

# **Daffodil DB**

## **System Guide**

**Version 4.1**

**March 2005**

Copyright © Daffodil Software Limited  
Sco 42,3<sup>rd</sup> Floor  
Old Judicial Complex, Civil lines  
Gurgaon - 122001  
Haryana, India.  
[www.daffodildb.com](http://www.daffodildb.com)

All rights reserved. Daffodil DB™ is a registered trademark of Daffodil Software Limited. Java™ is a registered trademark of Sun Microsystems, Inc. All other brand and product names are trademarks of their respective companies.

## **Table of Contents**

Preface	
Purpose	05
Target Audience	05
Related Documentation	06
Daffodil DB Editions	
Embedded Edition	07
Server Edition	07
Standards Compliance	08
Daffodil DB Architecture	
System Architecture	09
Deployment Architecture	11
Embedded Architecture	12
Traditional Client/Server Architecture	13
Multi-Tier Internet Architecture	14
Physical File Architecture	15
Tuning Daffodil DB for Performance	
Database Initial Size and Database Increment Size Factor	16
Multiple Data File support	17
Fetch Size	17
Prepared Statements	17
Data Type for a Column	18
Appropriate getXXX and setXXX Methods	18
Auto Commit	19
Connection Pooling	20
Java Heap Size	20
Daffodil DB run-time options	21
Using Daffodil DB Server in a Multi-user environment	22
MVCC Architecture	22
Locking Architecture	23
Transaction Support and Concurrency Levels	23
Transaction Isolation Levels supported by Daffodil DB	25
Multiple Connections and Multiple Threading	26

Backing Up and Restoring Daffodil DB Database*	
Back Up a Database	27
Restoring from a back up	27
Daffodil DB Security	
User Authentication	28
User Authorization	28
Daffodil DB Cryptography*	
Database Encryption	30
Encryption Algorithms supported by Daffodil DB	31
Setting Database Encryption in Daffodil DB	33
Daffodil DB Distributed Transaction Support*	35
Daffodil DB Unique Features	
ResultSet Listener Support	36
Session Serializable	37
Session Sharing	38
End Note	39
Sign Up for Support	40
We Need Feedback!	41
Appendix A: System Tables	43
Appendix B: Error Messages	91

\* Features that are not supported in One\$DB

## **Preface**

### **Purpose**

Daffodil DB System Guide describes how to start, shut-down and configure Daffodil DB RDBMS. It provides information on Daffodil DB architecture and also provides the information that the server administrator might need to keep Daffodil DB running with high performance and reliability in a server framework or a multi-user application server. Apart from all the above, this guide describes the standards on which Daffodil DB had been built, transaction capabilities and some of the unique features supported by Daffodil DB.

### **Target Audience**

This guide is targeted towards the Java developing community. Since Daffodil DB is an SQL-99 compliant, 100% pure Java RDBMS, this guide assumes that that you are familiar with the following concepts:

- Basic knowledge of JDBC (Java Database Connectivity).
- Basic knowledge of SQL (Structured Query Language).
- Basic knowledge of Database Concepts.
- Basic knowledge of Java Programming Language.

It is also assumed that the reader had already gone through “[Getting Started with Daffodil DB guide](#)”.

## Related Documentation

This Getting Started Guide for Daffodil DB gave a straight-line introduction to Daffodil DB Embedded and Server editions. For advanced topics, please see our related documentation set listed below.

<a href="#">Daffodil DB Getting Started Guide</a>	Designed to help new and intermediate Daffodil DB users navigate and perform common tasks like How to start and stop Daffodil DB, Understanding key variables used by Daffodil DB, User documentation bundled with Daffodil DB. Also briefly describes Daffodil DB Editions and Tools
<a href="#">Daffodil DB SQL Reference Guide</a>	Covers all the SQL-99 features supported by Daffodil DB. This ready reference tool describes in detail the syntax and semantics of SQL language statements and elements for Daffodil DB. It explains how to use SQL with Daffodil DB and how to perform various database operations on Daffodil DB such as creating tables or indexes, managing transactions and sessions, Daffodil DB security features etc.
<a href="#">Daffodil DB JDBC Reference Guide</a>	Explains how to use Daffodil DB and JDBC technology to develop applications. It describes the basic Daffodil DB and JDBC concepts like JDBC 3.0 features supported by Daffodil DB, how to create and access Daffodil DB databases through JDBC API, Daffodil DB support for JDBC and JTA and how to use Daffodil DB in a Distributed Transaction Processing environment.
<a href="#">Daffodil DB Tools Guide</a>	Explains how to use Daffodil DB Browser with Embedded as well as Server versions of Daffodil DB. Describes how to perform various database operations on Daffodil DB using Daffodil DB Browser such as creating a database, creating database objects, manipulating data, creating triggers etc.

## **Daffodil DB Editions**

Daffodil DB is available in two editions – Embedded Edition and Server Edition.

### **Daffodil DB Embedded Edition**

Daffodil DB Embedded Edition is a 100% Java Relational Database Management System suitable for multi-threaded applications requiring an embedded database. Daffodil DB Embedded Edition is a multiple connection RDBMS designed specifically for embedding it inside Java based applications. Daffodil DB fully supports multiple connections within the same JVM as well as multiple threads performing operations concurrently in each connection. Embedded in a single-user Java application, Daffodil DB can be practically made invisible to the user, since it requires no administration and runs on the same Java virtual machine (JVM) as does the application.

### **Daffodil DB Server Edition**

Daffodil DB Server Edition, built on Internet standards is a 100% Java Relational Database Management System (RDBMS) that provides robust relational data management for networked applications. Networked applications can use Daffodil DB Server Edition to provide ubiquitous data access from anywhere using standard Structured Query Language (SQL) and Java Database Connectivity (JDBC) interfaces. Daffodil DB Server can be placed on an Intranet with clients connecting to it via TCP/IP connections. Daffodil DB Server can also be integrated with an Application Server, or a Web Server, to provide database connectivity over the Internet.

## **Standards Compliance**

Daffodil DB has been built in compliance with the latest industry standards viz. SQL-99 and JDBC 3.0.

### **SQL- 99**

Daffodil DB supports almost all constructs of SQL-99 including triggers, updatable views, table level and column level constraints, schema, domain, user, roles, privileges, batch operations, join, aggregate functions, rich in-built function library, sub-queries, intersection and union, all isolation levels, save-point, rich set of SQL data types, stored procedures and distributed transactions (XA).

*Note: - For complete details on Daffodil DB SQL 99 support, refer “[Daffodil DB SQL Reference Guide](#)”.*

### **JDBC 3.0**

Daffodil DB JDBC Driver (Type 4) provides complete support for JDBC (Java Database Connectivity) 3.0 features including optional features like save-points, rowset, XA, connecting pooling, batch operations, updatable resultsets for all types of select statements, partial fetching in resultsets etc. Daffodil DB JDBC 3.0 Driver has been certified for J2EE applications by Sun Microsystems.

*Note: - For complete details on Daffodil DB JDBC 3.0 support, refer “[Daffodil DB JDBC Reference Guide](#)”.*



## Daffodil DB Architecture

### System Architecture

A Daffodil DB instance is associated with a physical directory, *DAFFODILDB\_HOME*, which contains zero or more databases. *DAFFODILDB\_HOME* contains one subdirectory for each Daffodil DB database, which has the name same as that of the database. *DAFFODILDB\_HOME* for a Daffodil DB instance determines which databases are available in the system and where the new databases are created.

A Daffodil DB instance can be associated with one and only one *DAFFODILDB\_HOME*; hence, Daffodil DB instance can not have databases with subdirectory in different directories.

### Defining DAFFODILDB\_HOME

*DAFFODILDB\_HOME* can be defined in any one of the following manner:

- **daffodilDB\_home** (system property) – to be passed at the time Daffodil DB server is started.
- **daffodilDB\_home** (environment variable)
- **<user.Home>/DaffodilDB** (system property)

Only one of the above variables is used to define *DAFFODILDB\_HOME*. They are checked for in the order they are mentioned above i.e. if one variable is defined, then the others are ignored.

For e.g. If *daffodilDB\_home* system property has been defined, then *daffodilDB\_home* environment variable and *user.Home* system property are ignored.

If neither of *daffodilDB\_home* system property nor *daffodilDB\_home* environment variables is defined, then *user.Home/DaffodilDB* is used as *DAFFODILDB\_HOME*. If DaffodilDB folder does not exist in *user.Home*, then a folder with the name **Daffodil DB** is created in *user.Home* and used as *DAFFODILDB\_HOME* by Daffodil DB instance.

If you specify a directory that does not exist as *DAFFODILDB\_HOME*, Daffodil DB automatically creates the new directory without any databases in it.

### Multiple Daffodil DB Systems on one machine

You could potentially have two instances of Daffodil DB running on the same machine simultaneously, but each instance must use different *DAFFODILDB\_HOME* i.e. two separate instances of Daffodil DB should not access databases within the same directory.

For example, in an embedded environment, an application that accesses Daffodil DB databases starts up the local JDBC driver, which in turn starts up an instance of Daffodil DB. For an embedded system, do not run Daffodil DB Browser and a JDBC application OR two JDBC applications in the same system or on the same database, at the same time as it can lead to the corruption of databases.

In an embedded system, more than one Daffodil DB embedded applications can be used at the same time, but every application should use different *DAFFODILDB\_HOME*.

In a client/server environment, you do not have to worry about two client applications corrupting a database. In a client/server environment, a client application starts up the remote JDBC driver and connects to an already running server. The server framework is designed to allow multiple applications to connect to the same database.

However, even in client/server environment, if you have more than one Daffodil DB servers running on the same machine (listening on different ports), then they should also use different *DAFFODILDB\_HOME*.

## **Deployment Architecture**

Daffodil DB is a relational database management engine implemented as a Java class library. Applications access data managed by the Daffodil DB engine via JDBC (Java Database Connectivity) API. Daffodil DB can be included in any java application in the form of a jar file. Daffodil DB can be configured to run in embedded mode (same JVM) or server mode (over the network) based on the requirements of the application. A Daffodil DB server can manage multiple databases of Daffodil DB.

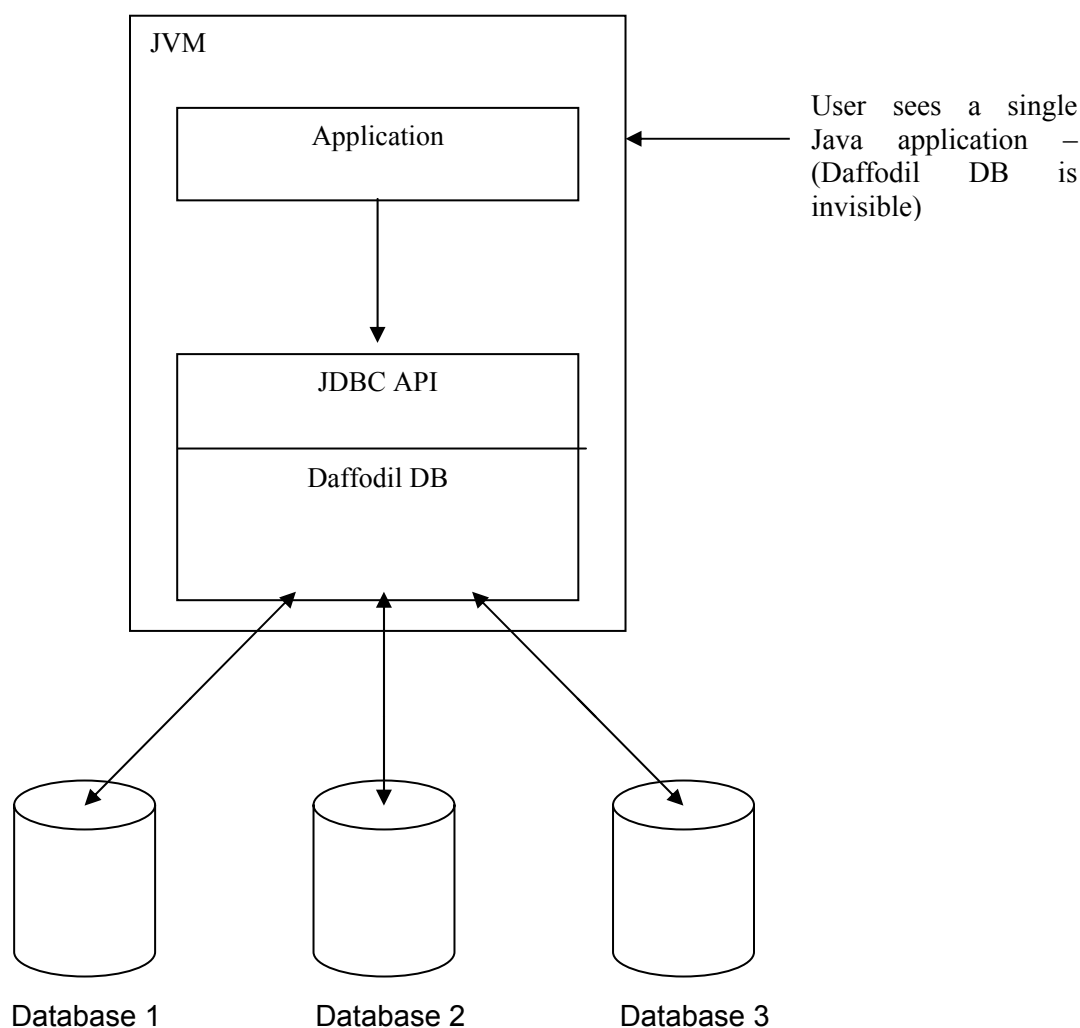
Daffodil DB can be used in any of the following architectures:

- Embedded Architecture
- Traditional Client/Server Architecture
- Multi-tier Internet Architecture (with a web server and/or application server)

## **Embedded Architecture**

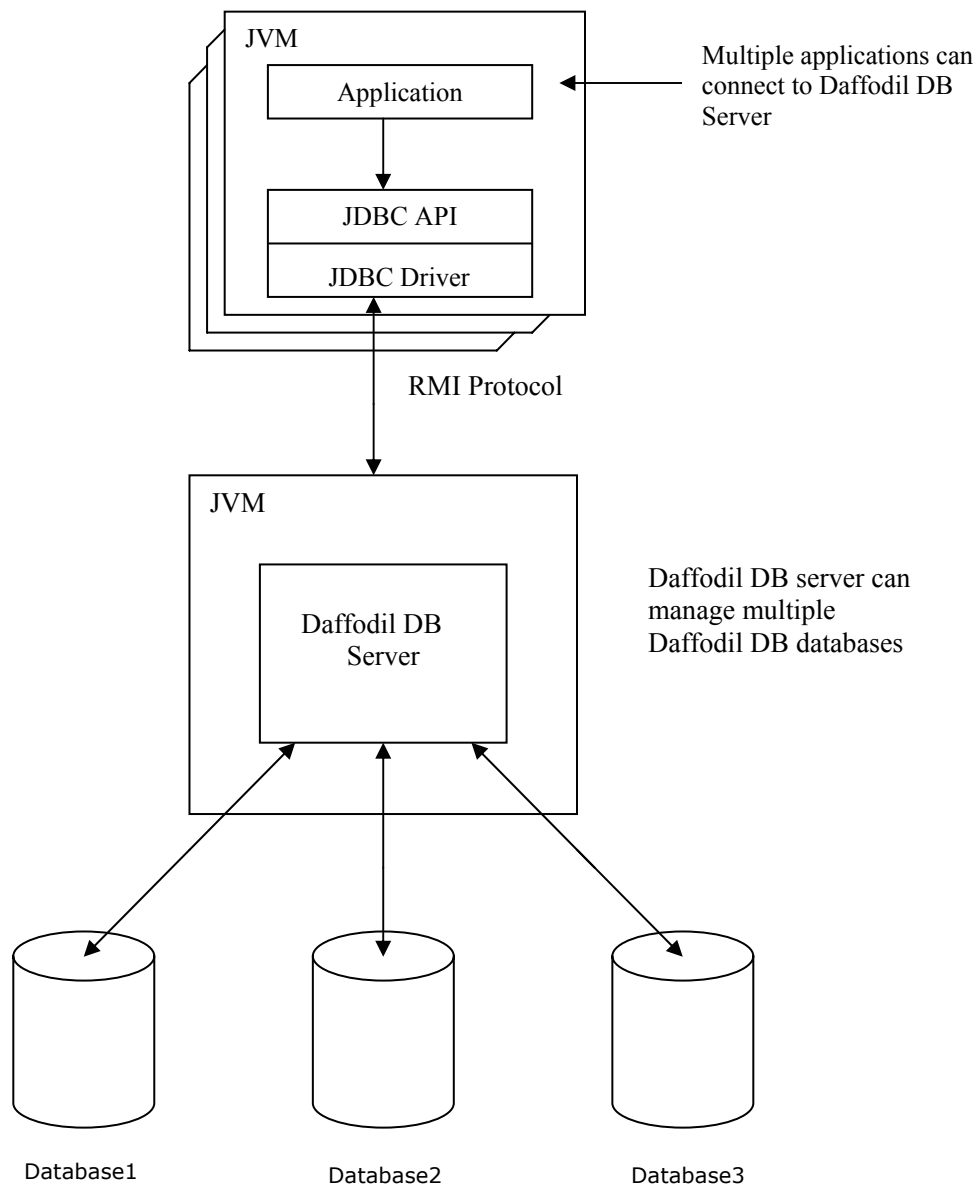
In an embedded environment, Daffodil DB is part of the java application and database engine runs in the same JVM (Java Virtual Machine) as does the application. Even in embedded architecture, application uses JDBC API to access the database, but the embedded JDBC driver simply transfers data to and from the database engine without involving need for any kind of network communication. Daffodil DB embedded edition supports multiple connections from the application. To connect, an application makes a call to the local Daffodil DB JDBC driver. JDBC driver automatically starts the embedded Daffodil DB server. The calling application is responsible for shutting down the embedded Daffodil DB database software.

When a Daffodil DB database is embedded in a Java application, the database is dedicated to that single application. If you want to deploy more than one copy of the application, where each application has its own copy of the Daffodil DB embedded edition, then a separate deployment license of Daffodil DB Embedded edition will be required for every such copy.



