# Migrating Informix Standard Functions

This section describes how to map Informix standard functions to equivalent SQL Server functions, and it provides solutions for emulating Informix functions.

## Equivalent Functions

The following Informix system functions are usable as they stand, in SQL Server code:

ABS, ACOS, ASCII, ASIN, ATAN, AVG, CASE, COS, COUNT, DAY, EXP, FLOOR, LOG10, LOWER, MAX, MIN, MONTH, NULLIF, POWER, REPLACE, SIN, SQRT, SUM, TAN, UPPER, YEAR

## Emulated Functions

The following Informix system functions can be emulated by using various SQL Server functions or Transact-SQL constructions.

### ADD\_MONTHS (date/dtime\_expr, integer)

The value returned is the sum of the DATE or DATETIME value of the first argument and an INTERVAL UNITS MONTH value that is based on the number of months that the second argument specifies.

Informix example:

add\_months (current, 5)

Solution:

In SQL Server, use the DATEADD function with a *datepart* of **month**.

SQL Server example:

DATEADD (month, 5, GETDATE())

### ATAN2(numeric-expression-1, numeric-expression-2)

Returns the arc-tangent, in radians, of the ratio of two numbers.

Informix example:

atan2 ( 0.52, 0.60 )

Solution:

In SQL Server, use the ATN2 function.

SQL SERVER EXAMPLE:

atn2 ( 0.52, 0.60 )

### CEIL (num\_expression)

Returns the smallest integer that is greater than or equal to its single argument.

Informix example:

ceil ( 32.3 )

ceil ( -32.3 )

Solution:

In SQL Server, use the CEILING function.

SQL Server example:

CEILING ( 32.3 )

CEILING ( -32.3 )

### CHAR\_LENGTH ( string-expression )

Returns the number of characters in a string.

Informix example:

char\_length ('InForMix')

Solution:

In SQL Server, use the LEN function.

SQL Server example:

LEN ('InForMix')

### CHARACTER\_LENGTH ( string-expression )

Returns the number of characters in a string.

Informix example:

char\_length ('InForMix')

Solution:

In SQL Server, use the LEN function.

SQL Server example:

LEN ('InForMix')

### CONCAT (expr\_1, expr\_2)

Returns a character string that appends the string representation of the value returned by its second argument to the string representation of the value returned by its first argument.

If one of its arguments returns a NULL value, the function returns the string representation of its other argument.

Informix example:

concat ('Some', 'String')

concat (null, 'String')

Solution:

In SQL Server, use the ‘+’ operation to concatenate strings. Use the ISNULL function for both arguments.

SQL Server example:

ISNULL('Some', '') + ISNULL('String', '' )

ISNULL (null, '') + ISNULL ('String', '')

### CURRENT

Returns the current date and time.

Informix example:

current

Solution:

In SQL Server, use the GETDATE function.

SQL Server example:

GETDATE()

### DECODE (expr, when\_expr, then\_expr, ..., else\_expr)

Returns different results depending on the values found in a specified column.

Informix example:

argn( 2, '1', 'a', '2', 'b', '3', 'c', 'z' )

argn( 15, '1', 'a', '2', 'b', '3', 'c', 'z' )

Solution:

In SQL Server, use the CASE function.

SQL Server example:

CASE 2

 WHEN 1 THEN 'a'

 WHEN 2 THEN 'b'

 WHEN 3 THEN 'c'

 ELSE 'z'

 END

CASE 15

 WHEN 1 THEN 'a'

 WHEN 2 THEN 'b'

 WHEN 3 THEN 'c'

 ELSE 'z'

 END

### LENGTH ( string-expression )

Returns the number of bytes in a string.

Informix example:

length( 'text' )

Solution:

In SQL Server, use the DATALENGTH function.

SQL Server example:

DATALENGTH('text')

### LOGN (float\_expression)

Returns the natural logarithm of a numeric argument.

Informix example:

logn ( 10 )

Solution:

In SQL Server, use the LOG function.

SQL Server example:

LOG ( 10 )

### LPAD (source\_string, length, pad\_string)

Returns a copy of source\_string that is left-padded to the total number of characters specified by length.

Informix example:

lpad('text', 5, '<>')

Solution:

In SQL Server, use the following CASE construction:

CASE WHEN @length>0 AND @length-LEN(@source\_string)>0 THEN

REPLICATE (@pad\_string, (@length-LEN(@source\_string))/LEN(@pad\_string))+

 substring(@pad\_string, 1, (@length-LEN(@source\_string))%LEN(@pad\_string))+@source\_string

WHEN @length>=0 AND @length-LEN(@source\_string)<=0 THEN

 SUBSTRING(@source\_string, 1, @length)

ELSE NULL END

If a negative value for the length argument returns NULL, or if the length argument is less than the length of the source string, SUBSTRING is returned. Otherwise, the source string is left-padded by the value specified in the pad\_string argument as needed.

