Oracle® SQL Developer

Supplementary Information for IBM DB2 Migrations

Release 3.0

E18460-02

February 2011

This document contains information for migrating from IBM DB2 to Oracle. It supplements the information about migration in *Oracle SQL Developer User's Guide*.

Oracle SQL Developer Supplementary Information for IBM DB2 Migrations, Release 3.0

E18460-02

Primary Author: Chuck Murray

Contents

3 Database Security

4 Offline Data Loading

Index

Preface

Oracle SQL Developer Supplementary Information for IBM DB2 Migrations describes several differences between IBM DB2 and Oracle. It also outlines how those differences are dealt with by SQL Developer during the migration process.

Audience

This guide is intended for anyone who is involved in converting an IBM DB2 database to Oracle using SQL Developer.

You should be familiar with relational database concepts and with the operating system environments under which you are running Oracle and IBM DB2.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/us/corporate/accessibility/index.html`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

For information about using Oracle SQL Developer, see *Oracle SQL Developer User's Guide* and the SQL Developer online help.

For information about installing Oracle SQL Developer, see *Oracle SQL Developer Installation Guide*.

Oracle error message documentation is only available in HTML. If you only have access to the Oracle Documentation CD, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

To download free release notes, installation documentation, white papers, or other collateral, go to the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at `http://www.oracle.com/technology/membership`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

This document provides supplementary reference information for migrating IBM DB2 UDB databases to Oracle using the Oracle SQL Developer tool. It includes information to help you plan for the migration and to solve any problems that might occur during or after the migration.

This documents supplements some DB2 migration information that is included in the SQL Developer online help and in *Oracle SQL Developer User's Guide*. For example, those sources include

- Information about third-party driver files (`db2jcc.jar` and `db2jcc_license_cu.jar`) that you must download and set as preferences in order to be able to create DB2 connections
- IBM DB2 offline capture notes, including the names of script files and the commands that you must enter to export schema information and table data

You should already know how to use the SQL Developer tool, including its migration capabilities. SQL Developer is described in *Oracle SQL Developer User's Guide* and in the online help.

The following supplementary reference information is available:

- Chapter 2, "Oracle and DB2 Compared"
- Chapter 3, "Database Security"
- Chapter 4, "Offline Data Loading"

2

Oracle and DB2 Compared

This chapter compares the MySQL database and the Oracle database. It includes the following major sections:

- Section 2.1, "Data Storage Concepts"
- Section 2.2, "Database Schema Migration"
- Section 2.3, "Data Types"

2.1 Data Storage Concepts

This section provides an overview of the data storage concepts and methods used by IBM DB2, and the similarities or differences between these methods and concepts and those used by Oracle.

2.1.1 Oracle Data Storage Concepts

An Oracle database consists of one or more tablespaces. Tablespaces provide logical storage space that links a database to the physical disks hold the data. A tablespace is created from one or more datafiles. Datafiles are files in the file system or an area of disk space specified by a device. A tablespace can be enlarged by adding more datafiles.

A basic Oracle database consists of a SYSTEM tablespace, where the Oracle data dictionary tables are stored. It can also consist of user-defined tablespaces. A tablespace is the logical storage location for database objects. For example, you can specify where a particular table or index is created in the tablespace. The size of a tablespace is determined by the amount of disk space allocated to it. Each tablespace is made up of one or more data files.

2.2 Database Schema Migration

The database schema contains the definitions of the tables, indexes, and other database-specific objects.

This section includes information about the following:

- Schema Object Similarities
- Database Schema Object Names
- Database Mappings
- Design Considerations

2.2.1 Schema Object Similarities

There are many similarities between schema objects in Oracle and IBM DB2. However, some schema objects differ between these databases. For more information about schema objects in Oracle, see *Oracle Database SQL Language Reference*. For more information about the IBM DB2 schema types that do not have a direct mapping in Oracle, see "Design Considerations"page 3.

Table 2–1 lists the similarities and differences between the schema objects in the two databases, and specifies if SQL Developer supports the migration of these schema objects.

Table 2–1 Schema Objects in Oracle and IBM DB2

IBM DB2	Oracle	Status
Check Constraint	Check Constraint	Migrated[1]
Column Default	Column Default	Migrated[2]
Database	Database	Migrated
Datalink	Binary File	Not Migrated
Foreign Key	Foreign Key	Migrated
Function	Function	Not Migrated
Group	Role	Migrated
Identity Column	Auto Increment Column	Migrated
Index	Index	Migrated
Primary Key	Primary Key	Migrated
Schema	User	Migrated[3]
Sequence	Sequence	Not Migrated
Stored Procedure	Stored Procedure	Not Migrated
Structured Data Type	Abstract Datatype	Not Migrated
Synonym Table	Alias	Migrated
Table	Table	Migrated
Table Alias	Public Synonym	Migrated
Table Space	Tablespace	Migrated
Tables containing structured data type	Table containing abstract datatype	Not Migrated
Temporary Table	Temporary Table	Not Migrated
Typed Table	Object Table	Not Migrated
Typed View	Object View	Not Migrated
UDB Package	PL/SQL Package	Not Migrated
UDB SQL Function	PL/SQL Function	Not Migrated
UDB SQL Procedure	PL/SQL Procedure	Not Migrated
UDB Trigger	PL/SQL Trigger	Not Migrated
Unique Key	Unique Key	Migrated
User	User	Migrated[4]

Table 2–1 (Cont.) Schema Objects in Oracle and IBM DB2

IBM DB2	Oracle	Status
View	View	Not Migrated
View containing structured data type	View containing abstract datatype	Not Migrated

[1] This schema object is migrated but not parsed by SQL Developer

[2] This schema object is migrated but not parsed by SQL Developer

[3] For more information about IBM DB2 schema mappings, see Chapter 4, "Offline Data Loading"

[4] SQL Developer maps user with the prefix USR_, such as USR_GUEST

2.2.2 Database Schema Object Names

Reserved words differ between Oracle and IBM DB2. Some Oracle reserved words are valid object or column names in IBM DB2. Use of reserved words as schema object names makes it impossible to use the same names across databases. SQL Developer appends an underscore and a numeric digit (*_n*) to the end of the name of an IBM DB2 object that is an Oracle reserved word.