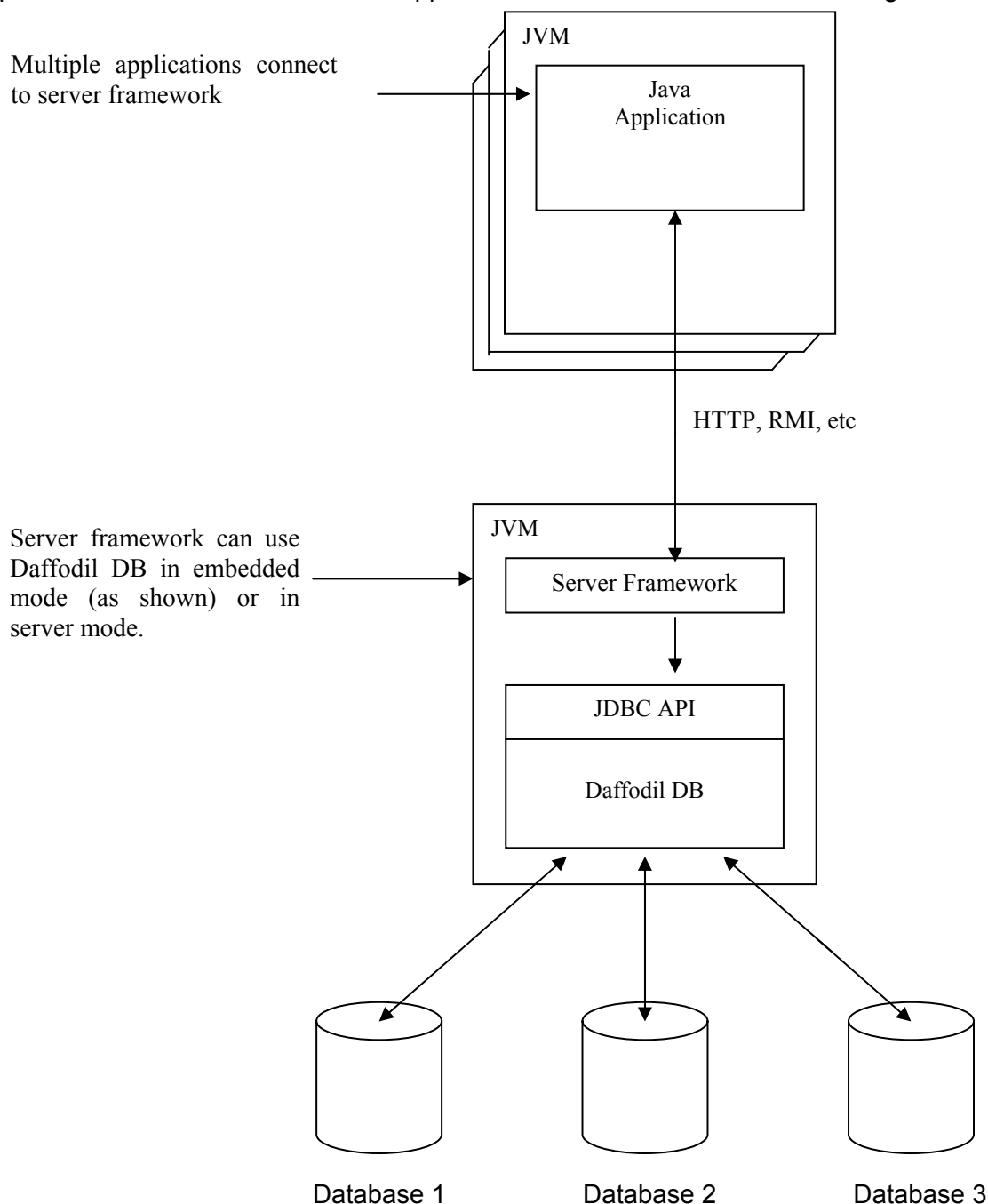
### Traditional Client/Server Architecture

In traditional client/server architecture, multiple applications can connect to Daffodil DB server directly. Daffodil DB server runs on a separate machine and the applications directly connect to the server. One Daffodil DB server can manage multiple Daffodil DB databases. Client applications connect to Daffodil DB Server using *Java RMI* protocol.



## **Multi-Tier Internet Architecture**

In a multi-tier architecture, Daffodil DB can be embedded inside a server framework. (A server framework is simply a server process which accepts and processes network communications e.g. a web-server and an application server etc). Daffodil DB Server can also be integrated with an Application Server, or a Web Server, to provide database capability over the Internet. Daffodil DB server can run in the same process as does the server framework or it can run on a separate process in the same machine or in a separate machine. Daffodil DB also supports Distributed Transaction Processing.



### **Physical File Architecture**

Daffodil DB database is stored in files that live in a directory of the same name as the database in the *Daffodil DB Home* Directory.

Daffodil DB Database contains physical files to store information on the disk and logical objects to access and control the information.

Different types of Physical files are Persistent Data Files, Recovery Files, Transient Data Files and Page Swap Files.

For detailed information about physical files and logical objects (database schema, privileges etc.), please refer to our document on [Daffodil DB Architecture](#)

### **Daffodil DB Server Instance**

A server instance is responsible for storing, retrieving and controlling information in different databases. A server instance controls multiple databases concurrently. The server consists of cache managers, parser, sessions and back-up manager.

For detailed information about Daffodil DB Server instance, please refer to our document on [Daffodil DB Architecture](#)

## **Tuning Daffodil DB for Performance**

Daffodil DB uses an in-built cost based optimizer for executing all types of queries. The optimizer determines the cost of scanning each row of the table. It looks at numerous possible query plans and determines as to which one is the most optimal and least expensive for a given query expression.

Using query optimizer, Daffodil DB generates highly optimal execution plan for any possible query. However, a user can use the following information to fine tune Daffodil DB application for better performance.

### **Database Initial size and Database Increment Size Factor**

For large databases, or databases which are expected to grow in size, the Database Initial Size and Database Increment Size Factor (which are specified at the time new Daffodil DB database is created) should be carefully chosen.

The default values used by Daffodil DB (initial size of 5 MB, and 20% of additional space every time when more space is required) are good enough for handling smaller databases. However, for big database, where the database size is expected to grow beyond 50 MB, both the above mentioned parameters must have a higher value.

For e.g. a database which is expected to grow to a size of around 300 MB, the Initial Size should be 200 MB and Database Increment Size Factor should be around 50%.

*Thumb rule for optimizing the value of these two parameters* - More the number of times physical space is added to the database, more fragmented is the physical database file and hence, slower is the database performance.



### **Multiple Data File support**

Every operating system has some limits in terms of file size, hence in order to support terabytes of data; Daffodil DB uses multiple data files for a single database. But to use this feature you need to provide couple of parameters while creating the database.

*Note: -For details on how to configure these parameters, refer [Daffodil DB JDBC Reference Guide](#).*

### **Fetch Size**

By knowing the number of rows of data you need to retrieve or fetch from the database, you can use [Statement.setFetchSize](#) (for each result set produced by the statement) or [ResultSet.setFetchSize](#) to configure this parameter for minimizing network traffic and improving database performance as well.

### **Prepared Statements**

Unlike static JDBC statement, dynamic or prepared statements are compiled only once, regardless of the number of times they are used. Use dynamic JDBC statement, whenever you need multiple executions of a particular SQL statement that has changing values associated with it. If the number of parameters changes from one insert to the next, call [PreparedStatement.clearParameters](#) () before setting parameters for PreparedStatement.

By using prepared statements instead of statements unnecessary compilation can be avoided, which will save a lot of time. In general, any query that you will use more than once should be a prepared statement. As a thumb rule, use prepared statements, whenever possible to increase the performance of the system.

### **Data type for a Column**

Daffodil DB treats any fixed-length column allowing null values, as fixed-length. Therefore, a **char** column that allows null values is treated as a fixed-length **char** column. As a result, the same data takes more disk space to store and can require more I/O and other processing operations in Daffodil DB compared to some other Database Servers. To get optimized performance in this scenario, where you expect lot of null values, it is recommended to declare that column as variable-length column rather than fixed-length column.

However, it's slower to read and process a varchar column as compared to a char column of the same size. Therefore, in cases, where value for a column does not vary much, it would be better to use a **char** datatype for that column in comparison to **varchar** datatype in terms of efficiency.

### **Appropriate getXXX and setXXX Methods**

JDBC is very flexible. It lets you use [\*java.sql.ResultSet.getFloat\*](#) to retrieve an integer data type, [\*java.sql.ResultSet.getObject\*](#) to retrieve any data type, and so on. ([\*java.sql.ResultSet\*](#) and [\*java.sql.CallableStatement\*](#) provide getXXX methods and [\*java.sql.PreparedStatement\*](#) and [\*java.sql.CallableStatement\*](#) provide setXXX methods). This flexibility is convenient but expensive in terms of performance.

For example, it is possible to retrieve any of the basic SQL types with *getString ()* method. Getting all values with *getString ()* can be very convenient, but it also has its limitations. For instance, if it is used to retrieve a numeric type, *getString ()* will convert the numeric value to a Java String object, and the value will have to be converted back to a numeric type before it can be operated on as a number.

For performance reasons, use the recommended getXXX method, when retrieving values, and use the recommended setXXX method, when setting values for parameters.

### Auto-commit

One way to improve insert and update performance is **to turn auto-commit off** (it is “on” by default) and committing the entire batch of insert/update operations in one go. Using this strategy, there will be less overhead for index/transaction maintenance and all transaction data will be flushed to the disk in a single disk write.

A new connection to a Daffodil DB database is in auto-commit mode by default, as specified by the JDBC standard. Auto-commit mode means that when a statement is completed, the method `commit ()` is called on that statement automatically. Autocommit in effect makes every SQL statement a transaction. The commit takes place whenever one statement completes or the next statement is executed, whatever occurs first.

Once auto-commit mode is disabled, no SQL statements will be committed until you call the method `commit ()` explicitly. All statements executed after the previous call to the method `commit ()` will be included in the current transaction and will be committed together as a unit. When auto-commit is disabled, you use a Connection object's `commit ()` and `rollback ()` methods to commit or roll back a transaction. The `commit ()` method makes permanent changes resulting from the transaction and releases locks. The `rollback ()` method undoes all the changes resulting from the transaction and releases locks. A transaction encompasses all the SQL statements executed against a single Connection object since the last commit or rollback. You do not need to explicitly begin a transaction. You can implicitly end one transaction and begin the new one after disabling auto-commit, changing the isolation level, or after calling commit or rollback. Committing or rolling back a transaction closes all open ResultSet objects and currently executing Statements in that Connection. It also releases any database locks held by the Connection at that point of time, whether or not these objects were created in different threads.

You can disable auto-commit with the Connection class's `setAutoCommit ()` method.

*Note: - For more details on how to use transactions with auto-commit off, refer [Daffodil DB JDBC Reference Guide](#).*

### **Connection Pooling**

In the basic DataSource implementation, there is a 1:1 correspondence between the client's Connection object and the physical database connection. When the Connection object is closed, the physical connection is dropped as well. Thus, the overhead of opening, initializing, and closing of the physical connection is incurred for each client session.

A connection pool solves this problem of overhead by maintaining a cache of physical database connections that can be reused across client sessions. Connection pooling greatly improves performance and scalability, particularly in a three-tier environment, where multiple clients can share smaller number of physical database connections.

### **Java Heap Size**

The Java heap tends to resist growing beyond its initial size, forcing frequent garbage collection with an even-smaller amount of free heap. Use the `-Xms` option to specify a larger initial heap size, while starting Daffodil DB Server.

In a multi-user environment, use a machine with a lot of memory and allocate as much memory as you can for the JVM to use. Use the `-Xmx` option to specify higher value for maximum java heap size. Daffodil DB can run in a small amount of memory (minimum required is 8 MB), but more the memory given, the faster it runs.

If you do not allocate memory using `-Xmx`, the JVM typically limits itself to 64 MB.

**Daffodil DB run-time options**

By providing couple of parameters in [daffodildb.ini](#) file, which exists or is created at *daffodilDB\_home*, user can control following performance activities:

**Logging turns on/off**

During commit process, in order to prevent database from being corrupted Daffodil DB uses a technique called logging. To save the cost of logging user can turn it off explicitly, as and when it requires.

[Powerfile.<dbname>=false](#) (default value is true)

**Read / Write clusters limit**

User can control the maximum number of read or write clusters in the memory.

[Clusters.read.<dbname>=400](#) (default value is 200)

[Clusters.write.<dbname>=400](#) (default value is 200)

## **Using Daffodil DB Server in a multi-user environment**

Daffodil DB Server provides robust relational data management capabilities for multi-user network applications. Multi-user applications can use Daffodil DB Server to provide ubiquitous data access from anywhere using standard SQL and JDBC interfaces. Daffodil DB has been designed for efficient and reliable execution in multi-user applications ensuring Data consistency and high performance even in high contention Transaction Processing applications.

If more than one client application tries to modify the same data, when Daffodil DB is running in server mode, the client who manages to get the data first obtains the lock on the data (either specific rows or the entire table). The second application has to wait until the first application commits or rollbacks the transaction and releases the lock. If two applications are only querying and not modifying data, they can both access the same data at the same time as there are no locking issues because of the MVCC architecture.

### **MVCC Architecture**

Unlike most other database systems, Daffodil DB does not use locks for concurrency control. Instead, it maintains data consistency by using multi-version model. When a database is queried, each transaction sees a snapshot of the data (a database version) as it was when the transaction/session started based on the isolation level of transaction, regardless of the current state of the underlying data. This protects the transaction from viewing inconsistent data that could be caused by other concurrent transaction updates on the same data rows, providing transaction isolation for each database session/transaction. MVCC architecture delivers a high degree of flexibility for designing applications. It allows many reads to happen in parallel with write operations while ensuring that the data is clean and transactionally consistent.

## Locking Architecture

To support multiple users and multiple transactions, Daffodil DB uses row-level locking or table-level locking automatically depending on the operation. The following types of locks are used, irrespective of the isolation level of the transaction, for different database operations in Daffodil DB:

- DDL operations - table-level locking
- DML operations - row-level locking
- DQL operations - no locking

The multi-version data model allows Daffodil DB server to avoid all inter-transactional table/row locking and deadlock issues. No tables or rows can be locked between concurrent transactions.

## Transaction Support and Concurrency Levels

The ANSI/ISO SQL standard (SQL-99) defines four levels of transaction isolation with differing degrees of impact on transaction processing throughput. These isolation levels are defined in terms of three phenomenons which are to be prevented while concurrently executing transactions.

They are:

- **Dirty Read**

A transaction reads data that has been written by another transaction, but has not been committed yet.

- **Non Repeatable (fuzzy) Read**

A transaction re-reads data it had previously read and finds if another committed transaction had modified or deleted the data.

- **Phantom Read**

A transaction re-executes a query returning a set of rows that satisfies a search condition and finds if another committed transaction had inserted additional rows, which satisfy the condition.

SQL-99 defines isolation level in terms of the phenomenon that a transaction running at a particular isolation level is permitted to experience.

Isolation Level	Dirty Read	Non Repeatable Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible

Daffodil DB automatically provides read consistency to a query such that all the data seen by a query comes from a single point of time (**statement-level read consistency**). Daffodil DB can also provide read consistency to all of the queries in a transaction (**transaction-level read consistency**) or all the queries in a session, which comprises of multiple transactions (**session level read consistency**).

Apart from supporting all the isolation levels described in SQL-99 standards, Daffodil DB also supports another isolation level, which is unique to Daffodil DB and is defined as “**Session Serializable**”.



## **Transaction Isolation Levels supported by Daffodil DB**

### **Read committed**

This is the default transaction isolation level. Each query executed by a transaction sees only data that was committed before the query (not before the transaction) was fired. A Daffodil DB query never reads dirty (uncommitted) data as Daffodil DB does not prevent other transactions from modifying the data read by any query and that data can be changed by other transactions while two executions of the query takes place. Thus, a transaction that executes a given query twice can experience both non-repeatable read as well as phantoms.

### **Repeatable Read**

In this mode, only committed rows are retrieved (as in the READ\_COMMITTED) but without considering the problem involved in the READ\_COMMITTED isolation level i.e. if same row is retrieved again in the same transaction, then exactly same value is retrieved. However, if a new row is added by another transaction and committed the insert (also delete or update) then retrieval for the same select statement second time may include the newly inserted (also deleted or updated) row, resulting in “phantom read”.

### **Read Uncommitted**

It is also known as ‘dirty read.’ In this mode, all rows, including uncommitted rows of other concurrent transactions are retrieved. For example, if a transaction T1 performs one row insert, transaction T2 retrieves that row even before T1 ends.

### **Serializable**

Serializable transactions see only those changes those were committed at the time when the transaction began, and changes made by the transaction itself through INSERT, UPDATE, and DELETE statements. Serializable transactions do not experience non-repeatable reads or phantoms.

### **Session Serializable**

Serializable Session isolation level applies serialization to a user session\*\* just like isolation level of “Serializable” applies serializability to one particular transaction. Transactions with isolation level “session serializable” can see only those changes that were committed when the session (and not the transaction) began, and also, the changes made by the transactions in the session itself through INSERT, UPDATE and DELETE statements. Like serializable transactions, serializable sessions do not experience non-repeatable reads or phantom reads.

\*\*A session is defined as one or more than one continuous transactions. A user can explicitly start and close a session like a transaction.

## **Multiple Connections and Multiple Threading**

Daffodil DB allows multiple simultaneous connections to a database, even in embedded mode. Daffodil DB Server is fully multi-threaded, and so you can have multiple threads performing concurrent operations through the same connection.

## **Backing up and Restoring Daffodil DB Database\***

### **Backup a Database**

Daffodil DB does have a very strong mechanism for backup. It supports online backup (while the database is running) as well as offline backup.

#### **Online Backup**

To perform online backup of a database, right click on the corresponding database node, choose **BackUp**, and then select **Online Backup**. Fill up the required information in the dialog box that is displayed.

#### **Offline Backup**

To perform offline backup of a database, right click on the corresponding database node, choose **BackUp**, and then select **Offline Backup**. Fill up the required information in the dialog box that is displayed.

Offline back up can also be performed by simply using operating system commands to copy the database directory. In this case, the database must be shut down before performing offline backup.

### **Restoring from a Backup**

To restore a back-up, right click on the corresponding database node, choose **BackUp**, and then select **Restore**.

Restore can also be performed by simply copying the database back-up file(s) at the desired location and then booting Daffodil DB Embedded/Server normally.

For more information on back-up and restore, please refer to *Daffodil DB Tools Guide*

\* Features that are not supported in One\$DB.

## **Daffodil DB Security**

Daffodil DB provides a facility for User Authentication and User Authorization.

### **User Authentication**

Daffodil DB authenticates user name and password before allowing a user to access the system. The user requesting a connection must provide a valid name and password, which Daffodil DB server verifies against the repository of users defined in the Daffodil DB database, to which the user is trying to connect to.

User authentication is done at the time a connection is established with Daffodil DB server.

Users can be created and dropped through “Create User” and “Drop User” SQL commands respectively.

*Note: - For details on creating and dropping users and SQL commands syntax, refer to Daffodil DB SQL Reference Guide.*

### **User Authorization**

User authorization is a means for checking users’ rights to access different parts of Daffodil DB database. Daffodil DB user authorization mechanism is based on the roles and privileges. Every Daffodil DB user has a name and password, and also owns tables, views and other resources that he or she had created. By default, Daffodil DB user has access only to his or her respective schema.

With Daffodil DB, you have the capability to create or drop roles, and to grant or revoke privileges. A role can contain multiple privileges, which you can apply to multiple users, without the need for applying each privilege to one user at a time. Any user can grant roles to other users or to other roles, if they have the authority to do the same. Any user with the authority may grant additional privileges to roles or to other users.

When a new database is created, then a user with the corresponding username and password is created in the new database, without any schema. This user can create his/her schema, and can access all public objects.

The following SQL commands are useful for controlling user authorization:

- Create Role – to create a new role
- Drop Role – to drop a role
- Grant – to grant a privilege to a user or a role.
- Revoke – to revoke a privilege from a user or a role.
- Set Role – to assume a role.

Privileges can be granted on the following database objects:

- Tables

- Columns
- Triggers
- Stored Procedures
- Domains

The following kinds of privileges are available to Daffodil DB users:

Privilege Type	Database Objects on which privilege can be applied
Select	Table Column
Insert	Table Column
Update	Table Column
Delete	Table
References	Table Column
Trigger	Trigger
Execute	Stored Procedure
Usage	Domain

*Note: - For details on schema, roles, privileges and SQL commands syntax, refer to Daffodil DB SQL Reference Guide.*

### **Automatic Crash Recovery**

If the server shuts down unexpectedly due to common failures like abnormal application termination, operating system crash, power failure, etc., all uncommitted transactions gets automatically rolled back. The next time database gets started it reverts back to the state of its last successfully committed transaction. Daffodil DB has a very powerful and robust crash recovery mechanism, which prevents database corruption against common failures and provides a fast recovery mechanism in case of failure (if occurs).