SQL Server example:

CASE WHEN 5>0 AND 5-len('text')>0 THEN

 REPLICATE ('<>', (5-len('text'))/len('<>'))+

 SUBSTRING('<>', 1, (5-len('text'))%len('<>'))+'text'

WHEN 5>=0 and 5-len('text')<=0 THEN

 SUBSTRING('text', 1, 5)

ELSE NULL END

### LTRIM (source\_string, pad\_string)

The LTRIM function removes specified leading pad characters from a string.

Informix example:

ltrim('>><<text>><<', '<>')

Solution:

SQL Server also has a function named LTRIM, but it can trim only spaces. To emulate this function in other cases, use the following construction:

SUBSTRING(@source\_string, PATINDEX('%[^'+@pad\_string+']%',@source\_string), LEN(@source\_string))

SQL Server example:

SUBSTRING('>><<text>><<', patindex('%[^'+'<>'+']%','>><<text>><<'), len('>><<text>><<'))

### MDY (int month, int day, int year)

The MDY function takes as its arguments three integer expressions that represent the month, day, and year, and it returns a type DATE value.

Informix example:

mdy ( 02, 27, 159 )

Solution:

In SQL Server, use multiple CAST, REPLICATE, and LEN functions.

SQL Server example:

CAST ( CAST (02 AS varchar(2)) + '/' + CAST (27 AS varchar(2))+ '/' + REPLICATE('0', 4-len(CAST(159 AS varchar(4))))+ CAST(159 AS varchar(4)) AS date)

### MOD (dividend, divisor)

Returns the integer remainder of a division of the integer part of the first argument (the dividend) by the integer part of the second argument (the divisor).

Informix example:

mod ( 24.2, 5 )

Solution:

In SQL Server, use the % operation and CAST function.

SQL Server example:

CAST( 24.2 AS numeric(38,0)) % CAST( 5 AS numeric(38,0))

### NVL (expr1, expr2)

NVL evaluates expression1. If expression1 is not NULL, then NVL returns the value of expression1. If expression1 is NULL, NVL returns the value of expression2.

Informix example:

nvl ( 24, 0 )

nvl ( null, 0 )

Solution:

In SQL Server, use the ISNULL function.

SQL Server example:

ISNULL ( 24, 0 )

ISNULL ( NULL, 0 )

### OCTET\_LENGTH ( string-expression )

Returns the number of bytes in a string.

Informix example:

octet\_length ('text')

Solution:

In SQL Server, use the DATALENGTH function.

SQL Server example:

DATALENGTH ('text')

### POW (float\_expression1, float\_expression2)

Raises its first numeric argument, the base, to the power of its second numeric argument, the exponent.

Informix example:

pow ( 10, 2 )

Solution:

In SQL Server, use the POWER function.

SQL Server example:

POWER ( 10, 2 )

### RANGE (numeric\_column)

Return the difference between the maximum and the minimum values.

Informix example:

range ( expr )

Solution:

In SQL Server, use the MIN and MAX aggregate functions.

SQL Server example:

MAX ( expr ) - MIN ( expr )

### ROOT (float\_expression1, float\_expression2)

Extracts a positive real root value, returned as a FLOAT data type, from its first numeric expression argument.

Informix example:

root ( 100, 2 )

Solution:

In SQL Server, use the POWER function.

SQL Server example:

POWER ( 100, 1/2 )

### ROUND (float\_expression1, int\_expression2)

Returns the rounded number to the specified precision value.

Informix example:

round ( 123.125, 2 )

Solution:

In SQL Server, use the ROUND function, but only for numeric and float data types.

Rounding of dates is not supported.

SQL Server example:

ROUND ( 123.125, 2 )

### RPAD (source\_string, length, pad\_string)

Returns a copy of source\_string that is right-padded to the total number of characters specified by length.

Informix example:

rpad('text', 5, '<>')

Solution:

In SQL Server, use the following CASE construction:

CASE WHEN @length>0 AND @length-LEN(@source\_string)>0 THEN

@source\_string + REPLICATE (@pad\_string, (@length-LEN(@source\_string))/LEN(@pad\_string))+

 SUBSTRING(@pad\_string, 1, (@length-LEN(@source\_string))%LEN(@pad\_string))

WHEN @length>=0 AND @length-LEN(@source\_string)<=0 THEN

 SUBSTRING(@source\_string, 1, @length)

ELSE NULL END

If a negative value for the length argument returns NULL, or if the length argument is less than the length of the source string, SUBSTRING is returned. Otherwise, the source string is right-padded by the value specified in the pad\_string argument as needed.

SQL Server example:

CASE WHEN 5>0 AND 5-LEN('text')>0 THEN

 'text' + REPLICATE ('<>', (5-LEN('text'))/LEN('<>'))+

 SUBSTRING('<>', 1, (5-LEN('text'))%LEN('<>'))

WHEN 5>=0 AND 5-len('text')<=0 THEN

 SUBSTRING('text', 1, 5)

ELSE NULL END

### RTRIM (source\_string, pad\_string)

The RTRIM function removes specified trailing pad characters from a string.