Object names are not case sensitive in Oracle or IBM DB2. For a list of Oracle reserved words, see *Oracle Database SQL Language Reference*

2.2.3 Design Considerations

This section describes table design issues that you must consider when converting IBM DB2 databases to Oracle. It contains information about the following:

- Database Mappings
- Referential Integrity Constraints
- Table space Mappings
- Alias Mappings

2.2.3.1 Database Mappings

You can create multiple databases to group segments of related data in Oracle and IBM DB2. If you choose to migrate a number of IBM DB2 databases using SQL Developer at one time, all the IBM DB2 databases are migrated to a single, specified Oracle database.

2.2.3.2 Referential Integrity Constraints

A referential integrity constraint is the rule that governs the relationship between columns in different tables. Integrity constraints are very similar in the Oracle and IBM DB2 databases. The differences are described in this section.

IBM DB2 supports the ON DELETE and ON UPDATE clauses in referential integrity constraints. Oracle only supports the ON DELETE clause in referential integrity constraints.

Table 2–2 summarizes the delete integrity constraint mappings between Oracle and IBM DB2, and specifies if SQL Developer supports the migration of these clauses.

Table 2–2 Delete Integrity Constraint Mapping

IBM DB2	Oracle	Status
ON DELETE RESTRICT	ON DELETE NO ACTION	Migrated

Table 2–2 (Cont.) Delete Integrity Constraint Mapping

IBM DB2	Oracle	Status
ON DELETE NO ACTION	ON DELETE NO ACTION	Migrated
ON DELETE CASCADE	ON DELETE CASCADE	Migrated
ON DELETE SET NULL	ON DELETE SET NULL	Not Migrated

2.2.3.3 Table space Mappings

In IBM DB2 each table is assigned to a table space that contains the primary data for that table. However, a table may optionally store the indexes in a second table space, and large objects in a third table space. You can assign more than one table to the same table space. Each table space consists of a collection of containers, each of which is a directory either in the file system, physical file, or device, such as a hard disk.

IBM DB2 provides system managed space (SMS) and database managed space (DMS). An SMS table space uses the facilities provided by the operating system to manage physical space. IBM DB2 directly manages the physical space in a DMS table space.

Oracle uses dictionary-managed tablespaces and locally-managed tablespaces methods to keep track of their free and used space. The dictionary-managed tablespaces are extents that are managed by the data dictionary, while locally-managed tablespaces are extents that are managed by the tablespace. Oracle recommends locally-managed tablespaces for the following reasons:

- Logical management of extents avoids recursive space management operations. This can occur in dictionary-managed tablespaces if consuming or releasing space in an extent results in another operation, which consumes or releases space in a rollback segment or data dictionary table.
- Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.
- SQL Developer automatically maps the DMS and SMS table spaces to Oracle locally-managed tablespaces. You can use SQL Developer to reset this to dictionary-managed tablespaces, on a per tablespace basis.

2.2.3.4 Alias Mappings

An alias in IBM DB2 is an alternative name for a table alias or view. A table alias can be created or dropped. No authority is required to use an alias. However, access to the tables and views referred to by an alias still require the appropriate authorization. An alias can be up to 30 characters long.

SQL Developer migrates a table alias to a public synonym. All user accounts created in Oracle by SQL Developer are granted the CREATE PUBLIC SYNONYM and DROP PUBLIC SYNONYM system-level privileges. For more information about these privileges, see Chapter 4, "Offline Data Loading".

2.2.3.5 Identity Column Mappings

In IBM DB2 identity columns provide a way to automatically generate unique, sequential and recoverable values for each row in a table. Each table can have only one identity column assigned to it. The column defined as an identity column must be an arithmetic and exact data type, for example SMALLINT, INTEGER, BIGINT, or DECIMAL, where a scale of zero can be used. An identity column has the following characteristics:

- It is defined as part of a table only when you create the table. Once you create a table, you cannot alter it to add an identity column.
- It generates values for a single table automatically.
- When you define it with the GENERATED ALWAYS command, the database manager always generates the values used.

Oracle always maps identity columns to AUTO INCREMENT columns. Oracle uses sequences and triggers to emulate identity columns. To explain this further, assume that you want to create an AUTO INCREMENT column called Auto_Val on a table called MyTable in Oracle. First, you have to create a sequence as in the following example:

```
CREATE SEQUENCE MySequence
START WITH 1
INCREMENT BY 1
NO MAX VALUE;
```

In this example, the START WITH value is 1 and this value is incriminated by 1 after every INSERT operation on the table. The NO MAX VALUE specifies that this value is incriminated and is not reset at a certain point. A trigger is then required that inserts the value created by this sequence into the Auto_Val column. You create this as follows:

```
CREATE TRIGGER MyTrigger
BEFORE INSERT ON MyTable
FOR EACH ROW Begin
SELECT MySequence.NEXTVAL INTO :MyTable.Auto_Val FROM DUAL;
END;
```

SQL Developer automatically generates this solution for identity columns. The GENERATED_ALWAYS attribute of identity columns is always obeyed by the AUTO_INCREMENT migration solution. Therefore, all identity columns are generated as if they have the GENERATED_ALWAYS attribute. SQL Developer warns the user that identity columns that do not specify the GENERATED_ALWAYS attribute are migrated with this attribute.