## **Daffodil DB Cryptography\***

Cryptography is the mathematical science of encrypting the data (translating into an unreadable format) and then decrypting (translating back into a readable format by someone with a secret key) using an algorithm.

### **Database Encryption**

Database environment would not be completely secured without consideration of encryption technology. The term database encryption refers to the practice of obscuring the meaning of a piece of data by means of encoding before storing it in tables. The database encryption is implemented in such a way that it can only be decoded, read and understood by people for whom the data is intended. It is the process of encoding data to prevent unauthorized parties from viewing or modifying it.

Daffodil DB supports Database encryption in both Embedded and Network editions. In this type of cryptography, the database pages themselves are encrypted, if you activate the encryption option while creating a database. Encrypting your database ensures that the data is accessed by other applications only through Daffodil DB.

Process of encrypting data in Daffodil DB is efficient and simple. While creating a new database users can specify an encryption algorithm and the value for encryption key, which will be used for encrypting the objects and data contained in tables.

\* Features that are not supported in One\$DB.

## **Encryption algorithms supported by Daffodil DB\***

Encryption algorithms vary in their strength (or their perceived ability to protect data) and performance, but have many aspects in common. A good algorithm has properties that make plain-text difficult to decipher.

To ensure a tight-leashed security, Daffodil DB currently supports seven encryption algorithms namely Blowfish, AES, TEA, DES, Twofish, DES3 and IDEA. The users can encrypt objects and data before storing them in a table with the help of these proven encryption algorithms.

Algorithm	Description	Key Length
Blowfish	An unpatented symmetric key cryptography algorithm developed by Bruce Schneier, with a public domain implementation. Variable nature of its key length, make it ideal for both domestic and exportable use. It is very secure and relatively fast, but its key setup is designed to be relatively slow.	Variable from 32 to 448 bits
AES (Advanced Encryption Standard)	A symmetric key encryption technique which is fast replacing the commonly used DES standard. AES uses a symmetric block cipher that encrypts and decrypts 128-bit blocks of data. AES supports key lengths of 128, and 256-bits.	128 bits/256 bits
TEA (Tiny Encryption Algorithm)	This is one of the fastest and most efficient cryptographic algorithms in existence. It encrypts 64 data bits at a time using a 128-bit key. It seems highly resistant to differential cryptanalysis, and achieves complete diffusion (where a one-bit difference in the plaintext will cause approximately 32-bit differences in the cipher text) after only six iterations.	128 bits
DES (Data Encryption Standard)	An encryption block cipher developed in 1977 by IBM. It applies a 56-bit key to each 64-bit block of data. It provides strong encryption based on symmetric cryptography. It works best on hardware, but is extremely slow on software.	56 bits

Twofish	Twofish is a 128-bit symmetric block encryption algorithm that accepts a variable-length key up to 256 bits. Twofish is an alternative to DES and provides more efficient and quicker encryption of data than DES. Its conservative design allows the ability to trade off key setup time for encryption speed, as well as sacrificing smaller memory requirements to obtain greater encryption speed.	Variable – upto 256 bits
DES3 or 3DES or Triple-DES	A variant of DES, Triple-DES is based on using DES three times. This means that the input data is encrypted three times. The Triple-DES is considered much stronger than DES; however, it is rather slow compared to some new block ciphers. It is designed to minimize memory footprint while maximizing speed.	56 bit key used 3 times (= 168 bits)
IDEA (International Data Encryption Algorithm)	Developed in Europe as an alternative to exportable American ciphers such as DES which were too weak for serious use. IDEA is a block cipher that uses a 128-bit length key to encrypt successive 64-bit blocks of plaintext. IDEA is patented and, with strictly limited exceptions for personal use. Using it requires a license from Ascom.	128 bits

For more information on the various cryptographic algorithms, go to <http://www.kremlinencrypt.com/crypto/algorithms.html>

\* Features that are not supported in One\$DB.



## Setting Database Encryption in Daffodil DB\*

Users can create an encrypted database with Daffodil DB in two ways

### 1. Through get connection method of JDBC 3.0.

If you are creating a database through JDBC Interface, you should use the following syntax to set up database encryption support.

**Important:** Users need to set ENCRYPTIONSUPPORT property as true at time of creation of a Daffodil DB database.

#### Syntax

```
String url = "jdbc:daffodilDB_embedded:<Database name>;create=true";
String driver = "in.co.daffodil.db.jdbc.DaffodilDBDriver";
Properties prop = new Properties();
prop.setProperty("user",<user name>);
prop.setProperty("password",<password>);
prop.setProperty("create","true");
prop.setProperty("ENCRYPTIONSUPPORT","true");
prop.setProperty("ENCIPHERALGO",<name of the encryption algorithm>);
prop.setProperty("ENCRYPTIONKEY",<specify the encryption key>);
Class.forName(driver);
java.sql.Connection con = DriverManager.getConnection(url,prop);
```

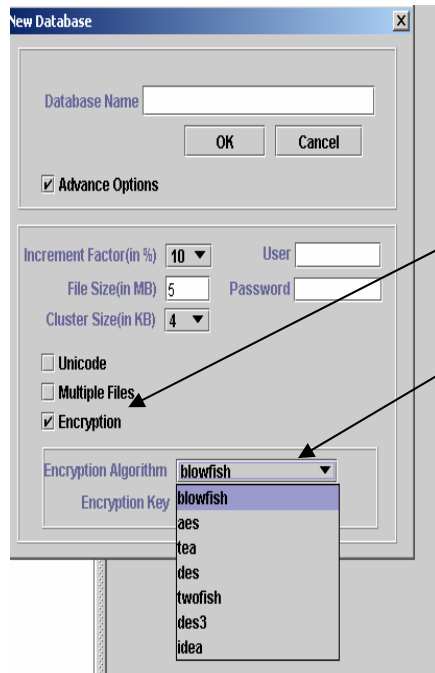
Example:

```
String url = "jdbc:daffodilDB_embedded:STUDENTDB;create=true";
String driver = "in.co.daffodil.db.jdbc.DaffodilDBDriver";
Properties prop = new Properties();
prop.setProperty("user","daisy");
prop.setProperty("password","daisy");
prop.setProperty("create","true");
prop.setProperty("ENCRYPTIONSUPPORT","true");
prop.setProperty("ENCIPHERALGO","tea");
prop.setProperty("ENCRYPTIONKEY","daisy");
Class.forName(driver);
java.sql.Connection con = DriverManager.getConnection(url,prop);
```

In the above stated example, a database named STUDENTDB is created which will be stored in encrypted form using TEA encryption algorithm with encryption key daisy.

\* Features that are not supported in One\$DB.

## 2. Through Daffodil DB Browser



- Open Daffodil DB browser
- Choose create Database
- Check the Encryption check box
- Choose the encryption algorithm
- Provide the encryption key

For more information on Daffodil DB database encryption, please read our article at:  
<http://www.daffodildb.com/daffodildb-encryption.html>

\* Features that are not supported in One\$DB.

## **Daffodil DB Distributed Transaction Support\***

Daffodil DB is a J2EE-compliant database. As such, Daffodil DB can act as a part of a larger system that includes, among other things, JNDI server, connection pool module, transaction manager, resource manager, and user applications. Within this system, Daffodil DB can serve as the resource manager.

Daffodil DB distributed transaction support allows more than one database or connection to participate in the same transaction. Distributed transactions use *XADataSource* entries rather than *DataSource* or *ConnectionPoolDataSource* entries. Such entries can participate in distributed transactions using the methods provided in the *XADataSource* implementation

In order to qualify as a resource manager in a J2EE system, J2EE requires the following basic areas of support:

- JNDI support
- Connection Pooling
- XA Support

You can use Daffodil DB in a Distributed Transaction Processing (DTP) environment to write enterprise JavaBeans that are transactional across multiple Daffodil DB Servers. Workgroup environments, such as J2EE and J2SE, where the data extends across multiple databases can benefit using Daffodil DB, because Daffodil DB JDBC driver supports the 2-phase commit protocol, which is used by Java Transactions API (JTA).

*Note: - For details on how to use Daffodil DB as a Resource Manager in a distributed transaction processing (DTP) environment, refer to Daffodil DB JDBC Reference Guide.*

\* Features that are not supported in One\$DB.

## Daffodil DB Unique Features

### ResultSet Listener Support

Unique concept of ResultSet listener support allows automatic updation of ResultSet or RecordSets (Note: - RecordSet is considered to be a synonym of ResultSet and we have used them interchangeably in this document.) that are updated according to the isolation level, if underlying database tables are changed, without any need for the Listener to execute the query again. Daffodil DB takes care of such changes, by firing an event on RecordSet, whenever there is any change in parent data objects that may affect ResultSet or RecordSet data in any way. For example if

```
myResultSet = statement.executeQuery ("Select * from City_Temperature");
```

And the ResultSet contains the records:

#### City\_Temperature

City	Temperature
New York	45
London	48
Delhi	91

Say, if the data in City\_Temperature table is changed, such that the temperature of Delhi is set to 100, then *myResultSet* will be updated automatically to reflect the latest information in the database, without explicitly pooling for the latest information from the database by *myResultSet*, and the *ResultSet* will automatically update to:

City	Temperature
New York	45
London	48
Delhi	100

The information that is updated in the ResultSet (and the users view on screen) depends upon the isolation levels of the transaction.

For e.g. if the transaction isolation level is set to "Read Committed" and data in the underlying table is changed by a different user and committed, only then the ResultSet will get updated.

However, if the transaction isolation level has been set to "Serializable" and data in the underlying table is changed by a different user and committed, then the ResultSet will not get updated.

The remarkable concept of ResultSet Listener support, gives the developers power of "call-backs" as in RMI clients. Using this concept, application designers can let Daffodil DB update the ResultSet, without firing query to the database engine, as ResultSets or RecordSets are automatically updated. Care should be taken that the ResultSet should be kept open for the duration, when utility is required. Highly volatile data that gets

updated frequently can use this utility to have access to fresh data without firing new queries periodically.

## **Session Serializable**

Apart from supporting isolation levels described in SQL-99 standards, Daffodil DB also supports another isolation level, which is unique to Daffodil DB and is defined as “Session Serializable”

The concept of Serializable Session applies serialization to a user session (a session is defined as one or more than one continuous transactions. A user can explicitly start and close a session like a transaction) that comprises of one or more transactions, in a way just like transaction isolation level of “Serializable” applies serializability to one particular transaction.

Serializable transactions (with transaction isolation level of “Serializable”) see only those changes that were committed at the time the transaction began, plus those changes made by the transaction through INSERT, UPDATE, and DELETE statements and do not experience non-repeatable reads or phantom reads. But Session Serializable transactions (with transaction isolation level of “Session Serializable”) see only those changes that were committed at the time the session began, plus those changes made by the transactions in the session itself through INSERT, UPDATE, and DELETE statements and do not experience non-repeatable reads or phantom reads during the entire session, which can comprise of multiple transactions.

In cases where data is volatile, this isolation level excels in providing user a secure version of data during transactions occurring in a session. Other users, from the original user's perspective, do not affect this user's copy of data. User can always have the liberty of jumping from one isolation level to another, and therefore perfectly balance their isolation levels.

All queries and transactions in a Daffodil DB Serializable session see the entire database as a single point, so this isolation level is suitable where multiple consistent (without non-repeatable reads and phantom reads) queries must be issued in a session, where multiple transactions are involved. A report-writing application that generates summary data might use Serializable session isolation level because it provides a consistent view of data at a single point of time, but also allows INSERT, UPDATE, and DELETE commands and allows the user to perform multiple transactions (commit to database) without losing the original consistent view of data.

## **Session Sharing**

Another unique feature of Daffodil DB, “**Session Sharing**” allows multiple users to work under the same session and thus, allowing them to view and edit each other’s *ResultSet* (). This unique feature has been made possible with the help of *ResultSet Listener* Support provided by Daffodil DB.

## **End Note**

Although this manual reflects the most current information possible, you should read the ***Daffodil DB Release Notes*** from time to time for latest information and updates on Daffodil DB.

Release Notes are available at: <http://www.daffodildb.com/daffodil-release-notes.html>

## **Sign Up for Support**

If you have started working with Daffodil DB, please remember to sign up for the benefits you are entitled to as a Daffodil DB customer.

For free support, be a part of our online developer community at [Daffodil Developer Forum](#)

For buying support packages, please visit: <http://www.daffodildb.com/support-overview.html>

For more information regarding support, write to us at: [support@daffodildb.com](mailto:support@daffodildb.com)



## **We Need Feedback!**

If you spot a typographical error in the *Daffodil DB System Guide*, or if you have thought of a way to make this manual better, we would love to hear from you!

Please submit a report in Bugzilla <http://www.daffodildb.com/bugzilla/index.cgi> OR write to us at: [feedback@daffodildb.com](mailto:feedback@daffodildb.com)

## **License Notice**

© 2004, Daffodil Software Limited  
All Rights Reserved.

This manual, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. The content of this manual is for informational purpose only, and is liable to change without prior notice. Daffodil Software Limited assumes no responsibility or liability whatsoever for any errors or inaccuracies that may appear in this documentation. No part of this product or any other product of Daffodil Software Limited or related documentation may be stored, transmitted, reproduced or used in any other manner in any form by any means without prior written authorization from Daffodil Software Limited.

## **Appendix A: System Tables**

This section describes Daffodil DB system tables. The Daffodil DB RDBMS stores information on data structures in the system tables. You cannot modify these system tables, because they are automatically updated as a result of SQL operations performed by the database system. The various system tables are:

**CHECK COLUMN USAGE**

Contains one row for each column identified by a <column reference> contained in the <search condition> of a check constraint, domain constraint, or assertion.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	Catalog name of the constraint being described.
CONSTRAINT_SCHEMA	varchar(128)	Unqualified schema name of the constraint being described.
CONSTRAINT_NAME	varchar(128)	Qualified identifier of the constraint being described.
TABLE_CATALOG	varchar(128)	Catalog name of the column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.
TABLE_SCHEMA	varchar(128)	Unqualified schema name of the column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.
TABLE_NAME	varchar(128)	Qualified identifier of the column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.
COLUMN_NAME	varchar(128)	Column name of the column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.

**CHECK TABLE USAGE**

Contains one row for each table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of a check constraint, domain, constraint, or assertion.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	Catalog name of the constraint being described.
CONSTRAINT_SCHEMA	varchar(128)	Unqualified schema name of the constraint being described.
CONSTRAINT_NAME	varchar(128)	Qualified identifier of the constraint being described.
TABLE_CATALOG	varchar(128)	Catalog name of the table identified by a <table name> contained in a <table reference> contained in the <search condition> of the constraint being described.
TABLE_SCHEMA	varchar(128)	Unqualified schema name of the table identified by a <table name> contained in a <table reference> contained in the <search condition> of the constraint being described...
TABLE_NAME	varchar(128)	Qualified identifier of the table identified by a <table name> contained in a <table reference> contained in the <search condition> of the constraint being described.

**CHECK CONSTRAINTS**

Each CHECK constraint occupies exactly one row in the current database.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	Catalog name of the constraint being described.
CONSTRAINT_SCHEMA	varchar(128)	Unqualified schema name of the constraint being described.
CONSTRAINT_NAME	varchar(128)	Qualified identifier of the constraint being described.
CHECK_CLAUSE	varchar(128)	CHECK_CLAUSE is the character representation.

**COLUMN PRIVILEGES**

The COLUMN\_PRIVILEGES table has one row for each column privilege descriptor. It effectively contains a representation of the column privilege descriptors.

Column name	Data type	Description
GRANTOR	varchar(128)	The value of GRANTOR is the <authorization identifier> of the user or role who granted column privileges.
GRANTEE	varchar(128)	The value of GRANTEE is the <authorization identifier> of some user or role.
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of the column on which the privilege being described was granted.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the column on which the privilege being described was granted.
TABLE_NAME	varchar(128)	The value of TABLE_NAME is the qualified identifier of the column on which the privilege being described was granted.
COLUMN_NAME	varchar(128)	The value of COLUMN_NAME is the column on which the privilege being described was granted.
PRIVILEGE_TYPE	varchar(128)	Type of privilege.
IS_GRANTABLE	varchar(128)	Specifies whether the grantee has the ability to grant permissions to others.

**COLUMNS**

The COLUMNS table has one row for each column descriptor. It effectively contains a representation of the column descriptors.

Column name	Data type	Description
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of the table containing the column being described.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the table containing the column being described.
TABLE_NAME	varchar(128)	The value of TABLE_NAME is the qualified identifier of the table containing the column being described.
COLUMN_NAME	varchar(128)	Column name.
ORDINAL_POSITION	Smallint	Column identification number.
DTD_IDENTIFIER	varchar(128)	It describes the data type of the column.
DOMAIN_CATALOG	varchar(128)	The value of DOMAIN_CATALOG is the unqualified schema name of the domain used by the column being described.
DOMAIN_SCHEMA	varchar(128)	The value of DOMAIN_SCHEMA is the unqualified schema name of the domain used by the column being described.
DOMAIN_NAME	varchar(128)	The value of DOMAIN_NAME is the unqualified schema name of the domain used by the column being described.
COLUMN_DEFAULT	varchar(1024)	Default value of the column.
IS_NULLABLE	varchar(1024)	Nullability of the column. If this column allows NULL, this column returns YES. Otherwise, NO is returned.
IS_SELF_REFERENCING	varchar(1024)	Self Referencibility of the column. If this column allows Self Referencibility, this column returns YES. Otherwise, NO is returned.



**DATA TYPE DESCRIPTOR**

Contains one row for each domain, one row for each column(in each table) and for each attribute (in each structured type) that is defined as having a datatype rather than a domain, one row for each distinct type, one row for the result type of each SQL parameter of each SQL-invoked routine, one row for the result type of each method specification, one row for each parameter of each method specification, and one row for each structured type whose associated reference type has a user-defined representation. It effectively contains a representation of the data type descriptors.