Informix example:

ltrim('>><<text>><<', '<>')

Solution:

SQL Server also has a function named RTRIM, but it can trim only spaces. To emulate this function in other cases, use the following construction:

REVERSE(SUBSTRING(REVERSE(@source\_string), PATINDEX('%[^'+@pad\_string+']%',REVERSE(@source\_string)), LEN(@source\_string)))

SQL Server example:

REVERSE(SUBSTRING(REVERSE('>><<text>><<'), PATINDEX('%[^'+'<>'+']%',REVERSE('>><<text>><<')), LEN('>><<text>><<')))

### SQLCODE

Returns the error code for the most recently executed SQL statement.

Informix example:

sqlcode

Solution:

In SQL Server, use the @@ERROR server variable.

SQL Server example:

@@ERROR

### STDEV (numeric\_column)

Return the standard deviation of a data set.

Informix example:

stdev ( expr )

Solution:

In SQL Server, use the STDEVP aggregate function.

SQL Server example:

STDEVP ( expr )

### SUBSTR ( string-expression, start [, length ] )

Returns a substring of a string.

Informix example:

substring( 'Test Message', 1, 4 )

Solution:

SQL Server also has a function called SUBSTRING, but it does not allow negative start and length. A user-defined function can be created to fully emulate Informix substring behavior.

SQL Server example:

SUBSTRING( 'Test Message', 1, 4 )

### SUBSTRING( string-expression FROM start [FOR length ] )

Returns a substring of a string.

Informix example:

substring( 'Test Message' from 1 for 4 )

Solution:

SQL Server also has a function called SUBSTRING, but it does not allow negative start and length. A user-defined function can be created to fully emulate Informix substring behavior.

SQL Server example:

SUBSTRING( 'Test Message', 1, 4 )

### TODAY

Returns the current date.

Informix example:

today

Solution:

In SQL Server, use the GETDATE function.

SQL Server example:

CAST(GETDATE() AS DATE)

### TRIM ({BOTH|LEADING|TRAILING } pad\_string FROM source\_string)

The TRIM function removes the pad characters specified by LEADING, TRAILING, or BOTH from a string.

Informix example:

trim(both '<>' from '>><<text>><<')

Solution:

In SQL Server, for leading trim, see the emulation of LTRIM, and for trailing trim, see the emulation of RTRIM. For both, apply the emulation of LTRIM and RTRIM, one after another.

### TRUNC (float\_expression1, int\_expression2)

Returns the truncated to specified precision value.

Informix example:

round ( 123.125, 2 )

Solution:

In SQL Server, use the ROUND function with 1 in the third argument, but only for numeric and float data types.

Rounding of dates is not supported.

SQL Server example:

ROUND ( 123.125, 2, 1 )

### VARIANCE (numeric\_column)

Return an estimate of the population variance, as the standard deviation squared.

Informix example:

variance ( expr )

Solution:

In SQL Server, use the VARP aggregate function.

SQL Server example:

VARP ( expr )

### WEEKDAY (date/dtime\_expr)

The WEEKDAY function takes a DATE or DATETIME argument and returns an integer in the range from 0 to 6 that represents the day of the week. Zero (0) represents Sunday, one (1) represents Monday, and so on.

Informix example:

weekday(current)

Solution:

In SQL Server, use the DATEPART function with a *datepart* of **dw** and the @@DATEFIRST server variable.

SQL Server example:

(DATEPART(dw, GETDATE())-1+@@DATEFIRST)%7

# Conclusion

This migration guide covers the differences between Informix 11 and SQL Server 2008 database platforms, and the steps necessary to convert an Informix database to SQL Server.

## About DB Best Technologies

DB Best Technologies is a leading provider of database and application migration services and custom software development. We have been focused on heterogeneous database environments (SQL Server, Oracle, Sybase, DB2, MySQL) since starting at 2002 in Silicon Valley. Today, with over 75 employees in the United States and Europe, we develop database tools and provide services to customers worldwide.

DB Best developed migration tools to automate conversion between SQL dialects. In 2005 Microsoft acquired this technology, which later became a family of SQL Server Migration Assistant (SSMA) products. We continue to develop new versions of SSMA, and support Microsoft customers who are migrating to SQL Server.

We also provide migration services covering all major steps of a typical migration project: complexity assessment, schema conversion, data migration, application conversion, testing, integration, deployment, performance tuning, training, and support.

For more details, visit us at <http://www.dbbest.com>, e-mail us at info@dbbest.com, or call 1-408-202-4567.

**For more information:**

<http://www.microsoft.com/sqlserver/>: SQL Server Web site

<http://technet.microsoft.com/en-us/sqlserver/>: SQL Server TechCenter

<http://msdn.microsoft.com/en-us/sqlserver/>: SQL Server DevCenter

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
* Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send feedback.