2.3 Data Types

This section describes the differences between the data types used in IBM DB2 and Oracle databases. This section contains information about the following:

- Data Types in Oracle and IBM DB2
- Character Data Types
- Graphic Data Types
- Numeric Data Types
- Date Data Types
- String Representation of Datetime Values
- IBM DB2 Datalinks
- User-Defined Distinct Data Types
- IBM DB2 Structured Data Types

2.3.1 Data Types in Oracle and IBM DB2

Table 2–3 lists the data types available in IBM DB2 and their Oracle equivalents.

All of the IBM DB2 data types support the NULL value. The NULL value is defined as a missing, unknown, or inapplicable value. Although all data types include the NULL value, columns defined as NOT NULL cannot contain NULL values.

Table 2–3 Data Types in Oracle and IBM DB2

IBM DB2 Data Type	Description	Oracle Data type
BLOB	Variable-length binary large object string that can be up to 2GB (2,147,483,647) long. Primarily intended to hold non-traditional data, such as voice or mixed media. BLOB strings are not associated with a character set, as with FOR BIT DATA strings.	BLOB
CHAR FOR BIT DATA*(length)*	Fixed-length character string with a length between 1 and 124 inclusive. If the length is omitted, it defaults to 1. Treated as binary data.	CHAR
CHAR VARYING FOR BIT DATA*(length)*	Variable-length character string with a maximum length of 32,672 bytes. Treated as binary data.	VARCHAR2
CHAR VARYING*(length)*	Variable-length character string with a maximum length of 32,672 bytes	VARCHAR2
CHAR*(length)*	Fixed-length character string with a length between 1 and 124. If the length is omitted, it defaults to 1.	CHAR
CHARACTER FOR BIT DATA*(length)*	Fixed-length character string with a length between 1 and 124 inclusive. If the length is omitted, it defaults to 1. Treated as binary data.	CHAR
CHARACTER VARYING FOR BIT DATA*(length)*	Variable-length character string with a maximum length of 32,672 bytes. Treated as binary data.	VARCHAR2
CHARACTER VARYING*(length)*	Variable-length character string with a maximum length of 32,672 bytes	VARCHAR2
CHARACTER*(length)*	Fixed-length character string with a length between 1 and 124. If the length is omitted, it defaults to 1.	CHAR
CLOB	Variable-length character large object string that can be up to 2GB (2,147,483,647) long. A CLOB can store single-byte character strings or multibyte, character-based data. A CLOB is considered a character string.	CLOB
DBCLOB	Variable-length character string that can store up to 1,073,741,823 double-byte characters. It has a DBCS CCSID associated with it. A DBCLOB is considered a graphic string.	CLOB
GRAPHIC*(length)*	Fixed-length character string with a length between 1 and 127 inclusive. If the length is omitted, it defaults to 1.	VARCHAR2

Table 2–3 (Cont.) Data Types in Oracle and IBM DB2

IBM DB2 Data Type	Description	Oracle Data type
LONG VARCHAR*(length)*	Variable-length character string with a maximum length of 32,700 bytes	VARCHAR2
LONG VARGRAPHIC*(length)*	Variable-length graphic string with a maximum length of 16,350 double-byte characters.	CLOB
VARCHAR FOR BIT DATA*(length)*	Variable-length character string with a maximum length of 32,672 bytes. Treated as binary data.	VARCHAR2
VARCHAR*(length)*	Variable-length character string with a maximum length of 32,672 bytes	VARCHAR2
VARGRAPHIC*(length)*	Variable-length graphic string with a maximum length of 16,336 double-byte characters	VARCHAR2

2.3.2 Character Data Types

There are several types of character data types. Each character data type in IBM DB2 can be categorized further into one of the types listed in the Table 2–4. The length of a mixed-data character data type is its total number of bytes, counting two bytes for each double-byte character and one byte for each single-byte character.

A character string is a sequence of bytes. The length of the string is the number of bytes in the sequence. If the length of the string is zero, then this is called the empty string. This is different from a NULL value.

Table 2–4 Categories of character strings

Description	Type
Bit data	Data not associated with a coded character set and is never converted.
Single byte character set (SBCS) data	Data in which every character is represented by a single character set. Each SBCS data character string has an associated coded character set identifier (CCSID).
Double byte character set (DBCS)	Data in which every character is represented by two bytes. Graphic and vargraphic data is stored in this format.
Mixed data	Data that contains a mixture of characters from the SBCS and the double byte character set (DBCS).

2.3.2.1 Fixed-Length Character Strings

All values of a fixed-length character string column have the same length. The length attribute of the column determines this. The length attribute must be between 1 and 32,766.

2.3.2.2 Varying-Length Character Strings

All values of a varying-length character string column have the same maximum length. The length attribute of the column determines this. The length attribute must be between 1 and 32,740. Table 2–5 provides definitions of each character string.

Table 2–5 Character String Definitions

String	Definition
CHAR or CHARACTER	A single character.
CHAR FOR BIT DATA or CHARACTER FOR BIT DATA	A single character. Treated as binary data.
CHAR*(length)* or CHARACTER*(length)*	A fixed-length sequence of characters ranging in length between 1 and 254.
CHAR(length) FOR BIT DATA or CHARACTER(length) FOR BIT DATA	This is a fixed-length sequence of characters. Treated as binary data ranging in length between 1 and 254.
VARCHAR*(length)* or CHARACTER VARYING*(length)* or CHAR VARYING*(length)*	This is a varying-length sequence of characters ranging in length between 1 and 32,672.
VARCHAR*(length)* FOR BIT DATA or CHARACTER VARYING*(length)* FOR BIT DATA or CHAR VARYING*(length)* FOR BIT DATA	This is a varying length sequence of characters. Treated as binary data ranging in length between 1 and 32,672.