Column name	Data type	Description
OBJECT_CATALOG	varchar(128)	Catalog name of the schema that contains the object whose descriptor includes the data type descriptor.
OBJECT_SCHEMA	varchar(128)	Unqualified schema name of the schema that contains the object whose descriptor includes the data type descriptor.
OBJECT_NAME	varchar(128)	Qualified identifier of the schema that contains the object whose descriptor includes the data type descriptor.
OBJECT_TYPE	varchar(1024)	The Value may be 'TABLE','DOMAIN','ROUTINE' as the case may be.
DTD-IDENTIFIER	varchar(128)	The value of DTD_IDENTIFIER is the implementation-dependent value that uniquely identifies the data type descriptor among all data type descriptors of the schema object identified by OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and OBJECT_TYPE.
DATA_TYPE	varchar(1024)	Data type being described.
CHARACTER_MAXIMUM_LENGTH	int	The maximum length in characters or bits if it is a character or bit type respectively.
CHARACTER_OCTET_LENGTH	int	Maximum length in octets if it is a character type.
COLLATION_CATALOG	varchar(128)	The qualified catalog name of the applicable collation if it is a

		character type.
COLLATION_SCHEMA	varchar(128)	Unqualified schema name of the applicable collation if it is character type.
COLLATION_NAME	varchar(128)	Qualified identifier of the applicable collation if it is a character type.
NUMERIC_PRECISION	int	The precision if data type is a numeric.
NUMERIC_PRECISION_RADIX	int	Radix of the precision if data type is a numeric.
NUMERIC_SCALE	int	The scale if data type is numeric.
DATETIME_PRECISION	int	The fractional second's precision if data type is a timestamp
INTERVAL_TYPE	varchar(1024)	If DATA_TYPE is 'INTERVAL', then its values are the value for <interval qualifier> for the data type being described; otherwise, it has the null value.
INTERVAL_PRECISION	int	If DATA_TYPE is 'INTERVAL', then the values are the interval leading field precision of the data type being described; otherwise, it has the null value.
USER_DEFINED_TYPE_CATALOG	varchar(128)	Its value is null if the data type being described is not a user-defined type otherwise, it is the qualified catalog name.
USER_DEFINED_TYPE_SCHEMA	varchar(128)	Its value is null if the data type being described is not a user-defined type otherwise, it is the unqualified schema name.
USER_DEFINED_TYPE_NAME	varchar(128)	Its value is null if the data type being described is not a user-defined type otherwise, it is the identifier.
SCOPE_CATALOG	varchar(128)	If DATA_TYPE is 'REF', then it is the qualified catalog name of the reference able table, otherwise it is null.
SCOPE_SCHEMA	varchar(128)	If DATA_TYPE is 'REF', then it has the unqualified catalog name of the reference able table, otherwise it is null.
SCOPE_NAME	varchar(128)	If DATA_TYPE is 'REF', then it is the qualified identifier of the

		reference able table, otherwise it is null.
MAXIMUM_CARDINALITY	int	The value of MAXIMUM_CARDINALITY is the maximum cardinality of the array type being described if DATA_TYPE is 'ARRAY' Otherwise it is null.

**DOMAIN CONSTRAINTS**

Contains one row for each user-defined data type, accessible to the current user in the current database, with a rule bound to it.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	The value of CONSTRAINT_CATALOG is the catalog name of the schema in which the domain constraint is defined.
CONSTRAINT_SCHEMA	varchar(128)	The value of CONSTRAINT_SCHEMA is the unqualified schema name of the schema in which the domain constraint is defined.
CONSTRAINT_NAME	varchar(128)	The value of CONSTRAINT_NAME is the name of the domain constraint.
DOMAIN_CATALOG	varchar(128)	The value of DOMAIN_CATALOG is the catalog name of the domain in which the domain constraint is defined.
DOMAIN_SCHEMA	varchar(128)	The value of DOMAIN_SCHEMA is the unqualified schema name of the domain in which the domain constraint is defined.
DOMAIN_NAME	varchar(128)	The value of DOMAIN_NAME is the qualified identifier of the domain in which the domain constraint is defined.
IS_DEFERRABLE	varchar(1024)	Specifies whether constraint checking is deferrable. Always returns NO.
INITIALLY_DEFERRED	varchar(1024)	Specifies whether constraint checking is initially deferred. Always returns NO.

**DOMAINS**

Contains one row for each user-defined data type, which is accessible to current user in the current database.

Column name	Data type	Description
DOMAIN_CATALOG	varchar(128)	Database in which the user-defined data type exists.
DOMAIN_SCHEMA	varchar(128)	User that created the user-defined data type.
DOMAIN_NAME	varchar(128)	User-defined data type.
DTD_IDENTIFIER	varchar(128)	The value of DTD_IDENTIFIER is the value of DTD_IDENTIFIER of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the column
DOMAIN_DEFAULT	varchar(1024)	The value of DOMAIN_DEFAULT is null if the domain being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of DOMAIN_DEFAULT is "TRUNCATED". Otherwise, the value of DOMAIN_DEFAULT is a character representation of the default value for the domain that obeys the rules specified for <default option>.

**KEY COLUMN USAGE**

Contains one row for each column that is constrained as a key in the current database. This information schema view returns information on the objects to which the current user has permissions.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	The value of CONSTRAINT_CATALOG is the catalog name of the constraint being described.
CONSTRAINT_SCHEMA	varchar(128)	The value of CONSTRAINT_SCHEMA is the unqualified schema name of the constraint being described.
CONSTRAINT_NAME	varchar(128)	The value of CONSTRAINT_NAME is the qualified identifier of the constraint being described.
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of the column that participates in the unique, primary key, or foreign key constraint being described.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the table name, and the column name of the column that participates in the unique, primary key, or foreign key constraint being described.
TABLE_NAME	varchar(128)	The value of TABLE_NAME is the qualified identifier of the table name, and the column name of the column that participates in the unique, primary key, or foreign key constraint being described.
COLUMN_NAME	varchar(128)	Column name
ORDINAL_POSITION	Int	Column ordinal position

**METHOD SPECIFICATION PARAMETERS**

Contains one row for each SQL parameter of every single method specification that is described in the METHOD\_SPECIFICATIONS base table.

Column name	Data type	Description
SPECIFIC_CATALOG	varchar(128)	Catalog name of the specific Name of the SQL-invoked method whose parameters are being described.
SPECIFIC_SCHEMA	varchar(128)	Unqualified schema name of the specific name of the SQL-invoked method whose parameters are being described.
SPECIFIC_NAME	varchar(128)	Qualified identifier of the specific name of the SQL-invoked method whose parameters are being described.
METHOD_CATALOG	varchar(128)	Catalog name of the specific name of the method name of the SQL-invoked method being described.
METHOD_SCHEMA	varchar(128)	Unqualified schema name of the method name of the SQL-invoked method being described.
METHOD_NAME	varchar(128)	Its values are the identifier of the method name of the SQL-invoked method being described.
METHOD_SPECIFIC_IDENTIFIER	varchar(128)	It corresponds to the DTD_IDENTIFIER in DATA_TYPE_DESCRIPTOR table.
ORDINAL_POSITION	int	Its value is the ordinal position of the SQL parameter in the SQL invoked method.
DTD_IDENTIFIER	varchar(128)	Unqualified schema name of the <table name> of the trigger being described.
PARAMETER_MODE	varchar(1024)	Its value is IN if the SQL parameter being described is an input parameter.
IS_RESULT	varchar(128)	Its value is YES if the parameter is the RESULT parameter of a type-

		preserving function, otherwise NO.
AS_LOCATOR	varchar(128)	Its value is YES if the parameter is passed AS LOCATOR, otherwise NO.
PARAMETER_NAME	varchar(128)	If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of PARAMETER_NAME is that <SQL parameter name>. otherwise its value is NULL.
FROM_SQL_SPECIFIC_CATALOG	varchar(128)	Catalog name of the from-sql routine for the parameter being described.
FROM_SQL_SPECIFIC_SCHEMA	varchar(128)	Unqualified schema name of the from-sql routine for the parameter being described.
FROM_SQL_SPECIFIC_NAME	varchar(128)	Qualified identifier of the from-sql routine for the parameter being described...



**METHOD SPECIFICATIONS**

Contains one row for each method specification.

Column name	Data type	Description
SPECIFIC_CATALOG	varchar(128)	Catalog name of the SQL-invoked method, the parameters of which are being described.
SPECIFIC_SCHEMA	varchar(128)	Unqualified schema name of the SQL-invoked method, the parameters of which are being described...
SPECIFIC_NAME	varchar(128)	Qualified identifier of the SQL-invoked method, the parameters of which are being described.
USER_DEFINED_TYPE_CATALOG	varchar(128)	Catalog name of the user-defined type with which the SQL-invoked method is associated.
USER_DEFINED_TYPE_SCHEMA	varchar(128)	Unqualified schema name of the user-defined type with which the SQL-invoked method is associated.
USER_DEFINED_TYPE_NAME	varchar(128)	Qualified identifier of the user-defined type with which the SQL-invoked method is associated.
METHOD_NAME	varchar(128)	Its value is the identifier of the method name of the described SQL-invoked method.
IS_STATIC	varchar(1024)	Returns YES if SQL invoked routine is a static method, else returns NO.
IS_OVERRIDING	varchar(1024)	Returns YES if SQL invoked routine is an overriding method, else returns NO.
METHOD_LANGUAGE	varchar(1024)	Its value is the explicit or implicit <language name> contained in the described method specification.
PARAMETER_STYLE	varchar(1024)	Its value is null if the defined method specifies LANGUAGE SQL, otherwise "SQL if The method specification specified is PARAMETER STYLE SQL. or GENERAL if The method

		specification specified is PARAMETER STYLE GENERAL”.
DTD_IDENTIFIER	varchar(1024)	It is the value of DTD_IDENTIFIER of the row in DATA_TYPE_DESCRIPTOR that describes the result type of the method.
IS_DETERMINISTIC	varchar(1024)	Its value is null if the defined method specifies LANGUAGE SQL; YES if the method is deterministic otherwise NO.
SQL_DATA_ACCESS	varchar(1024)	It can have following values: NONE –if the SQL-invoked routine does not possibly contain SQL. CONTAINS –if the SQL- invoked routine possibly contains SQL. READS – if the SQL-invoked routine possibly reads SQL-data. MODIFIES – if the SQL-invoked routine possibly modifies SQL-data.
IS_NULL_CALL	varchar(1024)	Returns YES, if SQL-invoked routine is a null-call function, otherwise returns NO.
TO_SQL_SPECIFIC_CATALOG	varchar(128)	Its value is NULL if the defined method specifies LANGUAGE SQL; otherwise it is the qualified catalog name of TO- SQL routine for the result type of the described SQL-invoked method.
TO_SQL_SPECIFIC_SCHEMA	varchar(128)	Its value is NULL if the defined method specifies LANGUAGE SQL; otherwise it is the unqualified schema name of TO-SQL routine for the result type of the described SQL- invoked method.
TO_SQL_SPECIFIC_NAME	varchar(128)	Its value is NULL if the defined method specifies LANGUAGE SQL; otherwise it is the qualified identifier of TO-SQL routine for the result type of the described SQL-invoked method.
AS_LOCATOR	varchar(1024)	Returns YES, if the return value is passed as LOCATOR

		otherwise returns NO.
CREATED	Timestamp	The value of CREATED is the value of CURRENT_TIMESTAMP at the time, when described SQL invoked method specification was created.
LAST_ALTERED	Timestamp	The value of LAST_ALTERED is the value of CURRENT_TIMESTAMP at the time when described SQL-invoked method specification was last altered. In case unaltered then its value is same as CREATED.

**PARAMETERS**

The PARAMETERS table has one row for each SQL parameter of every single SQL-invoked routine described in the ROUTINES base table.

Column name	Data type	Description
SPECIFIC_CATALOG	varchar(128)	The value of SPECIFIC_CATALOG is the catalog name of the value of specific name of the SQL-invoked method with described parameters.
SPECIFIC_SCHEMA	varchar(128)	The value of SPECIFIC_SCHEMA is the unqualified schema name of the value of specific name of the SQL-invoked method with defined parameters.
SPECIFIC_NAME	varchar(128)	The value of SPECIFIC_NAME is the qualified identifier of the value of specific name of the SQL-invoked method with described parameters.
ORDINAL_POSITION	smallint	Ordinal position of the parameter starting at 1. For the return value of a function, this is at 0.
PARAMETER_MODE	varchar(1024)	Returns IN if an input parameter, OUT if an output parameter, and INOUT if an input/output parameter.
IS_RESULT	varchar(1024)	Returns YES if indicates result of a routine that is a function. Otherwise, returns NO.
AS_LOCATOR	varchar(1024)	Returns YES if declared as locator. Otherwise, returns NO.
PARAMETER_NAME	varchar(128)	Name of the parameter. NULL if this corresponds to the return value of a function.
DTD_IDENTIFIER	varchar(128)	It describes the data type of the column.
FROM_SQL_SPECIFIC_CATALOG	varchar(128)	FROM_SQL_SPECIFIC_CATALOG is the catalog name of the value of specific name of the FROM-SQL routine for the described parameter.
FROM_SQL_SPECIFIC_SCHEMA	varchar(128)	FROM_SQL_SPECIFIC_SCHEMA is the unqualified schema name of the value of specific name of the FROM-SQL routine for the described parameter.
FROM_SQL_SPECIFIC_NAME	varchar(128)	FROM_SQL_SPECIFIC_NAME is the qualified identifier of the value of specific name of the from-sql routine for the parameter being described.

TO_SQL_SPECIFIC_CATALOG	varchar(128)	TO_SQL_SPECIFIC_CATALOG is the catalog name of the value of specific name of the to-sql routine for the result type of the described SQL-invoked method.
TO_SQL_SPECIFIC_SCHEMA	varchar(128)	TO_SQL_SPECIFIC_SCHEMA is unqualified schema name of the specific name of the TO-SQL routine for the result type of the described SQL-invoked method.
TO_SQL_SPECIFIC_NAME	varchar(128)	TO_SQL_SPECIFIC_NAME is qualified identifier of the specific name of the to-sql routine for the result type of the SQL-invoked method being described.

**REFERENTIAL CONSTRAINTS**

Contains one row for each foreign constraint in the current database. This information schema view returns information on the objects to which the current user has permissions.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	Its value is the catalog name of the described constraint.
CONSTRAINT_SCHEMA	varchar(128)	Its value is the unqualified schema name of the described constraint.
CONSTRAINT_NAME	varchar(128)	The value of CONSTRAINT_NAME is the qualified identifier of the described constraint.
UNIQUE_CONSTRAINT_CATALOG	varchar(128)	The value of UNIQUE_CONSTRAINT_CATALOG is the catalog name of the unique or primary key constraint applied to the referenced column list.
UNIQUE_CONSTRAINT_SCHEMA	varchar(128)	The value of UNIQUE_CONSTRAINT_SCHEMA is the unqualified schema name of the unique or primary key constraint applied to the referenced column list.
UNIQUE_CONSTRAINT_NAME	varchar(128)	The value of UNIQUE_CONSTRAINT_NAME is the qualified identifier of the unique or primary key constraint applied to the referenced column list.
MATCH_OPTION	varchar(1024)	Referential constraint-matching conditions. Always returns NONE, which means that no match is defined. The condition is considered a match if atleast one value in the foreign key column is NULL; Or All values in the foreign key column are not NULL and there is a row in the primary key table with exactly same key.
UPDATE_RULE	varchar(1024)	The action that is taken if a SQL-99 statement violates referential integrity is defined by this constraint. Returns either NO ACTION or CASCADE. If NO ACTION is specified on ON UPDATE for this

		<p>constraint, then the update of the primary key referenced in the constraint will not be propagated to the foreign key. If such update of a primary key will cause a referential integrity violation because at least one foreign key contains the same value, Daffodil DB will not execute any change to the parent or the referring tables. Daffodil DB also will raise an error. If CASCADE is specified on ON UPDATE for this constraint, then any change to the primary key value is automatically propagated to the foreign key value.</p>
DELETE_RULE	varchar(1024)	<p>The action that is taken if a SQL-99 statement violates referential integrity defined by this constraint. Returns either NO ACTION or CASCADE. If NO ACTION is specified on ON DELETE for this constraint, then the delete on the primary key referenced in the constraint will not be propagated to the foreign key. If such delete of a primary key will cause a referential integrity violation because at least one foreign key contains the same value, Daffodil DB will not execute any change to the parent or referring tables. Daffodil DB also will raise an error. If CASCADE is specified on ON DELETE on this constraint, then any change to the primary key value is automatically propagated to the foreign key value.</p>

**ROLE AUTHORIZATION DESCRIPTORS**

Contains a representation of the role authorization descriptors. A row is inserted into this table whenever a <grant role statement> or <role definition> is executed unless the necessary row already exists, in which case the existing row may be modified to change the IS\_GRANTABLE column. A row is deleted from this table whenever a <revoke role statement> or <drop role> is executed.

Column name	Data type	Description
ROLE_NAME	varchar(128)	The value of ROLE_NAME is the <role name> of some <role granted> by the <grant role statement> or <role name> of the <role definition>.
GRANTEE	varchar(128)	The value of GRANTEE is an <authorization identifier>, possibly PUBLIC, or <role name> specified as a <grantee> contained in a <grant role statement>, or the <authorization identifier> of the current SQL-session when the <role definition> is executed.
GRANTOR	varchar(128)	The value of GRANTOR is the <authorization identifier> of the user or role who granted the role identified by ROLE_NAME to the user or role identified by the value of GRANTEE.
IS_GRANTABLE	varchar(1024)	Returns YES if the described role is grantable. A role is grantable if WITH ADMIN OPTION is specified in the <grant role statement> or a <role definition> is executed.



## **ROLES**

The ROLES table has one row for each <role name> for every single role known to the database management system. A row is inserted into this table each time a <role definition> is executed. A row is deleted from this table each time the <drop role statement> is executed.

Column name	Data type	Description
ROLE_NAME	varchar(128)	The value of ROLE_NAME is the <role name> defined by <role definition>.

**ROUTINE PRIVILEGES**

Contains one row for each 'execute privilege descriptor' for an SQL invoked routine. It effectively contains a representation of the execute privilege descriptors.

Column name	Data type	Description
GRANTOR	varchar(128)	The value of GRANTOR is the <authorization identifier> of the user or role who granted execute privileges, on the SQL-invoked routine identified by SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, to the user or role identified by the value of GRANTEE for the described privilege.
GRANTEE	varchar(128)	The value of GRANTEE is the <authorization identifier> of some user or role, or 'PUBLIC' to indicate all users, to whom the privilege being described is granted.
SPECIFIC_CATALOG	varchar(128)	Catalog name of the SQL-invoked routine on which the privilege being described has been granted.
SPECIFIC_SCHEMA	varchar(128)	Unqualified schema name of the SQL-invoked routine on which the privilege being described has been granted.
SPECIFIC_NAME	varchar(128)	Qualified identifier of the SQL-invoked routine on which the described privilege has been granted.
PRIVILEGE_TYPE	varchar(1024)	Its value is EXECUTE if the user has EXECUTE privilege on the SQL-invoked routine identified by SPECIFIC_CATALOG, SPECIFIC_SCHEMA, & SPECIFIC_NAME.
IS_GRANTABLE	varchar(1024)	Its value is YES if the described privilege was granted WITH GRANT OPTION, otherwise NO.

**ROUTINES**

Contains one row for each stored procedure and function accessible to the current user in the current database. The columns that describe the return value apply only to functions. For stored procedures, these columns will be NULL.