2.3.2.3 Large Object Data Types

Large object data types store data ranging in size from zero bytes to two gigabytes. There are three large object data types:

- Character Large Objects (CLOBs)
- Double-Byte Character Large Objects (DBCLOBs)
- Binary Large Objects (BLOBs)

2.3.2.3.1 Character Large Objects (CLOBs) A character string comprised of single-byte characters with an associated code page. This data type contains text-oriented information, where the amount of information could grow beyond the limits of a regular VARCHAR data type. SQL Developer supports code page conversion of the information and compatibility with the other character types.

2.3.2.3.2 Double-Byte Character Large Objects (DBCLOBs) A character string comprised of double-byte characters with an associated code page. This data type contains text-oriented information where double-byte character sets are used. This data type has a maximum length of 1,073,741,823.

2.3.2.3.3 Binary Large Objects (BLOBs) A binary string comprised of bytes with no associated code page.

2.3.3 Graphic Data Types

A graphic string is a sequence of double-byte characters. The length of the string is the number of its characters in the sequence. Like character strings, graphic strings can be empty. All values of a fixed-length graphic string column have the same length, which is determined by the length attribute of the column.

The length attribute of the column determines the maximum length that a value can have. Table 2–6 above provides a description for each graphic data type. Each graphic data type can be further categorized into either DBCS data or UCS-2 data.

2.3.3.1 DBCS Data

Two bytes represent every character in this data type. Every DBCS graphic has a Coded Character Set Identifier (CCSID) that identifies a double-byte coded character set.

2.3.3.2 UCS-2 Data

A character from the Universal Coded Character Set (UCS-2) represent every character of this data type.

Table 2–6 Graphic String Definitions

String	Definition
GRAPHIC	This is a single graphic character.
GRAPHIC(length)	This is a fixed length graphic string of up to 127 characters.
VARGRAPHIC(length)	This is a varying length graphic string of up to 16,336 characters.
LONG VARGRAPHIC	This is a varying length graphic string of up to16,350 characters.

2.3.4 Numeric Data Types

All numbers have a sign and a precision. The precision is the total number of binary or decimal digits, excluding the sign. The sign is positive if the value is zero or above. Table 2–7 describes the numeric types available in IBM DB2 and their Oracle equivalents.

Table 2–7 Numeric Types in Oracle and IBM DB2

DB2 UDB Data Type	Description	Oracle Data Type
BIGINT	An eight-byte large binary integer with a precision of 19 digits. The range of this data type is -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	NUMBER(19, 0)
DECIMAL*(precision, scale)*	A packed-decimal number. The precision is the number of digits and can range between 1 and 31. The scale is the number of digits to the right of the decimal point, and can range from 0 to the value specified for precision. You can use DECIMAL*(precision)* for DECIMAL*(precision, 0)*. You can also use DECIMAL by itself for DECIMAL(5, 0). The position of the decimal point is determined by the precision and scale (number of digits in the fractional part of the number) of the number. All values of a decimal column have the same precision and scale. The range of a decimal variable or the numbers in a decimal column is -n to +n, where the absolute value of n is the largest number that can be represented with the applicable precision and scale. The maximum range is -10^{31+1} to 10^{31-1}.	FLOAT(24)
DOUBLE	A synonym for FLOAT.	FLOAT(53)

Table 2–7 (Cont.) Numeric Types in Oracle and IBM DB2

DB2 UDB Data Type	Description	Oracle Data Type
FLOAT*(precision)*	A floating-point number. The precision is the number of digits, which can range between 1 and 53. A precision value between 1 and 24 indicates a single-precision, floating-point number. A precision value between 25 and 53 indicates a double-precision, floating point number. If you do not specify a precision value then this is a double-precision, floating-point number. A single-precision, floating-point number is a 32-bit approximation of a real number with a range of 1.17549436 x 10^{-38} to 3.40282356 x 10^{+38}. A double-precision, floating-point number is a 64-bit approximation of a real number with a range of 2.2250738585072014 x 10^{-308} to 1.7976931348623158 x 10^{308}.	FLOAT
INTEGER	A four-byte binary integer. With a precision of 10 digits. The range of this data type is –2,147,483,648 to +2,147,483,647	NUMBER(11, 0)
NUMERIC*(precision, scale)*	A zoned-decimal number. The precision is the number of digits and can range between 1 and 31. The scale is the number of digits to the right of the decimal point, and can range from 0 to the value specified for precision. You can use DECIMAL*(precision)* for DECIMAL(*precision*, 0). You can also use DECIMAL by itself for DECIMAL(5, 0).	NUMBER
REAL	A single-precision floating-point number. The number can be zero, and can range from -3.402E^{+38} to -1.175E^{-37}, or from 1.175E^{-37} to 3.402E^{+38}.	FLOAT(24)
SMALLINT	A two-byte binary integer with a precision of 5 digits. The range of this data type is -32,768 to +32,767	NUMBER(6, 0)

2.3.5 Date Data Types

The base IBM DB2 date data types available and how they are mapped to Oracle are described in Table 2–8.

Table 2–8 Date Data Types in Oracle and IBM DB2

IBM DB2 Data type	Description	Oracle Data type
DATE	This type consists of: year, month, and day. It represents a calendar date. Year can range between 0001 and 9999, month can range between 1 and 12, and day can range between 1 and n, which can be 28, 29, 30, or 31, depending on the value of month and year. The length of a DATE column is 10 bytes, which is the appropriate length for a character string representation of the value.	DATE
TIME	This type consists of: hour, minute, and second. It represents a 24-hour clock time value. Hour can range between 0 and 24, minute can range between 0 and 59, and second can range between 0 and 59. You must set minute and second to 0 if hour is set to 24. The internal representation of a time is a string of 3 bytes. Each byte is two packed-decimal digits. The first byte represents the hour, the second byte the minute, and the last byte represents the second. The length of a TIME column is eight bytes, which is the appropriate length for a character representation of the value.	DATE
TIMESTAMP	This type consists of: year, month, day, hour, minute, second, and microsecond. It represents a calendar date and 24-hour clock time value. The ranges of year, month, and day are the same as the date data type. The ranges for hour, minute, and second are the same as the time data type. Milliseconds can range between 0 and 999. The internal representation of a TIMESTAMP is a string of 10 bytes, each of which consists of two packed-decimal digits. The first four bytes represent date, the next three bytes represent time, and the last three bytes represent microseconds. The length of a TIMESTAMP column is 26 bytes, which is the appropriate length for the character string representation of the value.	TIMESTAMP

2.3.6 String Representation of Datetime Values

The values of the data types DATE, TIME and TIMESTAMP are represented by an internal format. However, these data types can also be represented by character strings. There are no constants or variables with Datetime data types. You must assign a datetime value to a character string variable, in order to retrieve it. You cannot use a LOB string or LONG VARCHAR as the string to represent a datetime value, regardless of its length.

The valid string representations for DATE values in IBM DB2 are listed in Table 2–9.

Table 2–9 Valid string representations for DATE values in IBM DB2

Format Name	Format	Example
International Standards Organization (ISO)	yyyy-mm-dd	2002-05-30
IBM USA Standard (USA)	mm/dd/yyyy	05/30/2002
IBM European Standard (EUR)	dd.mm.yyyy	30.05.2002
Japanese Industrial Standard Christian era (JIS)	yyyy-mm-dd	2002-05-30
Site-defined (LOC)	Depends on database country code	Not Applicable

The string representation of DATE begins with a digit and has a length of at least eight characters. Trailing blanks may be included, and you can omit leading zeroes from the month and day sections.

Table 2–10 lists the valid string representations for TIME values in IBM DB2.

Table 2–10 Valid string representations for TIME values in IBM DB2

Format Name	Format	Example
International Standards Organization (ISO)	hh.mm.ss	17.30.01
IBM USA Standard (USA)	hh:mm AM or PM	5:30 PM
IBM European Standard (EUR)	hh.mm.ss	17.30.01
Japanese Industrial Standard Christian era (JIS)	hh:mm:ss	17:30:01
Site-defined (LOC)	Depends on database country code	

The string representation of a TIME starts with a digit and has a length of at least four characters. You can include trailing blanks, and a leading zero may be omitted. If you omit seconds, an implicit specification of zero seconds is assumed.

The string representation of a TIMESTAMP starts with a digit and has a length of at least 16 characters. And has the form: yyyy-mm-dd-hh.mm.ss.nnnnnn. You can include trailing blanks, and you can omit leading zeroes from month. Day and hour and microseconds may be truncated or entirely omitted. If you omit any trailing zeroes in microseconds, an implicit specification of 0 is applied for the missing digits.

You can omit the minutes specification for the USA standard time string format. If minutes are omitted, an implicit specification of 00 minutes is assumed, so 5p.m. is 5:00 PM. In addition, the hour must be between 1 and 12 inclusive in the case of 00:00 AM. There is a single space before the AM or PM.

Note: The ISO changed the time format so that it is identical to the JIS standard. You should use the JIS standard if an application requires the most up-to-date ISO format.

The .ss or :ss part in the ISO, EUR and JIS standards is optional.

Date, time and timestamp strings must contain only single-byte characters and digits in IBM DB2.

2.3.7 IBM DB2 Datalinks

SQL Developer does not support the migration of IBM DB2 datalinks. Where it is necessary to migrate IBM DB2 datalinks, Oracle recommends creating an Oracle binary file (BFILE). For more information about Oracle binary files, see *Oracle9i SQL Reference*.

2.3.8 User-Defined Distinct Data Types

IBM DB2 provides the capability to define new data types based on built-in data types. These data types are called user-defined distinct data types.

SQL Developer does not support the migration of user-defined distinct data types. Any column in any table that uses a user-defined distinct type (defined in IBM DB2) is defined with the base type in the Oracle Model.

2.3.9 IBM DB2 Structured Data Types

A structured data type is a user-defined data type, and its structure defined in the database. This data type comprises a sequence of named attributes, each of which has a data type. A structured data type may also be defined as a subtype of another structured type, called its supertype. A subtype inherits all the attributes of its supertype and can have additional attributes defined.

You can use a structured type as the data type of a table or view. The names and data types of the structured type become the names and data types of the columns of the table (typed table). Rows of the typed table represent instances of the structured type.

A structured data type instance can also be stored in the database as a value in a column. To store objects in table columns, columns are defined using the structured data type.

The logical mapping for an IBM DB2 structured data type in Oracle is an abstract datatype. SQL Developer does not support the migration of structured data types.

3

Database Security

This chapter describes IBM DB2 database security features and how they are handled by SQL Developer.

3.1 Mapping IBM DB2 UDB Groups and Users

Security in IBM DB2 UDB is managed by a system containing Authority Levels and Privileges. Users can only access objects which they have the required privilege or authority for.

SQL Developer maps IBM DB2 users to Oracle users. Users are mapped to Oracle with the prefix USR_, for example USR_GUEST. IBM DB2 groups are mapped to Oracle as roles. Some system level privileges and table and column level privileges are migrated from IBM DB2 to users in Oracle.

When migrating IBM DB2 groups, SQL Developer creates an Oracle role with the same name as the IBM DB2 group. In Oracle, a role is a set of privileges that can be granted to users, or to other roles. The users belonging to an IBM DB2 group cannot be determined from the system catalog tables so, therefore, SQL Developer does not grant the migrated roles to users. It is the responsibility of the Oracle database administrator to grant the role to the migrated users. For more information about granting, dropping, and altering roles, see *Oracle Database SQL Language Reference*.