Column name	Data type	Description
SPECIFIC_CATALOG	varchar(128)	Specific name of the catalog. For Daffodil DB™, this name is the same as ROUTINE_CATALOG.
SPECIFIC_SCHEMA	varchar(128)	Specific name of the catalog. For Daffodil DB™, this name is the same as ROUTINE_SCHEMA.
SPECIFIC_NAME	varchar(128)	Specific name of the catalog. For Daffodil DB™, this name is the same as ROUTINE_NAME.
ROUTINE_CATALOG	varchar(128)	Catalog name of the function.
ROUTINE_SCHEMA	varchar(128)	Owner name of the function.
ROUTINE_NAME	varchar(128)	Name of the function.
ROUTINE_TYPE	varchar(1024)	Returns PROCEDURE for stored procedures, and FUNCTION for functions.
MODULE_CATALOG	varchar(128)	NULL.
MODULE_SCHEMA	varchar(128)	NULL.
MODULE_NAME	varchar(128)	NULL.
UDT_CATALOG	varchar(128)	NULL.
UDT_SCHEMA	varchar(128)	NULL.
UDT_NAME	varchar(128)	NULL.
SECURITY_TYPE	varchar(1024)	The values of SECURITY_TYPE have the following meanings: DEFINER -- The external routine has the security characteristic DEFINER. INVOKER -- The external routine has the security characteristic INVOKER.
TO_SQL_SPECIFIC_CATALOG	varchar(128)	TO_SQL_SPECIFIC_CATALOG is the catalog name of the value of specific name of the to-sql routine for the result type of the described SQL-invoked routine.
TO_SQL_SPECIFIC_SCHEMA	varchar(128)	TO_SQL_SPECIFIC_SCHEMA is the unqualified schema name of the value of specific name of the to-sql routine for the result type of the described SQL-invoked routine.
TO_SQL_SPECIFIC_NAME	varchar(128)	TO_SQL_SPECIFIC_NAME is the qualified identifier of the value of

		specific name of the to-sql routine for the result type of the described SQL-invoked routine.
AS_LOCATOR	varchar(1024)	The values of AS_LOCATOR have the following meanings: YES -- When the return value of the described SQL-invoked routine is passed AS LOCATOR. NO – When the return value of the described SQL-invoked routine is not passed AS LOCATOR.
DTD_IDENTIFIER	varchar(128)	NULL.
ROUTINE_BODY	varchar(1024)	Returns SQL for an SQL-99 function and EXTERNAL for an externally written function. In Daffodil DB, functions will always be SQL.
ROUTINE_DEFINITION	varchar(12000 )	Definition text of the function or stored procedure if the function or stored procedure is not encrypted. Otherwise, returns NULL.
EXTERNAL_NAME	varchar(128)	NULL.
EXTERNAL_LANGUAGE	varchar(1024)	NULL.
PARAMETER_STYLE	varchar(1024)	NULL.
IS_DETERMINISTIC	varchar(1024)	Returns YES if the routine is deterministic. Returns NO if the routine is nondeterministic. Always returns NO for stored procedures.
SQL_DATA_ACCESS	varchar(1024)	Returns one of the following four values: NONE –When the function does not contain SQL. CONTAINS – When the function possibly contains SQL. READS – When the function possibly reads SQL data. MODIFIES – When the function possibly modifies SQL data. In Daffodil DB, returns READS for all functions, and MODIFIES for all stored procedures.
IS_NULL_CALL	varchar(1024)	Indicates if the routine will be called or not in case any of its arguments are set as NULL. In Daffodil DB, always returns YES.
SQL_PATH	varchar(1024)	NULL.

SCHEMA_LEVEL_ROUTINE	varchar(1024)	Returns YES if the function is a schema-level function, or NO if it is not a schema-level function. In Daffodil DB, it always returns YES.
MAX_DYNAMIC_RESULT_SETS	smallint	Maximum number of dynamic result sets is returned by a routine. Returns 0 in case of functions, and TBD if stored procedures.
IS_USER_DEFINED_CAST	varchar(1024)	Returns YES if function is a user-defined cast function, and NO if it is not a user-defined cast function. In Daffodil DB, it always returns NO.
IS_IMPLICITLY_INVOCABLE	varchar(1024)	Returns YES if the routine is implicitly invocable, and NO if function is not implicitly invocable. In Daffodil DB, it always returns NO.
CREATED	Timestamp	Time, when the routine was created.
LAST_ALTERED	Timestamp	The last time when the function was modified.

**SCHEMATA**

Contains one row for each database that has permissions for the current user. The INFORMATION\_SCHEMA.SCHEMATA view is based on the sysdatabases, sysconfigures, and syscharsets system tables.

Column name	Data type	Description
CATALOG_NAME	varchar(128)	Name of the database, where the current user has permissions.
SCHEMA_NAME	varchar(128)	Returns the name of the schema owner of object.
SCHEMA_OWNER	varchar(128)	Schema owner name.
DEFAULT_CHARACTER_SET_CATALOG	varchar(6)	Returns master, indicating the database where the default character set is defined.
DEFAULT_CHARACTER_SET_SCHEMA	varchar(3)	Returns DBO, indicating the name of the default character set owner.
DEFAULT_CHARACTER_SET_NAME	varchar(128)	Returns the name of the default character set.

**TABLE CONSTRAINTS**

Contains one row for each table constraint in the current database. This information schema view returns information on the objects to which the current user holds permissions.

Column name	Data type	Description
CONSTRAINT_CATALOG	varchar(128)	The value of CONSTRAINT_CATALOG is the catalog name of the described constraint.
CONSTRAINT_SCHEMA	varchar(128)	The value of CONSTRAINT_SCHEMA is the unqualified schema name of the described constraint.
CONSTRAINT_NAME	varchar(128)	The value of CONSTRAINT_NAME is the qualified identifier of the described constraint.
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of the name of the table to which the described table constraint applies.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the table to which the described table constraint applies.
TABLE_NAME	varchar(128)	The value of TABLE_NAME is the qualified identifier of the name of the table to which the described table constraint applies.
CONSTRAINT_TYPE	varchar(1024)	The values of CONSTRAINT_TYPE have the following meanings: FOREIGN KEY – When the constraint being described is a foreign key constraint. UNIQUE – When the constraint being described is a unique constraint. PRIMARY KEY – When the constraint being described is a primary key constraint. CHECK – When the constraint being described is a check constraint.
IS_DEFERRABLE	varchar(1024)	Specifies whether constraint checking is deferrable. It Always returns NO.
INITIALLY_DEFERRED	varchar(1024)	Specifies whether constraint checking is initially deferred. It Always returns NO.

**TABLE PRIVILEGES**

Contains one row for each table privilege granted to or by the current user in the current database.

Column name	Data type	Description
GRANTOR	varchar(128)	The value of GRANTOR is the <authorization identifier> of the user or role who granted table privileges.
GRANTEE	varchar(128)	The value of GRANTEE is the <authorization identifier> of some user or role, or 'PUBLIC' to indicate all users.
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of the table on which the described privileges had been granted.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the table on which the described privileges had been granted.
TABLE_NAME	Varchar(128)	The value of TABLE_NAME is the qualified identifier of the table on which the described privileges had been granted.
PRIVILEGE_TYPE	varchar(1024)	Type of privilege.
IS_GRANTABLE	varchar(1024)	Specifies whether the grantee has the ability to grant permissions to others.
WITH_HIERARCHY	varchar(1024)	The values of WITH_HIERARCHY have the following meanings: YES – When the privilege being described was granted WITH HIERARCHY OPTION and is thus grantable. NO – When the privilege being described was not granted WITH HIERARCHY OPTION and is thus not grantable.



**TABLES**

Contains one row for each table in the current database for which the current user holds permissions.

Column name	Data type	Description
TABLE_CATALOG	Varchar(128)	The value of TABLE_CATALOG is the catalog name of the schema in which the table is defined.
TABLE_SCHEMA	Varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the schema in which the table is defined.
TABLE_NAME	Varchar(128)	The value of TABLE_NAME is the name of the table.
TABLE_TYPE	Varchar(1024)	Type of table. Can be VIEW or BASE TABLE.
SELF_REFERENCING_COLUMN_NAME	varchar(128)	The value of SELF_REFERENCING_COLUMN_NAME is the name of the self-referencing column of the table, if any.
REFERENCE_GENERATION	varchar(1024)	The values of REFERENCE_GENERATION have the following meanings: SYSTEM GENERATED – When the values of the self-referencing column of the table are generated. USER GENERATED – When the values of the self-referencing column of the table are generated by the user. DERIVED – When the values of the self-referencing column of the table are generated from columns of the table. NULL – When the described table does not have a self-referencing column.
UDT_CATALOG	varchar(128)	The value of UDT_CATALOG is the catalog name of TY.
UDT_SCHEMA	varchar(128)	The value of UDT_SCHEMA is the unqualified schema name of TY.
UDT_NAME	varchar(128)	The value of UDT_NAME is the type name of TY.

**TRANSFORMS**

Contains one row for each transform.

Column name	Data type	Description
USER_DEFINED_TYPE_CATALOG	varchar(128)	Catalog name of the user-defined type for which the described transform applies.
USER_DEFINED_TYPE_SCHEMA	varchar(128)	Unqualified schema name of the user-defined type for which the described transform applies.
USER_DEFINED_TYPE_NAME	varchar(128)	Qualified identifier of the user-defined type for which the described transform applies.
SPECIFIC_CATALOG	varchar(128)	Catalog name of the SQL-invoked routine that acts as the transform function for the described transform. The value of GROUP_NAME is the identifier that acts as the name of a transform group.
SPECIFIC_SCHEMA	varchar(128)	Schema name of the SQL-invoked routine that acts as the transform function for the described transform. The value of GROUP_NAME is the identifier that acts as the name of a transform group.
SPECIFIC_NAME	varchar(128)	Qualified identifier of the SQL-invoked routine that acts as the transform function for the described transform.
GROUP_NAME_IDENTIFIER	varchar(128)	The value of GROUP_NAME is the identifier that acts as the name of a transform group.
TRANSFORM_TYPE	varchar(1024)	Its value is 'TO SQL' if the transform being described identifies a to-sql function else its value is 'FROM SQL' if the described transform identifies a from-sql function.

**TRIGGERS**

Contains one row for each trigger. It effectively contains a representation of the trigger descriptors.

Column name	Data type	Description
TRIGGER_CATALOG	varchar(128)	Catalog name of the described trigger.
TRIGGER_SCHEMA	varchar(128)	Unqualified schema name of the described trigger.
TRIGGER_NAME	varchar(128)	Qualified identifier of the described trigger.
EVENT_MANIPULATION	varchar(1024)	Its values may be: INSERT if the <trigger event> is insert. DELETE if the <trigger event> is delete. UPDATE if the <trigger event> is update.
EVENT_OBJECT_CATALOG	varchar(128)	Catalog name of the <table name> of the described trigger.
EVENT_OBJECT_SCHEMA	varchar(128)	Unqualified schema name of the <table name> of the described trigger.
EVENT_OBJECT_TABLE	varchar(128)	Qualified identifier of the <table name> of the described trigger.
ACTION_ORDER	int	Its value is the ordinal position of the triggered action in the list of triggers.
ACTION_CONDITION	varchar(1024)	The value of ACTION_CONDITION is a character representation of the <search condition> in the <triggered action> of the described trigger.
ACTION_STATEMENT	varchar(1024)	ACTION_STATEMENT is a character representation of the <triggered SQL statement list> in the <triggered action> of the described trigger.

ACTION_ORIENTATION	varchar(1024)	Its values may be: ROW if the <trigger action> specifies 'FOR EACH ROW'. STATEMENT if the <trigger action> specified 'FOR EACH STATEMENT'.
CONDITION_TIMING	varchar(1024)	The values of CONDITION_TIMING have the following meaning: BEFORE The <trigger action time> is BEFORE. AFTER The <trigger action time> is AFTER.
CONDITION_REFERENCE_OLD_TABLE	varchar(128)	Its value is the <old value correlation name> of the described trigger.
CONDITION_REFERENCE_NEW_TABLE	varchar(128)	Its value is the <new value correlation name> of the described trigger.
CREATED	Timestamp	The value of CREATED is the value of CURRENT_TIMESTAMP at the time when the described trigger was created.

**TRIGGERED UPDATE COLUMNS**

Contains one row for each column identified by a <column name> in a <trigger column list> of a trigger definition.

Column name	Data type	Description
TRIGGER_CATALOG	varchar(128)	Catalog name of the described trigger.
TRIGGER_SCHEMA	varchar(128)	Unqualified schema name of the described trigger.
TRIGGER_NAME	varchar(128)	Trigger name of the described trigger.
EVENT_OBJECT_CATALOG	varchar(128)	Catalog name of the table containing the described column.
EVENT_OBJECT_SCHEMA	varchar(128)	Schema name of the table containing the described column.
EVENT_OBJECT_TABLE	varchar(128)	Table name of the table containing the described column.
EVENT_OBJECT_COLUMN	varchar(128)	Column name on the described trigger event.

**TRIGGER\_COLUMN\_USAGE**

The TRIGGER\_COLUMN\_USAGE base table has one row for each column of a table identified by a <table name>, which is contained in a <table reference> that in turn is contained in the <search condition> of a <triggered action>. It is explicitly / implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger where trigger event is not UPDATE.

Column name	Data type	Description
TRIGGER_CATALOG	varchar(128)	Catalog name of the described trigger.
TRIGGER_SCHEMA	varchar(128)	Unqualified schema name of the described trigger.
TRIGGER_NAME	varchar(128)	Qualified identifier of the described trigger.
TABLE_CATALOG	varchar(128)	Catalog name of a column of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or is explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger where trigger event is not UPDATE.
TABLE_SCHEMA	varchar(128)	Schema name of a column of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or is explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger where trigger event is not UPDATE.
TABLE_NAME	varchar(128)	Qualified identifier of a column of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or is explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger where trigger event is not UPDATE.
COLUMN_NAME	varchar(128)	Column name of a column of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or is explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger where event is not UPDATE.

**TRIGGER TABLE USAGE**

Contains one row for each table identified by a <table name> contained in the <search condition> of a <triggered action> or referenced in a <triggered SQL statement> of a <trigger definition>.

Column name	Data type	Description
TRIGGER_CATALOG	varchar(128)	Catalog name of the described trigger.
TRIGGER_SCHEMA	varchar(128)	Unqualified schema name of the described trigger.
TRIGGER_NAME	varchar(128)	Qualified identifier of the described trigger.
TABLE_CATALOG	varchar(128)	Catalog name of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger.
TABLE_SCHEMA	varchar(128)	Schema name of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger.
TABLE_NAME	varchar(128)	Qualified identifier of a table identified by a <table name>, which is contained in the <search condition> of a <triggered action>, or referenced in a <triggered SQL statement> of a <trigger definition> of the described trigger.

**USAGE PRIVILEGES**

Contains one row for each usage privilege descriptor. It effectively contains a representation of the usage privilege descriptors.

Column name	Data type	Description
GRANTOR	varchar(128)	The value of GRANTOR is the <authorization identifier> of the user or role who grant usage privileges, on the object of the type identified by OBJECT_TYPE that is identified by OBJECT_CATALOG, OBJECT_SCHEMA, and OBJECT_NAME, to the user or role identified by the value of GRANTEE for the described usage privilege.
GRANTEE	varchar(128)	The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the described privilege is granted.
OBJECT_CATALOG	varchar(128)	Catalog name of the object to which the privilege applies.
OBJECT_SCHEMA	varchar(128)	Unqualified schema name of the object to which the privilege applies.
SPECIFIC_NAME	varchar(128)	Qualified identifier of the object to which the privilege applies.
OBJECT_TYPE	varchar(1024)	Its values may be 'DOMAIN', 'CHARACTER SET', 'COLLATION' and 'TRANSLATION' depending on the object to which the privilege applies.
IS_GRANTABLE	varchar(1024)	Its value is YES if the described privilege was granted WITH GRANT OPTION and is thus grantable otherwise the value is NO.



**USER\_DEFINED\_TYPE\_PRIVILEGES**

Contains one row for each user-defined type privilege descriptor. It effectively contains a representation of the privilege descriptors. A row is inserted into this table when a <grant statement> is executed, unless necessary row already exists, in which case the existing row may be modified to change the IS\_GRANTABLE column. One or more rows are deleted from this table when a <revoke statement> is executed for the user-defined type name.

Column name	Data type	Description
GRANTOR	varchar(128)	The value of GRANTOR is the <authorization identifier> of the user or role who granted access privileges, on the described TYPE USAGE privilege to the user or role identified by the value of GRANTEE.
GRANTEE	varchar(128)	The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the described user defined type privilege is granted.
USER_DEFINED_TYPE_CATALOG	varchar(128)	Catalog name of the user defined type to which the privilege applies.
USER_DEFINED_TYPE_SCHEMA	varchar(128)	Unqualified schema name of the user defined type to which the privilege applies.
USER_DEFINED_TYPE_NAME	varchar(128)	Qualified identifier of the user defined type to which the privilege applies.
PRIVILEGE_TYPE	varchar(1024)	Its value is TYPE USAGE if the user has TYPE USAGE privilege on this user-defined type.
IS_GRANTABLE	varchar(1024)	Its value is YES if the described privilege was granted WITH GRANT OPTION and is thus grantable, otherwise its value is NO.

**USER\_DEFINED\_TYPES**

Contains one row for each user-defined type.

Column name	Data type	Description
USER_DEFINED_TYPE_CATALOG	varchar(128)	Catalog name of the user-defined type that is defined.
USER_DEFINED_TYPE_SCHEMA	varchar(128)	Unqualified schema name of the user-defined type that is defined.
USER_DEFINED_TYPE_NAME	varchar(128)	Qualified identifier of the user-defined type that is defined.
USER_DEFINED_TYPE_CATEGORY	varchar(1024)	Its value is STRUCTURED or DISTINCT accordingly if the user defined type is a structured or distinct type.
SOURCE_DTD_IDENTIFIER	varchar(128)	Its value is NULL if USER_DEFINED_TYPE_CATEGORY is STRUCTURED; otherwise its value is DTD_IDENTIFIER of the row in DATA_TYPE_DESCRIPTOR that describes the source type of distinct type.
IS_INSTANTIABLE	varchar(1024)	Its value is YES if USER_DEFINED_TYPE is instantiable, otherwise its value is NO.
IS_FINAL	varchar(1024)	Its value is YES if USER_DEFINED_TYPE cannot have subclass, otherwise its value is NO.
ORDERING_FORM	varchar(1024)	Its value may be: NONE – If two values of this type are not compared. FULL – If two values of this type are not compared for equality or relative order. EQUALS – If two values of this type are not compared for equality only.
ORDERING_CATEGORY	varchar(1024)	Its value may be: RELATIVE – If two values of this type can be compared with a relative routine. MAP – If two values of this type can be compared with a map routine. STATE – If two values of this type

		can be compared with a state routine.
ORDERING_ROUTINE_CATALOG	varchar(128)	Catalog name of the specific name of the SQL-invoked routine that is used for ordering the user-defined type.
ORDERING_ROUTINE_SCHEMA	varchar(128)	Unqualified schema name of the specific name of the SQL-invoked routine that is used for ordering the user-defined type.
ORDERING_ROUTINE_NAME	varchar(128)	Qualified identifier of the specific name of the SQL-invoked routine that is used for ordering the user-defined type.
REFERENCE_TYPE	varchar(1024)	Its value may be: SYSTEM GENERATED – If REF values for tables of this structured type are system generated. USER GENERATED – If REF values for tables of this structured type are user generated of the data type specified by USER GENERATED TYPE. DERIVED – If REF values for tables of this structured type are derived from the columns corresponding to specified attributes.
REF_DTD_IDENTIFIER	varchar(128)	Its value is: NULL – If the value of REFERENCE_TYPE is not 'USER GENERATED', otherwise its value is DTD_IDENTIFIER, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the user-generated REF values of the structured type.

**USERS**

Contains one row for each <authorization identifier> referenced in the Information Schema. These are <authorization identifier>s that may grant privileges as well as the ones that may create a schema, or currently own a schema created through <schema definition>.

Column name	Data type	Description
USER_NAME	varchar(128)	The values of USER_NAME are known <authorization identifier>s.

**VIEW COLUMN USAGE**

Contains one row for each column in the current database that is used in view definition. This information schema view returns information on the objects to which the current user has permissions.

Column name	Data type	Description
VIEW_CATALOG	varchar(128)	The value of VIEW_CATALOG is the catalog name of the described view.
VIEW_SCHEMA	varchar(128)	The value of VIEW_SCHEMA is the unqualified schema name of the described view.
VIEW_NAME	varchar(128)	The value of VIEW_NAME is the qualified identifier of the described view.
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of a column of a table that is explicitly or implicitly referenced in the <query expression> of the described view.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of a column of a table that is explicitly or implicitly referenced in the <query expression> of the described view.
TABLE_NAME	varchar(128)	The value of TABLE_NAME is the qualified identifier of a column of a table that is explicitly or implicitly referenced in the <query expression> of the described view.
COLUMN_NAME	varchar(128)	The value of COLUMN_NAME is the column name of a column of a table that is explicitly or implicitly referenced in the <query expression> of the described view.

**VIEW TABLE USAGE**

Contains one row for each table in the current database that is used in a view. This information schema view returns information on the objects to which the current user has permissions.

Column name	Data type	Description
VIEW_CATALOG	varchar(128)	The value of VIEW_CATALOG is the catalog name of the described view.
VIEW_SCHEMA	varchar(128)	The value of VIEW_SCHEMA is the unqualified schema name of the described view.
VIEW_NAME	varchar(128)	The value of VIEW_NAME is the qualified identifier, respectively, of the described view.
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of a table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the described view.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of a table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the described view.
TABLE_NAME	varchar(128)	The value TABLE_NAME is the qualified identifier of a table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the described view.

**VIEWS**

Contains one row for each row in the TABLES table with a TABLE\_TYPE of 'VIEW'. Each row describes the query expression that defines a view. The table effectively contains a representation of view descriptors.

Column name	Data type	Description
TABLE_CATALOG	varchar(128)	The value of TABLE_CATALOG is the catalog name of the schema in which the table is defined.
TABLE_SCHEMA	varchar(128)	The value of TABLE_SCHEMA is the unqualified schema name of the schema in which the table is defined.
TABLE_NAME	varchar(128)	The value of TABLE_NAME is the name of the viewed table.
VIEW_DEFINITION	varchar(1024)	If the character representation of the <query expression>, which is contained in the corresponding view descriptor can be represented without truncation, then the value of VIEW_DEFINITION is that character representation. Otherwise, the value of VIEW_DEFINITION is NULL.
CHECK_OPTION	varchar(1024)	The values of CHECK_OPTION have the following meanings: 'CASCADE' -- The corresponding view descriptor indicates that the view has CHECK OPTION that is to be applied as CASCADE. 'LOCAL' -- The corresponding view descriptor indicates that the view has CHECK OPTION that is to be applied as LOCAL. 'NONE' -- The corresponding view descriptor indicates that the view does not have CHECK OPTION.
IS_UPDATABLE	varchar(1024)	Returns YES if the view is updateable, otherwise it returns NO.
IS_INSERTABLE_INTO	varchar(1024)	Returns YES if the view is insertable-into, otherwise it returns NO.

**INDEXCOLUMNS**

Contains one row for each column corresponding to the indexes in the index table.

Column name	Data type	Description
TABLE_CATALOG	varchar(128)	Catalog name of the table to which the index belongs.
TABLE_SCHEMA	varchar(128)	Unqualified schema name of the table to which the index belongs.
TABLE_NAME	varchar(128)	Qualified identifier of the table to which the index belongs.
IndexName	varchar(128)	Name of the index.
Column_Name	varchar(128)	Name of the column on which index is associated.
Ordertype	Boolean	Its value is TRUE if order is ascending, otherwise FALSE.
Ordinal_Position	int	Column identification number



**INDEXINFO**

Contains one row corresponding to each index

Column name	Data type	Description
TABLE_CATALOG	varchar(128)	Catalog name of the table to which the index belongs.
TABLE_SCHEMA	varchar(128)	Unqualified schema name of the table to which the index belongs.
TABLE_NAME	varchar(128)	Qualified identifier of the table to which the index belongs.
IndexName	varchar(128)	Name of the index.
IndexTableName	varchar(256)	Name of the table where index data is maintained
NumberOfRecords	int	Number of records in the table
RootNodeAddress	long	Address of root node of btree.
RootClusterSize	int	Size of root cluster.
RootRecordNumber	int	Position of the root record
FixedVariable	boolean	Its value is TRUE if all the columns to which index belongs are fixed, otherwise FALSE.

**TRANSACTION**

Contains one row for each committed transaction. This table is used to store last transaction id, session id and the record commit time.

Column name	Data type	Description
USERNAME	varchar(128)	Name of the user.
COMMITTIME	long	Time at which record was committed
SAVETRANSACTIONID	long	Last transaction id in the database.
SAVESESSIONID	long	Last session id in the database.
TRANSACTIONCOMPLETE	boolean	Its value is TRUE if transaction is complete, otherwise FALSE.

## **Appendix B: Error Messages**

The following table describes Daffodil DB error messages. In the message column, the numbers enclosed by braces represent parameter strings based on context of the error occurred.

DSE0={0}

DSE12=Access denied. Do not have 'CREATE or DROP' permission.

DSE14=An aggregate may not appear in the where clause unless it is in a subquery contained in a having clause or a SELECT list, and the being aggregated is an OUTER reference.

DSE15=OLD and NEW alias name cannot be identical.

DSE16=Function {0} not supported.

DSE17=Ambiguous column name {0}.

DSE22=Feature {0} not supported.

DSE27=Insufficient privileges.

DSE28=In auto commit mode.

DSE35=Syntax error converting data type {0} to {1}.

DSE40=Cannot concatenate character data type value {0} to byte data type value {1}.

DSE85=Cannot call Statement.executeUpdate() with this query.

DSE86=Cannot call Statement.executeQuery() with this query.

DSE87=Syntax error converting data type {0} to {1}.

DSE103=Cannot concatenate date with date.

DSE104=View or function {0} is not updatable because it contains aggregates.

DSE105=Cannot execute query through execute query method.

DSE146=Cannot read from the input stream.

DSE191=Cannot be dropped as this procedure is being referred by some other procedure.

DSE198=Cannot define the SQL data access as no-SQL.

DSE200=A routine definition can have at most one <deterministic characteristic>.

DSE201=A routine definition can have at most one <SQL-data access indication>.

DSE202=A routine definition can have at most one <dynamic result sets characteristic>.

DSE203=A routine definition can have at most one <specific name>.

DSE205=A routine definition can have at most one <language clause>.

DSE206=A routine definition can have at most one <parameter style clause>.

DSE210=Cannot specify <parameter mode> in function definition.

DSE212=Cannot use name and index in same statement.

DSE213=Cardinality cannot be greater than one.

DSE214=Cardinality does not match.

DSE224=Catalog name in constraint definition does not match catalog of table definition.

DSE225=Catalog name can not be changed as table name or schema name does not exist.

DSE226=Catalog name in schema definition and table definition for table {0} does not match.

DSE227=Catalog name in the view definition is not matching with that of schema definition.

DSE228=Catalog name of the trigger definition is not matching with that of schema definition.

DSE231=Character string type can not be null.

DSE232=Characteristics cannot be null.

DSE235=Check constraint definition does not exist.

DSE236=Check option cannot be specified with recursive view type.

DSE237=Check the code.

DSE239=Checkdatatype called from booleanvalueexpressionandbooleanvalueexpression.

DSE242=Callable statements are not supported.

DSE243=Collate clause can not specified for non-character column type.

DSE250=Column names in each table must be unique. Column name {0} in table {1} is specified more than once.

DSE251=ALTER TABLE DROP COLUMN statement failed because column {0} does not exist in table {1}.

DSE252=ALTER TABLE DROP COLUMN statement failed because {0} is the only data column in table {1}. A table must have at least one data column.

DSE255=Column name {0} does not exist in target table {1}.

DSE256=Invalid column name(s) {0}.

DSE260=Invalid column name(s) {0} in Group by clause.

DSE262=Column type not set.

DSE264=Column descriptor does not exist. TABLE CATALOG {0} TABLE SCHEMA {1} TABLE NAME {2} COLUMN NAME {3} PRIVILEGE TYPE {4}.

DSE265=Table descriptor does not exist. TABLE CATALOG {0} TABLE SCHEMA {1} TABLE NAME {2} PRIVILEGE TYPE {3}.

DSE266=Column {0} does not exist in table {1}.

DSE267=Column name {0} passed with value {1}.

DSE269=Column name specified in columnlist is not present.

DSE270=Columns corresponding to the constraint not found {0}.

DSE272=Cannot use duplicate column names in index key list. Column name {0} listed more than once.

DSE273=Columns specified in trigger column list are not unique.

DSE274=Trigger {0} uses the invalid column/columns {1} in trigger column list.

DSE275=Commit action can not be specified with type {0}.

DSE276=Commit is not allowed with read uncommitted level.

DSE279=Connection already closed.

DSE286={0} is not a constraint.

DSE287=Non deferrable constraint.

DSE288=ALTER TABLE DROP CONSTRAINT statement failed because constraint {0} is being referenced foreign key constraint {1}.

DSE289=Unique constraint referred by the referencing column should not be deferrable.

DSE294=Current row is not valid.

DSE295=Data cannot be loaded.

DSE297=Invalid data.

DSE301=Data not loaded in the {0} descriptor.

DSE306=Data not valid.

DSE310=Data type descriptor does not have any rows.

DSE316=Database {0} does not exist.

DSE314=Database {0} already exists.

DSE319=Data type should be of character type in case collate clause is specified.

DSE323=Default value of the domain descriptor is not set.

DSE324=Definition of rule {0} is not found.

DSE325=Degree can not be greater than one.

DSE326=Degree is exceeding to one in between predicate.

DSE330=Deletion is not allowed with read uncommitted level.

DSE334=Divide by zero error encountered.

DSE336=Domain constraint descriptor not loaded.

DSE337=Domain definition does not exist.

DSE338=Domain descriptor not set.

DSE339=Domainconstraint does not exist. Catalog: {0} schema: {1} constraint: {2}.

DSE340=Do not have permission to {0} the table {1}.

DSE342=Do not have any permission to SELECT a row.

DSE355=Error while getting the connection.

DSE356=Error in reading the stream.

DSE358=Exception.

DSE382=INSERT statement conflicted with FOREIGN KEY constraint {0}. The conflict occurred in table {1}.

DSE390=Global session not set.

DSE393=exception not handled correctly.

out-of-range datetime value.

DSE396={0} specified should be between {1} and {2}.

DSE401=Illegal call.

DSE402=Illegal call for condition: {0}.

DSE408=Illegal condition for join retriever optimal {0}.

DSE410=Database is incompatible, might be created in higher version.

DSE411=Illegal method call.

DSE412=Illegal multiplier passed {0}.

DSE413=Illegal timestamp denominator {0}.

DSE415=Invalid fetch direction. In case of type forward, only fetch forward is valid direction.

DSE416=Incompatible type {0}.

DSE419=Illegal/invalid argument passed in function {0}.

DSE478=Index cannot be maintained on column of type {0}.

DSE479=Index passed [{0}] must be >0 && <= {1}.

DSE482=Cannot find Index name {0}.

DSE483=Cannot defer a constraint that is not deferrable.

DSE484=Insert failed.

DSE487=Insertion is not allowed with read uncommitted level.

DSE489=Integrity constraint violation.

DSE492=Internal error data type descriptor is not set or is null.

DSE493=Internal error: columndescriptor not set.

DSE494=Internal exception: table descriptor not initialized.

DSE495=Internal exception: schema descriptor not set.

DSE496=Internal exception: column descriptor not set.

DSE504=Invalid column.

DSE505=Invalid column index: {0}.

DSE508=Invalid column name {0}.

DSE511=Invalid concurrency {0}.

DSE513=Invalid data passed.

DSE514=Invalid data type {0}.

DSE515=Cannot find data type/domain {0}.

DSE517=Invalid default option <{0}> {1}.

DSE518=Invalid fetch size.

DSE520=Invalid grantor.

DSE523=Invalid log file.

DSE524=Invalid logging level.



DSE525=Invalid method call.

DSE526=Invalid no of columns in function.

DSE527=Invalid number {0}.

DSE528=Invalid operator {0}.

DSE529=Invalid operator (-) for string manipulation.

DSE530=Invalid quantifier {0}.

DSE531=Invalid query.

DSE532=Invalid query to execute.

DSE533=Invalid query to execute update method.

DSE536=Invalid result from server.

DSE537=Syntax error converting datetime from character string.

DSE538=Invalid URL.

DSE540=This feature is not supported in ONE\$DB.

DSE541=Invalid value for fetch direction: {0}.

DSE542=Invalid value for holdability {0}.

DSE543=Invalid value for isolation level {0}.

DSE545=Invalid value passed for maxrows {0}.

DSE548=Iterator is not at valid row.

DSE550=Key can not be null.

DSE552=Key does not exist {0}.

DSE554=Key passed is null.

DSE555=Keys cannot be null key1 {0} key2 {1}.

DSE558=Length cannot be null.

DSE562=List is empty.

DSE563=Locator not allowed in procedures.

DSE565={0} not supported.

DSE567=Method can not be called with parameters.

DSE715=Neither schema name nor authorization identifier specified.

DSE716=NEW ROW alias cannot be specified with DELETE type of event.

DSE717=NEW ROW cannot be specified without FOR EACH ROW.

DSE718=NEW TABLE alias cannot be specified with DELETE type of event.

DSE720=Next of {0} is null.

DSE721=Column name not set

DSE723=No data exists corresponding to the data.

DSE725=No getter method called before this call.

DSE728=Cannot find the procedure {0} which has {1} arguments.

DSE729=Cannot drop the procedure {0}, because it does not exist.

DSE731=The variable name {0} has already been declared. Variable names must be unique within a stored procedure.

DSE732=No type available.

DSE736=Not a select query.

DSE737=Not a valid column.

DSE738=Not a valid direction {0}.

DSE740=Not a valid out parameter index {0} outparameters {1}.

DSE749=ColumnNames are null.

DSE750=Null not allowed in this column.

DSE752=Number of columns in query expression and view column list are not equal.

DSE753=Procedure parameters mismatch.

DSE754=Number of values are not equal to number of columns

DSE756=Object not convertible

DSE759=Object privilege should be of usage type in this case

DSE773=The value you entered is not consistent with the data type or length of the column.

DSE775=Old or old row, new or new row, old table, and new table shall be specified at most once each within the <old or new values alias list>.

DSE776=Old row alias can not be specified with insert type of event.

DSE777=Old row can not specified without for each row.

DSE778=Old table alias can not be specified with insert type of event.

DSE784=Parameter passed is {0}.

DSE785=Parameters and values count does not match.

DSE786=Parameters are null.

DSE792=Parameterstyleclause cannot be specified in procedure statement.

DSE798=Position must be greater then 0.

DSE799=Precision {0} can not be greater maximum value {1}.

DSE804=Problem while checking for index table.

DSE806=Problem while creating the database.

DSE808=Problem in getting retriever for table.

DSE810=Problem in getting table {0}.

DSE812=Problem in getting the constraints of the table.

DSE813=Problem in run method of default option.

DSE818=Put quotes around the values passed.

DSE819=Query not produced an update count.

DSE820=Query not produced ResultSet.

DSE822=Query expression not declared local temporary table.

DSE823=Query expression can not have target specifications.

DSE824=Query expression shall specify row type of element.

DSE825=Query failed.

DSE828=Query is either null or empty.

DSE832=Query time out must be greater than 0. value passed is {0}.

DSE836=Range is exceeding {0} start position {1} is greater than length of blob {2}.

DSE841=Record is not present.

DSE848=Record number is exceeding the total number of records {0}.

DSE849=Record number {0} passed is deleted.

DSE851=Recursive shall be specified.

DSE853=Referenceable view specification can not be specified with recursive.

DSE857=Referential constraint definition does not exist.

DSE861=Restrict called from {0}.

DSE869=Result can be specified at most one time.

DSE870=ResultSet not updateable.

DSE871=ResultSet type is forward only.

DSE874=Invalid role specification.

DSE875=Role authorization descriptor does not exist. Role name: {0} grantee: {1}.

DSE876=Role/user {0} already exists in database.

DSE877=Rollback is not possible, not a valid session.

DSE879=Row is locked by another user.

DSE883=Rowset is read-only.

DSE884=Rsb of rsbi {0}, record.

DSE885=Rule name passed is {0}.

DSE887=Runtime exception: table descriptor not set.

DSE889=Save point {0} already exists.

DSE890=Invalid save point.

DSE891=Scale {0} can not be greater maximum value {1}.

DSE892=Scale {0} can not be greater precision {1}.

DSE893={0} scale can not be negative.

DSE895=Schema and catalog specified does not exist.

DSE896=Schema {0} does not exist.

DSE897=Schema in the constraint definition does not match schema of table.

DSE902=Schema name does not match with that of table {0}.

DSE903=Schema name does not match with that of view defintion.

DSE904=Schema name not specified.

DSE905=Schema name does not match with that of trigger defintion.

DSE906=Schema name's domain definition does not match.

DSE907=Schema owned by {0} has not granted drop privilege to this user.

DSE915=Session does not exist.

DSE918={0} should contained either in aggregate function or group by columns.

DSE920=Should specify parameter name.

DSE925=Some problem.

DSE928=Specific name is already present in the schema.

DSE930=Specified table is not a base table {0}.

DSE931=Splitindexandnonindexconditions called from betweenpredicate.

DSE934=SQL invoked routines can only have execute privilege.

DSE937=Start position {0} is greater than the length of blobclob {1}.

DSE939=Start position {0} is greater than the length of blob {1}.

DSE940=Statement is closed.

DSE945=Sum or avg aggregate function can not accept character string as argument.

DSE946=Syntax error.

DSE953=Systemfield {0} does not exist.

DSE954=Ambiguous table name {0}.

DSE955=Table cannot be a local temporary one.

DSE956=Table definition does not contain column definitions.

DSE957=Table definition does not contain table elements.

DSE958=Table definition does not contain table content source.

DSE959=Invalid object name {0}.

DSE962=Table is not a base table. Alter table not allowed.

DSE963=Table is of reference able type.

DSE964=Table is referenced from the check constraints, can not be dropped.

DSE970=Table name does not exist

DSE972=Temporary table should have all privileges.

DSE973=The data type should be of Character String type but is of {0}.

DSE974=The data type does not require scale.

DSE975=Length {0} of column {1} can not be less than 128 for default clause

DSE976=Cannot find the index name {0} on table {1}.

DSE977=The is no corresponding index columns.

DSE978=The key is not present.

DSE982=The key passed is null or invalid.

DSE983=The length {0} can not be greater than valid length {1}.

DSE985=The length {0} is greater than valid length {1}.

DSE986=The list is empty.

DSE987=The literal length {0} is greater than {1}.

DSE989=The precision {0} is greater than valid precision {1}.

DSE991=The recordId {0} not an instance of query record.

DSE992=The record identity {0} not an instance of query record.

DSE994=The time precision {0} is greater than valid {1}.

DSE998=View {0} does not exist.

DSE999=There is no record corresponding to recordId {0}.

DSE1010=Transformgroupspecification cannot be specified in procedure statement.

DSE1012=You can specifiy trigger privilege only on base tables.

DSE1013=Trigger definition does not exist.

DSE1014=Trigger subject table is not a base table.

DSE1017=Truncated length {0} is greater than length of blobclob {1}.

DSE1018=Type {0}.

DSE1019=Inappropriate type {0}.