3.2 Mapping IBM DB2 UDB Authorization Levels to Oracle

IBM DB2 authorities can be categorized into two types: instance level authorities and database level authorities.

3.2.1 Instance Level Authorities

There are three instance level authorities available in IBM DB2 UDB:

- Administration Authority (SYSADM)
- System Control Authority (SYSCTRL)
- System Maintenance Authority (SYSMAINT)

All three instance level authorities are controlled by the installation platform's security system. Due to cross-platform differences in security models, SQL Developer does not migrate instance level authorities. For more information about Oracle system level authorities, see *Oracle Database Administrator's Guide*.

3.2.2 Database Level Authorities

There are six database level authorities available in IBM DB2. These authorities are available on a specific database rather than a single IBM DB2 instance. There is no direct mapping for IBM DB2 database level privileges in Oracle.

The following table outlines the privileges assigned to users and schemas that are migrated as users to Oracle by SQL Developer.

Oracle Privilege	Description
CONNECT	This privilege grants users the right to use Oracle. Users who have the CONNECT role may also create tables, views, sequences, clusters, synonyms, sessions and links to other databases.
RESOURCE	This privilege grants users right to create their own tables, sequences, procedures, triggers, indexes and clusters.
CREATE PUBLIC SYNONYM	This privilege grants users the right to create public synonyms in an Oracle database. This emulates the ability in IBM DB2 UDB for users to create aliases.
DROP PUBLIC SYNONYM	This privilege grants users the right to drop public synonyms in the Oracle database.

There are over 100 separate system privileges alone in Oracle. For more information about how to configure authorization in Oracle, see *Oracle Database Administrator's Guide*.

3.3 Mapping IBM DB2 UDB Privileges to Oracle

A database privilege grants the right to perform a specific action or operation on a specific object. In IBM DB2 privileges can be categorized into the following types:

- Table and Column Privileges
- Index Privileges
- Sequence Privileges
- Schema Privileges
- Package Privileges

3.3.1 Table and Column Privileges

The following privileges can be granted for a table and column.

3.3.1.1 CONTROL Privilege

Grants ALTER, INDEX, REFERENCES, INSERT, DELETE and UPDATE privileges on a table. This privilege also gives the right to grant any applicable privilege on a table to other users and groups. This privilege also grants the right to drop a table and to update the statistics that apply to a table.

3.3.1.2 ALTER Privilege

Grants the right to alter a tables definition, for example to add a column to the table. It also grants the right to add a comment to a table, using the COMMENT statement, and to create a trigger on a table, using the CREATE TRIGGER statement).

3.3.1.3 DELETE Privilege

Grants the right to delete rows from a table.

3.3.1.4 INDEX Privilege

Grants the right to create indexes on a table.

3.3.1.5 INSERT Privilege

Grants the right to insert rows into a table.

3.3.1.6 REFERENCE Privilege

Grants the right to create or drop foreign key constraints in other tables that reference a parent key in the current table. A REFERENCE privilege may be granted on a whole table or on columns.

3.3.1.7 SELECT Privilege

Grants the right to select data from a table using the SELECT statement and also grants the right to use table in a subquery.

3.3.1.8 UPDATE privilege

Grants the right to update rows in a table. The UPDATE privilege may be granted on a whole table or on columns.

The following tables list the privilege mappings from IBM DB2 UDB to Oracle for tables and columns respectively.

IBM DB2 UDB Table Privilege	Oracle Table Privilege
CONTROL	ALTER, DELETE, INDEX, INSERT, REFERENCES, SELECT, UPDATE
ALTER	ALTER
DELETE	DELETE
INDEX	INDEX
INSERT	INSERT
REFERENCES	REFERENCES
SELECT	SELECT
UPDATE	UPDATE

IBM DB2 UDB Column Privilege	Oracle Column Privilege
CONTROL	INSERT, REFERENCES, UPDATE
INSERT	INSERT
REFERENCES	REFERENCES
UPDATE	UPDATE

3.3.2 Index Privileges

The CONTROL privilege is the only privilege available for an index. This privilege is granted to a single user, or groups of users, and grants the user the drop privilege on an index. A user must have SYSADM or DBADM authorities to grant this privilege, however it is granted automatically to the index creator.

To drop an index in Oracle, the index must be contained in your user account or you must have the DROP ANY INDEX system privilege. In order to drop an index defined on a constraint you must disable or drop the constraint itself, before dropping the index.

However, the DROP ANY INDEX system privilege in Oracle grants the right to drop an index in any user account in the database. As grants the right to drop only a specified index, and not all indexes, this is not mapped to the index CONTROL privilege in IBM DB2 UDB. SQL Developer determines the creator of an index, and this user is granted the right to drop the index in the Oracle database.

Oracle also provides the ability to alter an index. You can rebuild or coalesce an index, alter its real and default storage characteristics, but you cannot change its column structure. To alter an index, your user account must contain the index, or you must have the ALTER ANY INDEX system privilege. Again, when SQL Developer maps indexes from IBM DB2 to Oracle, it determines the creator of the index, and this user is granted the right to alter privileges on the index.

3.3.3 Sequence Privileges

Sequence objects are not migrated by SQL Developer. Therefore the migration of the privileges that apply to them are outside the scope of this document.

3.3.4 Schema privileges

An IBM DB2 schema is a collection of named objects. Schemas provide a logical classification for objects in an IBM DB2 database and can contain tables, indexes, views, and triggers. A schema is itself an object in the database and is created using the CREATE SCHEMA statement. A schema name is used as the high order part of a two part object name, for example tab_*schema.mytable*. If the schema name is specified during the creation of an object, the object is assigned to that schema. If it is not specified, the value for the IMPLICIT_SCHEMA is used. This is the default schema name.