DSE1021=Type {0} not defined in database.

DSE1022=Type not registered.

DSE1024=mismatched type.

DSE1025=Types of table does not match.

DSE1026=Column {0} is not the same data type as referencing column {1}.

DSE1029=Unable to get the authorization identifier.

DSE1030=Unable to get the schema name.

DSE1032=Violation of UNIQUE KEY constraint {0}. Cannot insert duplicate key in object {1}.

DSE1036=Unsupported format.

DSE1037=Update cascade failed.

DSE1043=Update primary key returned by retriever cannot be null.

DSE1046=Usageprivilegedescriptor does not exist. Grantor: {0} grantee {1}  
OBJECT\_CATALOG: {2} OBJECT\_SCHEMA: {3} OBJECT\_NAME {4} OBJECT\_TYPE {5}.

DSE1047=User cannot be blank or \_SYSTEM.

DSE1051=Value cannot be null.

DSE1066=View column list must be specified with recursive view type.

DSE1067=View definition does not exist.

DSE1075=Year must be between 9999 and -4713.

DSE1077=You cannot repeat a routine characteristic.

DSE1081=Illegal argument exception.

DSE1082=The column specified in view query is not valid.

DSE1083=Column {0} not allowed as constraint column. Type is {1}.

DSE1084=Cannot use connect in routine definition.

DSE1086=There are no primary or candidate keys in the referenced table {0} that match the referencing column list in the foreign key {1}.

DSE1087=The length of check clause {0} is greater than the permitted length {1}.



DSE1088={0} is System Field; change the name of the column.

DSE1090=Create View failed because no column name was specified for functional column.

DSE1103=Length of escape character can not be greater than one.

DSE1104=Recordsetbuffer {0} record buffer {1}.

DSE1105={0} is not convertible in binary.

DSE1107=Problem in getting view characteristic.

DSE1108=Problem in getting column characteristidefc of table {0}.

DSE1109=Problem in getting default clause for table {0}.

DSE1110=Problem in getting unique constraint.

DSE1111=Problem in getting check constraints.

DSE1112=Problem in getting trigger characteristics.

DSE1122=Problem in checking for has deferred constraints.

DSE1124=Update failed due to dataexception.

DSE1126=Delete failed due to retrievalexception.

DSE1128=Commit failed due to data exception.

DSE1129=Perform failed due to data exception.

DSE1130=Column {0} not found in characteristics of table {1}.

DSE1131=Column index {0} not found in characteristics of table {1}.

DSE1133={0} data type not supported.

DSE1134=Dependent privilege descriptor still exist.

DSE1135=There is already an object named {0} in the database.

DSE1136=Column {0} already exists in the table {1}.

DSE1137=There is already an object named {0} in the database.

DSE1138={0} type table privilege already exists for grantor {1} grantee {2} table {3}.

DSE1139={0} type column privilege already exists for grantor {1} grantee {2} table {3} column name {4}.

DSE1141=Trigger with {0} name already exists.

DSE1142=There is already an index on table {1} named {0}.

DSE1143=Column {0} already exists for index {1} of table {2}.

DSE1144={0} domain already exists.

DSE1145=Domain constraint {0} already exists.

DSE1146=Schema {0} already exists.

DSE1147=Column {0} already exists for constraint {1}.

DSE1148={0} type usage privilege already exists for GRANTOR {1} GRANTEE {2} object {3}.

DSE1149={0} type routine privilege already exists for grantor {1} grantee {2} routine {3}.

DSE1151=Grantee {0} already exists for role {1}.

DSE1152=Method specification already exists for SPECIFIC CATALOG {0} SPECIFIC SCHEMA {1} SPECIFIC NAME {2}.

DSE1153=Number of values is not equal to number of parameters.

DSE1157=Parameters not set properly or not required.

DSE1160=Argument passing is not proper.

DSE1164=Updation is not allowed with read uncommitted level.

DSE1166=Passed object length {0} is more than {1}.

DSE1167=Required object of {0} passed {1}.

DSE1168=Property {0} of column index {1} contains invalid value {2}.

DSE1171=No such privilege descriptor found to delete.

DSE1172=Invalid privilege descriptor.

DSE1173=Some dependent entry exist.

DSE1174=Incorrect syntax at position {0} near {1}.

DSE1175=Problem in adding column {0}.

DSE1178=Time out {0}.

DSE1179=Catalog name is not same in object name {0} and schema descriptor {1}.

DSE1180=Schema and table descriptor both are set.

DSE1181=Granted or revoked privilege {0} is not compatible with object.

DSE1182=No action found in action list.

DSE1183=Table descriptor is null.

DSE1184=Access mode is read only.

DSE1199=Constraint cannot be created as data exists in table.

DSE1205=Cannot add multiple PRIMARY KEY constraints to table {0}.

DSE1206=File growth can be from 10 - 100 only is ?.

DSE1207=Insufficient privileges to drop the database.

DSE1208=Invalid username/password.

DSE1209=Insufficient privileges.

DSE1210=User {0} with password {1} does not exist.

DSE1211=Table is locked by another user.

DSE1214=Key is not valid.

DSE1216=Invalid status{0}.

DSE1219=Cannot insert in functional/aggregate columns.

DSE1249=Not a select query.

DSE1250=Cardinality for multiple rows is invalid.

DSE1251={0} statement conflicted with CHECK constraint {1} defined as {2}. the conflict occurred in table {3}.

DSE1252=Length of columns {0} not equal to length of REFERENCES {1}.

DSE1254=Problem in firing replace event.

DSE1255=Violation of PRIMARY KEY constraint {0}. Cannot insert duplicate key in object {1}.

DSE1256=Insert error: column name or number of supplied VALUES does not match table definition.

DSE1257=The name {0} is not permitted in this context. Only constants, expressions, or variables allowed here. Column names are not permitted.

DSE1258=Type of join applied in the query is not valid.

DSE1259=Some column has been updated with value not satisfying the query.

DSE1260=Cursor [ {0} ] already created in the current SQL SESSION.

DSE1261=Cursor {0} already opened

DSE1262=Column cannot be of reference type.

DSE1263=Cannot UPDATE the CURSOR, CURSOR is not updatable.

DSE1267=Cannot perform INSERT/UPDATE/DELETE directly in a VIEW.

DSE1269=Mismatch in length of columns and values.

DSE1270=The query is not valid.

DSE1271=The query does not support ORDER BY.

DSE1272=Non-updatable type of column in where clause,cannot INSERT/UPDATE in this query.

DSE1273=Value of some reference column not provided or invalid column in query.

DSE1274=Problem in Execution of Trigger For Table {0} While Performing Action [{1}] due to {2}.

DSE1275=Problem in Execution of Insert Statement for Table {0}.

DSE1276=Problem in Execution of Update Statement for Table {0}.

DSE1277=Problem in Execution of Delete Statement for Table {0}.

DSE1278=Referenced Constraint Violation [RESTRICT] for Table = {0} .

DSE1284=INSERT statement conflicted with FOREIGN KEY constraint {0} MATCH PARTIAL. Cannot update the NULL values in all referencing column(s).

DSE1285=INSERT statement conflicted with FOREIGN KEY constraint {0} MATCH PARTIAL. Cannot insert the NULL values in all referencing column(s).

DSE1286=DELETE statement conflicted with FOREIGN KEY constraint {0}. The conflict occurred in table {1}.

DSE1287=Referenced constraint violation.

DSE1288=UPDATE statement conflicted with UNIQUE KEY constraint {0}. The conflict occurred in table {1}.

DSE1289=UPDATE statement conflicted with PRIMARY KEY constraint {0}. The conflict occurred in table {1}.

DSE1290=UPDATE statement conflicted with FOREIGN KEY constraint {0}. The conflict occurred in table {1}.

DSE1291=Cannot insert the value NULL into column {0}, table {1}; column does not allow nulls. INSERT failed.

DSE1292=Cannot update the value NULL into column {0}, table {1}; column does not allow nulls. UPDATE failed.

DSE1293=Table name does not exist in mapping.

DSE1294=TableDetails for table name {0} passed in event is not found.

DSE1295=There are {0} columns in the INSERT statement than values specified in the values clause. the number of values in the VALUES clause must match the number of columns specified in the INSERT statement.

DSE1300=Aggregate function cannot be used in the insert value.

DSE1301=Select statement can yield only one row.

DSE1303=Invalid column {0} has been used in trigger action for table {2}.

DSE1304=Invalid column for trigger specified in the trigger statement.

DSE1305=Primary or Unique column should be autoincremental.

DSE1306=Cannot insert a record with HASRecord columnvalue equal to false.

DSE1307=Target Specification cannot be null

DSE1308=Invalid user {0}

DSE1309=Invalid password

DSE2001=Now we have to insert record.

DSE2002=Key is not valid.

DSE2003=Record number {0} passed is deleted.

DSE2004=Record is not present.

DSE2005=Get the record from this location {0}.

DSE2006=Record is partial.

DSE2007=Record number is exceeding the total number of records {0}.

DSE2008=Key {0} value {1} pair not present.

DSE2009=Duplicate keys are not allowed.

DSE2010=Cluster is locked by another user.

DSE2012=Database is locked by another user.

DSE2013=Type {0}.

DSE2014=Column does not exist {0}.

DSE2015=There is no default index on table {0}.

DSE2016=Error while updating the indexinfortable {0}.

DSE2017=Index {0} already exists for table {1}.

DSE2018=Number of values are not equal to number of columns.

DSE2019=Iterator is at invalid position,1 : after last , -1 : before first.

DSE2023=Server is already closed {0}

DSE2025={0}.

DSE2027=Cannot add database with more than one file and numtfile support as FALSE.

DSE2028=Length of new files :: {0} initial size :: {1} increment factor is not possible with the database files u r trying to ADD.

DSE2029=Initial size must be greater than size of file exist.

DSE2030=File size :: {0},initial size :: {1}, increment factor :: {2} not possible.

DSE2031=Page Size is invalid.Valid range is 4k-32k.

DSE2032=Column not found : {0}.

DSE2034=Table is in use : {0}.

DSE2035=No index created in table : {0}.

DSE2036=Index already exist.

DSE2037=Table name found.

DSE2038=Element is deleted.

DSE2039=Node size should be greater than 2.

DSE2040=Index table not initialized.

DSE2041=NODE IS NONLEAF

DSE2042=NO Valid Key In Node

DSE2043=POSITION PASSED IS GREATER THAN ELEMENT COUNT == POSITION {0} AND ELEMENT COUNT {1}

DSE2044=SPLITTING WILL OCCUR NOW

DSE2045=BTree Can't Give Value Of This Column

DSE2046=Database version is not Compatible

DSE2047=Increament Factor can't be negative

DSE2048=Encryption key length can't be greater than 256

DSE2049=DatabaseName {0} contains illegal character.

DSE2050=Dead Lock Detected.

DSE2051=Either path specified for database or database name is too long which exceeds OS limits.

DSE2052=Transaction is active - Unable to set isolation level - First commit or rollback.

DSE2053=Cursor can not be declared without a procedure.

DSE2054={0} is not supported in one dollar db.

DSE3513=Type of iterator is not Updatable

DSE3514=Invalid Column {0} for table {1}

DSE3515=Column {0} Not Found in Mapping

DSE3516=Table {0} Does Not Exist In Mapping

DSE3517=Invalid Column {0}

DSE3518=Position of Iterator Not Initialised

DSE3519=Illegal Column Type

DSE3520=Value Of Column {0} Not Found in All Iterators

DSE3521=View Tables from SQL Hierarchy does not match fully with the Plan Hierarchy

DSE3522=Type Of Iterator {0} is not initialized

DSE3523=Illegal Call to Iterator

DSE3524=Clone not supported.

DSE3526=Table {0} does not lie in plan hierarchy.

DSE3527=Cost not caculated for Condition {0}.

DSE3528= ? is not allowed in Order BY and Group By Clause

DSE3529=Either of joinlevelorder {0} and singletablelevelorder {1} should present.

DSE3530=Condition {0} not solvable on any plan.



DSE3531=Cost not calculated for this plan.

DSE3532=Listener is not supported in case of set operator and subquery.

DSE3533=Columnnames specified in from sub query and derived column list are not equal in length.

DSE3534=Alias name should be specified with aggregate/expressional or scalar functional columns.

DSE3535=Duplicate columns specified in from subQuery or view.

DSE3536=Aggregate columns cannot be present in onCondition.

DSE3537=Relation {0} does not belong to any plans.

DSE3538=Condition {0} can not be shifted to single table level.

DSE3539=Wrong event type {0} passed.

DSE3540=HasRecord column cannot be used in select list when group by is present.

DSE3541=A column has been specified more than once in the order by list. Columns in the order by list must be unique.

DSE3542=Ambiguous column naming in select list.

DSE3543=ORDER BY items must appear in the select list if the statement contains a UNION operator.

DSE3544=Order Column {0} not present in selectList of query involving set operator.

DSE3545=The ORDER BY position number {0} is out of range of the number of items in the select list.

DSE3546=HasRecord Column contains invalid table name {0}.

DSE3547=Condition {0} Execution Plan is not initialised.

DSE3548=Columns and their order should be equal in length.

DSE3549=Data type and size is not initialized for scalar function.

DSE3550=? is not allowed in expression in selectlist

DSE3551=Aggregate COlumns cacnot be present for insertion in insert statement.

DSE3552=Invalid Count value inside Top funtion.

DSE3553=Iterator is at Invalid status.

DSE3554=Child length can not be more than {0}.

DSE3555=Reference {0} Value is not found.

DSE3556=Execution Plan not initialized for table {0}.

DSE3557=Column {0} does not exist in columncharacteristics of table {1}.

DSE3558=Columns present in Comparison Predicate is Not Equal to 2.

DSE3559=COLUMNS present in Comparison Predicate is Null.

DSE3560=Type of Aggregate function is not initialised.

DSE3561=Aggregate Function Quantifier type is not initialised.

DSE3562=References and Values passed are not equal in length.

DSE3563=Reference {0} not found in {1}.

DSE3564=Mapping is not proper intialized.

DSE3565=Inappropriate type in {0}

DSE3566=Having clause can not be given without any aggregate columns or group by

DSE3570=Invalid data type {0}.

DSE3571=Collator of column {0} and column {1} does not match, hence we cannot compare

DSE3572=No Proper Comparator...data type 1 {0} data type 2 {1}

DSE3573=No value found for Parameter Name {0}

DSE3574=All column selected.

DSE3575=InComparable DataType {0}

DSE3576=Column {0} used in NATURAL join cannot have qualifier

DSE3577=CROSS JOIN IS ONLY POSSIBLE

DSE3802=Not found.

DSE3804=Npe from variable column.

DSE4104=Error converting data type varchar to float.

DSE4106=Cannot insert in {0} data type value.

DSE4107=Invalid data type {0} is passed for argument {1} in function {2}.

DSE4108=Invalid data type {0} is passed in function {1}.

DSE4109=Invalid operator for data type. operator EQUALS minus, type EQUALS {0}.

DSE4111=The {0} aggregate operation cannot take a {1} data type as an argument.

DSE4112=Syntax error converting the {0} value {1} to a column of data type {2}.

DSE4113=Datatype has been set to {0} type for decimal value.

DSE4114=The BLOB,CLOB data types cannot be compared or sorted, except when using is null operator.

DSE4115=Invalid column name {0}.

DSE4116=Iterator not aligned to any valid location. (First call beforefirst() or first()).

DSE4117=Iterator not aligned to any valid location. (First call afterlast() or last()).

DSE4118=Object do not belong to supported datatypes.

DSE4119=The {0} requires one argument.

DSE4120=Object is an instance of ignorevalue.

DSE4121=Syntax error converting the {0} value {1} to a column of data type {2} and value {3}.

DSE4122=Cannot perform an aggregate function on an expression containing an aggregate or a subquery.

DSE4123=Cannot Move to the key = {0}

DSE4124=Iterator is not Initialized.

DSE4125=Column Not Found.

DSE4126=Method not supported.

DSE4127=There are no Childs for Class -> {0}.

DSE5001=User Defined Function {0} not supported

DSE5002=Invalid Grantee in Grant/Revoke Statement

DSE5003=Invalid Event (Listener fired with SELECT Type Event with Operation Type update)

DSE5004=Invalid Operation Type in Event

DSE5005=Can't specify autoIncrement for more than one column

DSE5006=Can't specify default option for autoIncrement column

DSE5007=Invalid data type for autoIncrement

DSE5008=There are no primary or candidate keys in the referenced table {0} that match the referencing column list in the foreign key {1}.

DSE5009=No primary or unique constraint present

DSE5010=Schema contains some Database Objects i.e. tables, views, Domains, Routines, Triggers etc. Can not be dropped

DSE5011=Invalid user name {0}

DSE5012=Cannot have more than one null call clause

DSE5013=Cannot have more than one transformgroupspecification

DSE5014=Routines does not support multiple privileges

DSE5015=Not a valid view both materialized and INTO tablename should be specified

DSE5016=Invalid catalog name for table

DSE5017=Datatypedescriptor and columns table result mismatch

DSE5018=Contains columns refering domain

DSE5019=Current user is not authorized to drop routine

DSE5020=Record to be deleted not present

DSE5021=Language other than SQL is not supported

DSE5022=Null-Call clause shall not be specified in case of SQL invoked procedures

DSE5023=No owner exist for Schema {0}

DSE5024=Procedure Exist for same name and same number of parameters

DSE5025=Catalog/Schema of routine name and specific name should be same

DSE5026={0} has no references rights on table {1}

DSE5027=Result Set is Closed

DSE5028=View {0} is not a materializedview.

DSE5029=Problem in case of delete event on RecordSetBufer {0}.

DSE5030=Problem in case of insert event on RecordSetBufer {0}.

DSE5031=Insert not allowed for query {0}.

DSE5032=Error Occured while setting values for Record {0}.

DSE5034=Column {0} is not updatable in query {1}.

DSE5036=There is already one record in update state.

DSE5037=Deleterow cannot be called for record yet to be inserted.

DSE5038=Loadrecordforidentity cannot be called for record yet to be inserted.

DSE5040=Ignore Values cannot be passed in this sequence for client parameters.

DSE5041=Number of Parameter Infos for client parameters {0} are more than parameters in query {1}.

DSE5042=Autonumber value cannot exceed {0}.

DSE5045=Invalid event Type {0}.

DSE5046=Role Dependency graph is not initialized properly.

DSE5047=Privilege Dependency graph is not initialized properly.

DSE5101=Cannot drop view {0}, because this view is a materailized view.

DSE5102=Cannot drop materailized view {0}, because this view is not a materailized view.

DSE5502=Invalid database name {0}.

DSE5503=You can not change the ISOLATION LEVEL.

DSE5504=SEQUENCE {0}.NEXTVAL exceeds maxvalue and cannot be instantiated.

DSE5505=SEQUENCE {0}.NEXTVAL goes below minvalue and cannot be instantiated.

DSE5506=Sequence already exists {0}.

DSE5507=Duplicate or conflicting {0} specifications.

DSE5508=Increment must be a non-zero integer.

DSE5509=Minvalue cannot be less than {0}.

DSE5510=MAXVALUE Can not be greater than {0}

DSE5511=Order not Supported

DSE5512=MINVALUE must be less than MAXVALUE

DSE5513=START WITH should lie between MINVALUE and MAXVALUE

DSE5514=Absoulte of the INCREMENT value must be less than or equal to MAXVALUE minus MINVALUE and Should not be "0"

DSE5515=MAXVALUE cannot be made to be less than the current value

DSE5516=Sequence {0} does not exist

DSE5517=MINVALUE cannot be made greater than the current value

DSE5518=Couldn't Move to Keys Specified.

DSE5519=Database {0} already connected.

DSE5520=Date {0} should lie between {1} and {2}.

DSE5521=Pointer is not set after referesh.

DSE5522=Database is in use.

DSE5523=Cannot create directory on specified path.

DSE5524=Remove ChildServerSession.

DSE5525=START WITH cannot be less than minvalue.

DSE5526=START WITH cannot be greater than maxvalue.

DSE5528=No record found for keys specified.

DSE5529=U cannot get parentsessionid without starting any save point.

DSE5530=Sessionid list contains only one element.No Start save point exists.

DSE5531=SessionID's are not hidden. Call hideSavepoint first.

DSE5532=StartSavePoint NOT Started or no of start savepoints started are less than 2.

DSE5533=Allow parallelsavepoint NOT started yet.

DSE5534=Commit last savepoint first.

DSE5535=Ignore parallel savepoint first.

DSE5536=No element removed from list.

DSE5537=Invalid instance of iterator {0}.

DSE5538=No referenced table found for column {0}.

DSE5539=Databasefile {0} is either removed or deleted from path.

DSE5540=Invalid constraint name or cannot defer a constraint that is not defferable.

DSE5541=Invalid constraint mode {0}.

DSE5542=Can not specify {0} more than one time.

DSE5543=System Database is not properly created, delete the directory from the path first.

DSE5544=UserDatabase is not properly created,drop the database first.

DSE5545=Value {0} is out of range of {1} data type.

DSE5546=Value larger than specified precision allows for this column.

DSE5547=Data type {0} is not convertible into data type {1}.

DSE5548=Column name {0} appears more than once in the result column list.

DSE5549=Invalid date-time result.

DSE5551=XML file at specified Path {0} not found.

DSE5552=Blob data file at specified Path {0} not found.

DSE5553=Clob data file at specified Path {0} not found.

DSE5554=XML File Write Exception.

DSE5555=Blob data file Write Exception.

DSE5556=Clob data file Write Exception.

DSE5557=Error while getting LOB data.

DSE5558=Database is in Read Only Mode.

DSE5560=Schedule {0} to be dropped does not exist.

DSE5561=Schedule {0} already exists.

DSE5562=Database {0} for which scheduler is added does not exist.

DSE5564=Database to be backed up is in use.

DSE5566=Database {0} to be restored already exists.

DSE5567=Database {0} to be backed up already exists.

DSE5568=Database {0} can not be restored.

DSE5569=Database can not be restored with name {0}.

DSE5570=Database {0} can not be backed up.

DSE5571=Database can not be backed up with name {0}.

DSE5572=Invalid path {0}.

DSE5573=SystemDatabase and database {0} not compatible.



DSE5574=Can't take backup as Source Path And Destination Path are same.

DSE5575=User {0} is not having privilege to take Backup.

DSE5576=Database name can not be {0}.

DSE5577=Cannot get Connection as Backup is under process.

DSE5578=Adding Schedule is not supported on this version.

DSE5579=Backup correpted - some files are either removed or deleted from path - Start backup after removing all files from the path.

DSE5581=Access denied. Save point allready started.

DSE5582=Exception: Trigger in recursion exceeding count 16.

DSE5583=Exception: Statement Trigger in recursion exceeding count 16.

DSE5584= Table has been already dropped.

DSE5590= Feature{0} not supported below Version3.0

DSE5591= Version incompatible as executable jar doesn't support Backward Compatibility

DSE6001={0} is non-comparable data type, cannot be used in Distinct/Predicates/Union/Intersect queries.

DSE6004=Invalid type {0} in Like predicate.

DSE6005=Having Clause should not contain ContainsClause

DSE6006=ORDER BY columns must appear in the select list if DISTINCT is contained in Select List.

DSE6007=Expression {0} cannot be specified in search condition.

DSE6008=Sequence number not allowed here.

DSE6009=SubQuery is not allowed in order by clause.

DSE6010=FullTextIndex name should be given if index exists on multiple column.

DSE6011=Contains clause not supported for queries involving more than one table or view.

DSE6012=Select Query is not supported in select list.

DSE6013=Sequence is not allowed in Group by and order by clause

DSE7001=Foreign key {0} has implicit reference to object {1} which does not have a primary key defined on it.

DSE7002=Number of referencing columns in foreign key differs from number of referenced columns, table {0}.

DSE7003=Foreign key {0} references invalid table {1}.

DSE7005=Invalid column(s) {0} specified in CHECK constraint {1}.

DSE7006=Foreign key {0} references invalid column {1} in referenced table {2}.

DSE7007=Foreign key {0} references invalid column {1} in referencing table {2}.

DSE7008=Cannot alter table {0} because this table does not exist in database.

DSE7009=Cannot drop the table {0}, because it does not exist in the database.

DSE7010=Cannot drop the view {0}, because it does not exist in the database.

DSE7051=[clientstatement]you cannot SET client parameters for the query {0}.

DSE7052=[rsb]one record is already taken for insertion FIRST COMMIT that and then try again.

DSE7053=[rsb]Invalid call.

DSE7054=[rsb]you cannot call update row on inserted record.

DSE7055=[rsb]there is already one record in UPDATE state.

DSE7056=[rsb]this record was not updated.

DSE7057=Cannot drop the index {0} from table {1}, because it does not exist in the database.

DSE7058=User {0} already exists in the database.

DSE7059=Cannot drop the user {0}, because it does not exist.

DSE7060=Cannot create an index on {0}, because this table does not exist in database.

DSE7061=Udt support not available.

DSE7062=Duplicate username listed.

DSE7063=Invalid interval {0}.

DSE7064=Column count cannot be greater than one.

DSE7065=Cannot drop the trigger {0}, because it does not exist in the database.

DSE7066=Invalid grantee in grant statement.

DSE7067=Option {0} can be defined at most once.

DSE7068=HAS RECORD cannot be used in Group by clause.

DSE7069=Illegal Mapping.

DSE7070=Count cannot be less than or equal to zero, specified count is {0}.

DSE7071=Chained table info for table {0} not found.

DSE7072=Table details mapping does not contain table details {0}.

DSE7073=Table {0} already has a primary key defined on it.

DSE7074=More than one key specified in column level FOREIGN KEY constraint, table {0}.

DSE7075=ALTER TABLE ADD CONSTRAINT statement conflicted with FOREIGN KEY constraint {0}. The conflict occurred in table {1}.

DSE7076=Catalog {0} does not exist.

DSE7077=Column names in each view must be unique. Column name {0} in view {1} is specified more than once.

DSE7078=ALTER TABLE DROP COLUMN statement conflicted with FOREIGN KEY constraint {0}. The conflict occurred in table {1}, column {2}.

DSE7079=ALTER TABLE ALTER COLUMN SET DEFAULT statement failed because column {0} does not exist in table {1}.

DSE7080=ALTER TABLE ALTER COLUMN DROP DEFAULT statement failed because column {0} does not exist in table {1}.

DSE7081=ALTER TABLE DROP COLUMN statement conflicted with VIEW {0}. The conflict occurred in table {1}, column {2}.

DSE7082=ALTER TABLE DROP COLUMN statement conflicted with TRIGGER {0}. The conflict occurred in table {1}, column {2}.

DSE7083=ALTER TABLE DROP COLUMN statement conflicted with CHECK constraint {0}. The conflict occurred in table {1}, column {2}.

DSE7084=ALTER TABLE ADD CONSTRAINT statement conflicted with UNIQUE constraint {0}. The conflict occurred in table {1}.

DSE7085=ALTER TABLE ADD CONSTRAINT statement conflicted with PRIMARY KEY constraint {0}. The conflict occurred in table {1}.

DSE7086=ALTER TABLE ADD CONSTRAINT statement conflicted with CHECK constraint {0}. The conflict occurred in table {1}.

DSE7087=ALTER TABLE ADD COLUMN statement conflicted with PRIMARY KEY constraint {0}. ALTER TABLE only allows columns to be added that can contain nulls or have a DEFAULT definition specified. Column {1} cannot be added to table {2} because it does not allow nulls and does not specify a DEFAULT definition.  
KEY constraint {0}. The conflict occurred in table {1}.

DSE7088=ALTER TABLE ADD COLUMN statement conflicted with UNIQUE KEY constraint {0}. The conflict occurred in table {1}.

DSE7089=ALTER TABLE ADD COLUMN statement conflicted with CHECK CONSTRAINT {0}. ALTER TABLE only allows columns to be added that can contain nulls or have a DEFAULT definition specified. Column {1} cannot be added to table {2} because it does not allow nulls and does not specify a DEFAULT definition.

DSE7090=Database {0} specified is either removed or deleted from path.

DSE7091=HAS RECORD ColumnDetail contains invalid table name(s) {0}.

DSE7092=Duplicate column name {0}.

DSE7093=Alias Name Should Be specified with Aggregate/Expressional or Scalar Functional columns.

DSE7094=Invalid boolean type {0}.

DSE7095=ColumnNames specified are not equal in length.

DSE7096=Node size should be greater than 2.

DSE7097=Invalid Instance Of Iterator {0}

DSE7098=Invalid status in Top Iterator

DSE7099=Iterator for table not found

DSE8000=Position not set

DSE8001=Existing value and delete event fired

DSE8002=Specified System table {0} not found

DSE8003=Table details mismatch

DSE8004=Column name for index passed {0} is not Found

DSE8005=Index not found of reference {0} in references {1}

DSE8006=Invalid join type {0}

DSE8007=Query passed is null.

DSE8008=Column {0} is not a functional column.

DSE8009=Invalid number

DSE8010=DROP TABLE statement conflicted with VIEW {0}. The conflict occurred in table {1}.

DSE8011=DROP TABLE statement conflicted with TRIGGER {0}. The conflict occurred in table {1}.

DSE8012=DROP TABLE statement conflicted with CHECK constraint {0}. The conflict occurred in table {1}.

DSE8013=DROP TABLE statement conflicted with FOREIGN KEY constraint {0}. The conflict occurred in table {1}.

DSE8014=DROP VIEW statement conflicted with VIEW {0}. The conflict occurred in view {1}.

DSE8015=DROP VIEW statement conflicted with TRIGGER {0}. The conflict occurred in view {1}.

DSE8016=Inconsistent data type, value should be characterstringliteral

DSE8017="DATE" Keyword must specify in default clause for column {0}

DSE8018=Inconsistent data type, value should be booleanliteral for column {0}

DSE8019=Default value is too large for column {0}

DSE8021=Method {0} is Wrongly Called For SelectedColumnIterator

DSE8022=Invalid timestamp data type value. Timestamp format must be yyyy-mm-dd hh:mm:ss

DSE8023=Invalid time data type value. Time format must be hh:mm:ss.

DSE8024=Invalid date data type value. Date format must be yyyy-mm-dd.

DSE8025=Acess Denied Do not have Select privileges on column {0} of table {1}

DSE8026=Invalid or Parameterised queries

DSE8027=Invalid column(s) {0} specified in trigger condition for trigger {1}

DSE8028=Invalid column(s) {0} specified in trigger statement for trigger {1}

DSE8029=Invalid default clause for data type {0}

DSE8030=Object type {0} is not compatiabale with privilege type {1}

DSE8031=Row and table simuntaneoulsy can not be present in trigger definition

DSE8032=Old or new values alias list can not be specified for Statement trigger

DSE8033=Invalid country code {0}

DSE8034=Invalid Language code {0}

DSE8035=Loop statement does not contain terminating statement

DSE8036=Beginning label should be specified in case ending label used

DSE8037=Beginning label and ending label used should match for a statement

DSE8038=Invalid label used in leave or iterate statement

DSE8039=Duplicate variable declaration

DSE8040=Cursor {0} already closed

DSE8041=No value found for parameter {0}

DSE8042=Only prepared statement like functionality supported in case of statements other than CALL statement

DSE8043=Only NEXT operation allowed on non scrollable curosr

DSE8044=Cardinality mismatch between query specification and fetch target list

DSE8045=Invalid Variable name in fetch target list

DSE8046=Cursor in non updatable

DSE8047=Table name {0} not present in query specification of cursor {1}

DSE8047=Variable {0} not declared in procedure {1}

DSE8049=The maximum size for all index columns can not exceed {0} bytes. The index {1} with size {2} bytes can not be created.

DSE8050=such column list already indexed.

DSE8051=Cursor {0} either not opened or has been closed

DSE8052=Either EXECUTE not called or command does not produce result set

DSE8053=Either URL or dataSourceName must be specified to create connection

DSE8054=No such DataSource {0}

DSE8055=Data Source name can not be null

DSE8056=Transaction Isolation Level passed is Not Valid

DSE8057=Invalid ResultSetType {0}

DSE8058=Invalid concurrency {0}

DSE8059=UDTs are not supported

DSE8060=Parameter Index can not be less than 1,passed parameter index is {0}

DSE8061=Not A Valid Direction {0}

DSE8062=In case of type Forward only, only valid direction is fetch forward

DSE8063=Fetch Size can not be less than 0

DSE8064=UpdateRow called while curosr is on newly insert row

DSE8065=Invalid cursor position

DSE8066=Before Calling insertRow, cursor must be on insert row

DSE8067=Auto column value can not be NULL

DSE8068=Do not have rights to revoke privileges on object {0}

DSE8069=length/precision/largeobjectlength can not be zero for datatypes

DSE8090=Cursor statements cannot be executed through Daffodil DB Shell/Daffodil DB Browser.

DSE8091=Invalid column Indexes for GeneratedKeys

DSE8092=Invalid role {0}.

DSE8093=The role can not be granted to itself or any of its applicable roles

DSE8094=Invalid role/user {0}.

DSE8095=No such role authorization descriptor found to delete.

DSE8096=Role to be granted does not lie in applicable role of grantor {0} or doesn't have WITH ADMIN OPTION.

DSE8097=Currently active User {0} can not be dropped

DSE8098=Role {0} is currently active/Lies in applicable roles.

DSE8100=Incompatible data type {0} and {1}.

DSE8101=Syntax error converting data type {0} to {1}.

DSE8102=Cannot create index on view

DSE8103=object can not be created with name greater than 128 characters

DSE8104=System or Administrator user can not be dropped

DSE8105=Do not have CREATE/DROP user permission.

DSE8106=Syntax error converting {0} data type to date data type.

DSE8107=Length {0} exceeds premissable values for length or is invalid.

DSE8108=Invalid check constraint condition {0}

DSE8109=Invalid column name/references in triggered action



DSE8110=Do not have DROP permission on {0}

DSE8111=Invalid start index passed to the {0} function. Start index can not be less than one.

DSE8112=Invalid length parameter passed to the {0} function. Length can not be negative.

DSE8113=repeat counter too large.

DSE8114=Invalid counter passed to the {0} function. Counter can not be negative.

DSE8116=Cannot alter domain {0} because this domain does not exist in database.

DSE8115=can not add column as check constraint applied is violated.

DSE8117= {0} length can not exceed {1}

DSE8118=Problem in creating file {0} either it contains illegal characters or blank spaces

DSE8119=Invalid columnName or indexName used in ContainClause{0}

DSE6002=Blank or Stop words cannot be used for searching.

DSE6003=For Update Option cannot be used if query contains multiple tables, view, group by or having clause.

DSE8120=Specified class is not valid {0}

DSE8121=wrong name: {0}

DSE8122=Specified jar name is not valid {0}

DSE8123=Number of parameters in procedure declaration does not match with number of parameters in java method

DSE8124=Datatype mismatch in parameters at {0}

DSE8125=Data Type not supported

DSE8126=Method {0} does not exist in {1}

DSE8127=Class does not have <init>(java.sql.Connection) constructor

DSE8128 =Contains Clause of the query contained only ignored words

DSE8129={0} must be the current user.

DSE8131=Role {0} not granted to user {1}

DSE8132=Do not have sufficient privileges for select

DSE8133=CurrentUser/CurrentRole does not have execute privileges on {0}.

DSE8134=Incorrect alias {0}

DSE8135=Alias name not provided for column {0}

DSE8136=This Feature not supported in version {0}

DSE8137=Do not have sufficient privileges for create schema with {0} catalog.

DSE8138=Contain clause not supported in searched condition.

DSE8139={0}

DSE8141=Number of arguments exceed actual number of columns

DSE8142=Parameterized statement not allowed in triggers.

DSE8143=User/Role {0} has Insufficient privileges.

DSE8144=Length of column {0} should be equal to 1031.

DSE8145=Error while executing {0} method

DSE8146="TIME" Keyword must specify in default clause for column {0}

DSE8147="TIMESTAMP" Keyword must specify in default clause for column {0}

DSE8148=Value larger than specified precision not allowed for column {0}

DSE8149=Data type value is invalid for column {0} / Date format must be yyyy-mm-dd.

DSE8150=length/precision/largeobjectlength of datatype can not zero for column {0}

DSE1186=Cursor {0} is not scrollable

DSE8151=Do not have CREATE/DROP database permission for {0} session.

DSE8152=Contains clause not allowed in view definition.

DSE8153={0} in triggered action for trigger {1}

DSE8154=More than one primary key constraint can not be applied on column {0}

DSE8155=More than one unique key constraint can not be applied on column {0}

DSE8156=More than one not null constraint can not be applied on column {0}

DSE8157=Both primary key and unique key constraint's can not be applied on column {0}

DSE8158=Column {0} should exist on search condition

DSE8159=Parameters statement not allowed in View {0}

DSE8160=Exception: Procedure in recursion exceeding count 32

DSE8161=Unique/Primary key constraint already exist on table {0}

DSE8162=Query expression's length {0} can not exceed "4192" characters for view {1}.

DSE8163=ALTER TABLE DROP COLUMN statement conflicted with PRIMARY/UNIQUE/FOREIGN KEY constraint {0}. The conflict occurred in table {1}, column {2}.

DSE8164=Parameterised query not allowed in Stored Procedure.

DSE8165=Default value's length not supported up to "1024".

DSE8166=Current session does not have rights on schema {0}

DSE8167=Grantor {0} does not have any privilege with grant option on object {1}.

DSE8168=Can not create default schema {0}.

DSE8169=Can not drop default schema {0}.

DSE8170=Result of repeat function exceeds the maximum length of char data type.

DSE8171=Constraint {0} should be less than 128 characters

DSE8172=Parameterized statement is not allowed in domains.

DSE8173=Invalid parameter {0}.

DSE8174=Aggregate function is not allowed in procedure call.

DSE8175=ColumnName/VariableName {0} is invalid/not declared.

DSE8176=Current user {0} must have references rights on column {1} of table {2}.

DSE8177=The value you entered is not consistent with the data type or length of the parameter.

DSE8178=Datatype {0} should be predefined datatype.

DSE8179=User {0} must be the database owner OR admin user.

DSE8180=Database can not be created with user {0}.

DSE8181=Database owner {0} can not be dropped.

DSE8182=Procedure {0} does not exist in databse.

DSE8183=Can not declare variable {0} with same name.

DSE8184=SubQuery not allowed in procedure parameters.

DSE8185=Role {0} does not exist in database.

DSE8186=Trigger table alias does not match with alias {0} used in when condition.

DSE8187=Can not add multiple unique OR primary key constraint on same column in table {0}

DSE8188=Acess Denied. Do not have Usage privileges on domain {0}

DSE8189=Schema statements (Create, Alter, And Drop) are not allowed in procedures.