The following privileges can be granted for a schema:

3.3.4.1 CREATIN Privilege

This grants the right to create objects in the schema.

3.3.4.2 ALTERIN Privilege

This grants the right to alter the definition of a table in the schema, and to add a comment for any object in the schema.

3.3.4.3 DROPIN Privilege

This grants the right to drop an object in the schema.

The IBM DB2 schema object does not have a direct mapping in Oracle. An Oracle database can have multiple users, each of which has a user account. Each user account is a logical collection of database objects, such as tables and indexes. Therefore, a schema in IBM DB2 is mapped to a user in Oracle. When SQL Developer encounters an IBM DB2 schema during migration, it creates a user account in the target Oracle database using the name of the IBM DB2 schema.

The following explains how SQL Developer emulates the IBM DB2 schema in Oracle, using a table called TAB owned by a user called USR in an IBM DB2 schema called SCH:

1. SQL Developer creates a user account called SCH in the Oracle database.
2. SQL Developer creates a table called TAB in the SCH user account. The SCH user account automatically becomes the owner of that table.
3. SQL Developer creates another user account called USR in the Oracle database. USR is mapped to a user called USR in Oracle.
4. SQL Developer creates a private synonym called TAB in the USR user account. This synonym ensures that a user connected as USR can manipulate the TAB table without having to explicitly prefix the table name with SCH.

All objects that resided in the SCH schema are created in the SCH user account in Oracle. All table and column level privileges granted to the USR user in IBM DB2, for objects residing in the SCH schema, are migrated to Oracle. The USR user account in Oracle is granted the appropriate privileges on objects residing in the SCH user account.

3.3.5 Package privileges

Package objects are not migrated from IBM DB2 by SQL Developer.

4 Offline Data Loading

SQL Developer uses the IBM DB2 Export utility, and the Oracle SQL*Loader utility, to provide an offline data loading capability. This is useful for migrating large volumes of table data. You can use the IBM DB2 Export utility to extract data from an IBM DB2 database, and save it to a file in one of several supported file formats. You specify the data you want to extract using a SQL query. SQL Developer automatically generates the required scripts to enable offline data dump as well as loading of the generated data. The generated scripts support the offline data loading of all types of data, including LOB data.

This chapter includes information about the following:

- Script Directory Structure
- Unloading Data from an IBM DB2 Database
- Loading Data into an Oracle database

4.1 Script Directory Structure

All offline data export scripts are stored in target folder that you selects. To generate scripts, click **Tools**, then **Migration**, then **Script generation**, then **Generate data move scripts**.

SQL Developer creates a directory using *timestamp* in these subdirectories that represents the date and time you generated the offline data loading scripts. For example, a subdirectory called 21-08-07_17-56-16 indicates that you generated the scripts at 17:56 pm on September 21, 2007. SQL Developer creates the batch command file `unload_script.bat` or `unload_script.sh` in this subdirectory. This file contains commands used to extract data from the IBM DB2 source database, and store the data in delimited ASCII files.

The *timestamp* directory also contains SQL*Loader control files and a SQL*Loader script named `oracle_ctl.bat` or `oracle_ctl.sh` that are used to load the data into the Oracle database. Control files are contained within the Oracle folder.

The directory structure and its contents is summarized in the following table:

Directory	Description
DB2\ *timestamp*	The TIMESTAMP directory contains a subdirectory called ORACLE. It also contains a file named `unload_script.bat` or `unload_script.sh`. This file contains the commands used to export the data from the IBM DB2 database into delimited ASCII files. The exported data is automatically generated into the TIMESTAMP subdirectory. The directory also contains a file called `oracle_ctl.bat` or `oracle_ctl.sh`. This file contains the SQL*Loader commands used to load the data files, generated by `unload_script.bat` or `unload_script.sh`, into the specified Oracle database.
DB2\ *timestamp*\Oracle	This directory contains a SQL*Loader control file (.ctl) for each table from which you want to move. Any error during data move is saved as a log file here.

4.2 Unloading Data from an IBM DB2 Database

After generating the offline data loading scripts, you can use them to unload the data from the IBM DB2 database. To unload the data from the IBM DB2 database you must execute the EXPORT.BAT file. For more information about uploading data, see Section 4.2.1, "Procedure to Unload Data from an IBM DB2 Database".

The format for the export command in the `unload_script.bat` or `unload_script.bat.sh` file is as follows:

- For DB2 version 9 data export:

```
db2 export to <file name> of DEL modified by lobsinsepfiles coldel"#"
timestampformat=\"YYYY/MM/DD HH.mm.ss\" datesiso nochardel <select query>
```

- For DB2 version 8 data export:

```
db2 export to <file name> of DEL modified by coldel"#"
timestampformat=\"YYYY/MM/DD HH.mm.ss\" datesiso nochardel <select query>
```

DB2 version 9 supports LOB data in separate files, which is better for migrating large data sizes. With version 8, to support large LOB data, you must modify the oracle ctl file command and db2 command in `unload_script.bat` or `unload_script.sh`.

The following table provides a description of the commands used in the `unload_script.bat` or `unload_script.sh` file:

Parameter	Description
`Db2`	This parameter invokes the IBM DB2 command line environment.
`export to`	This parameter is the call to the IBM DB2 export utility.
`DB2\`*`TABLE_NAME`*`.DAT`	This parameter is the name of the file that stores the data extracted from the IBM DB2 table where *TABLE_NAME* is the name of the table. All of the data files are generated in the DB2 directory. This enables the file to locate the data files when they are required.

Parameter	Description
`of del`	This parameter instructs the IBM DB2 Export utility to insert the data into delimited ASCII format. In this format, column delimiters separate column data, and row delimiters separate rows. This is the file format supported by the Oracle SQL*Loader utility.
`lobsinsepfiles`	This parameter specifies whether to create a separate file for each lob object in a table row or to use a single delimited file for all table data.
`nochardel`	This parameter instructs the IBM DB2 Export utility *not* to enclose character-based data in single quotes.
`coldel#`	This parameter instructs the IBM DB2 Export utility to use the "#" (hash or pound-sign) character as the column delimiter.
`<select query>`	This parameter is the SQL command that instructs the IBM DB2 Export utility to load all column data from the specified table residing in the specified schema.

The table data is exported to files with names in the format <*catalog*>.<*schema*>.<*table*>`.dat`. The format of file is as follows: `data1#<COL_DEL>` `#data2#<COL_DEL>`...`<ROW_DEL>` where `COL_DEL` and `ROW_DEL` come from migration offline preference settings.

Before you execute the DB2 data dump script, you must log in by entering a command in the following format:

```
db2 connect to <catalog> user <user name> using <password>
```

You can then execute the script using the logged connection session.

4.2.1 Procedure to Unload Data from an IBM DB2 Database

To unload data from the IBM DB2 database using the `unload_script.bat` or `unload_script.sh` file, you must perform the following steps:

1. Activate the IBM DB2 Command Line Processor (CLP) Window by using the following command in a Microsoft DOS window command prompt:

   ```
   db2cmd
   ```

2. Activate the required database using the following command in the CLP window:

   ```
   db2 acitvate database database_name
   ```

3. Connect to the database using the following command in the CLP window:

   ```
   db2 connect to database_name user user_name using password
   ```

4. Execute the `unload_script.bat` or `unload_script.sh` file by entering the command in the CLP window. For example:

   ```
   unload_script.bat
   ```

   The preceding example unloads data for the tables specified in the `unload_script.bat` file.

   The IBM DB2 Export utility generates all data files in the DB2 directory for subsequent processing by the `oracle_ctl.bat` or `oracle_ctl.sh` file.

Note: To successfully execute the IBM DB2 Export utility you must have SYSADM or DBADM authority, or CONTROL or SELECT privileges, for each table you want to export.

4.3 Loading Data into an Oracle database

After generating the data files you must then load the data from these files into the Oracle database. To load the data into the Oracle database you must execute the `oracle_ctl.bat` or `oracle_ctl.sh` file.

The format for the SQL*Loader command in the `oracle_ctl.bat` or `oracle_ctl.sh` file for a table is as follows:

```
sqlldr %1/%2 control=Oracle\TABLE_NAME.ctl log=_TABLE_NAME.log
```

where `%1/%2` represents the username/password supplied as command-line arguments.

The following table provides a description of each of the commands and parameters used when executing the SQL*Loader utility in the `oracle_ctl.bat` or `oracle_ctl.sh` file:

Parameter	Description
`sqlldr`	This parameter of the command invokes the Oracle SQL*Loader utility.
`%1/%2`	This parameter is the user name and password of the user you want to connect to the Oracle database as. These are passed as command line arguments from the command window. For example: `oracle_ctl.bat "user" "password"`
`control=DB2\TABLE_NAME.ctl`	This parameter specifies the SQL*loader control file you want to use during the data move. The name of the control file is the same as the table name. The control file contains all the details about the data to be loaded, including the format of any date, time, or timestamp data, the column delimiter being used and the identification of any LOB columns.
`log=TABLE_NAME.log`	This parameter specifies the name of the log file generated by the SQL*loader utility. The name of the log file is the same as the table name.

4.3.1 Procedure to Unload Data from the IBM DB2 Database

To load the data from the data files into the Oracle database you must execute the `SQL_LOAD_SCRIPT.BAT` file to load the data by entering the following command:

- Windows: `oracle_ctl.bat`
- Linux: `oracle_ctl.sh`

Note: The `oracle_ctl.bat` or `oracle_ctl.sh` file does not have to be executed from within the CLP window. You can execute the file from a command prompt.

4.3.1.1 SQL*Loader Control File

The format for the SQL*Loader control file for a table is as follows:

```
load data
infile 'SAMPLE.EMP_PHOTO.dat'
 "str '<EORD>\r\n'"
into table SAMPLE.EMP_PHOTO
fields terminated by '#<EOFD>#'
trailing nullcols
(
EMPNO CHAR(255),
PHOTO_FORMAT ,
PICTURE_ref  FILLER,
PICTURE LOBFILE(PICTURE_ref) TERMINATED BY EOF
)
```

On Linux systems, in the preceding example replace `"str '<EORD>\r\n'"` (the row data separator) with `"str '<EORD>\r\n'"`.

The following table provides a description of each of the commands and parameters used in the SQL*Loader control file:

Parameter	Description
`Load data`	This parameter tells SQL*loader that this is the beginning of a new data load.
`infile 'TABLE_NAME.dat`	This parameter specifies the name of a data file containing data that you want to load.
`into table TABLE_NAME`	This parameter specifies the name of the table where the data is loaded.
`fields terminated by '#<EOFD>#'`	This parameter specifies that each field is terminated by the `'#<EOFD>#'` characters.
`FIELD_NAME _ref FILLER`	This parameter specifies an alias name for the LOB data file that contains the LOB data. The SQL*Loader utility retrieves the name of the LOB data file from the main data file during the data loading operation. This parameter only applies to LOB data columns.
`FIELD_NAME LOBFILE (lobFilename) TERMINATED BY EOF)`	This parameter specifies that the LOB data file for the specified field be terminated by the EOF (End of File) marker. This parameter only applies to LOB columns.

For more information about the SQL*Loader utility, see *Oracle Database Utilities*.

Index