

The Instant Data Warehouse

User Guide

Data Warehousing Utilities

Version 1.3
8th February 2003
Author: Peter Nolan
Peter@peternolan.net

Table of Contents

1. CHANGE CONTROL LOG.....	3
2. AUDIENCE	4
3. WHY HAVE DATA WAREHOUSING UTILITIES?	5
4. WHAT UTILITIES ARE AVAILABLE?	6
5. WHAT COMPONENTS ARE COMMON ACROSS THE IDW?.....	6
5.1. Tables Required in Target Database	7
6. PARAMETERS	8
6.1. Definition of Parameters	8
7. THE DATA TRANSFER UTILITY.....	12
7.1. Why Have the Data Transfer Utility?	12
7.2. What Does the Data Transfer Utility do?.....	13
7.3. How Does the Data Transfer Utility Determine Primary Keys?	13
7.4. Tables Required in Target Database	13
7.5. Examples of invoking the Data Transfer Utility	14
8. BATCH MAINTENANCE UTILITY	15
8.1. Why Have the Batch Maintenance Utility?	15
8.2. What Does the Batch Maintenance Utility Do?	15
8.3. Tables Required in the Target Database	16
8.4. Examples of Invoking the Batch Maintenance Utility	16
9. SQL STATEMENT PROCESSOR UTILITY.....	17
9.1. Why Have the SQL Statement Processor Utility?	17
9.2. What Does the SQL Statement Processor Utility Do?	17
9.3. Tables Required in the Target Database	17
9.4. Examples of Invoking the SQL Statement Processor Utility.....	17
10. THE SQL SERVER 2000 DDL GENERATOR UTILITY	18
10.1. Why Have The SQL Server 2000 DDL Generator Utility?	18
10.2. What Does The SQL Server 2000 DDL Generator Utility Do?	18
10.3. Tables Required in the Target Database.....	18
10.4. Example of Invoking The SQL Server 2000 DDL Generator Utility	21
11. ERROR MESSAGES	22
11.1. Notes on Error Messages.....	28

1. CHANGE CONTROL LOG

#	Date	Name	Description
1.0	12/1/2003	P. Nolan	Initial version for publication with the Instant Data Warehouse.
1.1	20/1/2003	P. Nolan	Addition of the following utilities: <ul style="list-style-type: none">• CDWBM01/02 – Batch Maintenance Utilities• CDWU002 – Batch processing of SQL Statements against ODBC data source names.• CDWU003 – Generate SQL Server 2000 table/view DDL from a definition held in an ODBC data source.
1.2	30/1/2003	P. Nolan	Error and audit messages not being properly issued when running on Oracle.
1.3	8/2/2003	P. Nolan	Addition of CDWU004 – Generate Oracle table/view DDL from a definition held in an ODBC data source.

2. AUDIENCE

The intended audiences for the Instant Data Warehouse User Guide for Utilities are:

- Customers of the Instant Data Warehouse.
The utilities surrounding the Instant Data Warehouse are documented in this document rather than in the User Guide of the Instant Data Warehouse. This is to facilitate the free distribution of the utilities without doubling up on documentation..
- Technical developers who choose to use Data Warehousing Utilities without using the Instant Data Warehouse.

This document is intended to be read by IT Professionals who are familiar with using command line utilities in Unix or the windows console interface.

3. **WHY HAVE DATA WAREHOUSING UTILITIES?**

Every data warehousing project requires some functions to be available or performed that are very common across many projects. For example:

- The need to transfer data from one location to another:
- The need to know a 'batch processing date' that is separately maintained from the system date.
- The need to protect batches from being accidentally double processed. A not uncommon incident that almost always requires a restore.
- The need to run and re-run many DDL scripts during development.
- The need to be able to easily maintain sets of test data without needing to keep lots of database images, which each need to be maintained in the face of change.

In most cases in the past we have typically written tools/scripts etc for each site or just live with whatever the database manager provided and the local tools of the organisation. Usually these things are pretty inadequate and using them consumes a lot of time that could be better spent actually developing code.

Take for example the simplest task of adding a column to a table and changing another column (that is used in many places) from say varchar (20) to varchar (255). Consider what is involved in this trivial change. You have to change the table DDL, view DDL, test data, ETL code, and try it all out again. Each hand made change introduces the chance for making a mistake.

So, it would be really handy to be able to:

- Make the change to the DDL/View in a place that is really quick.
- Generate all the DDL and run it quickly and easily (as a command).
- Reload the test data for the affected tables.
- Add new values to the new column in the test data.
- Unload the test data and store it away again.
- Make as little change to the ETL as possible.
- Minimise the amount of effort required to re-test the ETL with the new column and changed column and know that it is working.

In response to all these issues I developed these utilities as a part of the development of the Instant Data Warehouse. Since they are very useful outside the Instant Data Warehouse environment I am choosing to package the utilities separately and make them freely available to any one who wants to use them. This is both a small contribution back to the data warehousing community and a little 'low cost advertising' for the Instant Data Warehouse.

Since the utilities are written in C++/ODBC and tested on windows 2000 there will be quite a population of people who could benefit from these tools. If you know other data warehousing professionals who would like to use these tools please feel free to pass along my website address. I recommend that new users download the latest version of the Utilities.

If you know other utilities that would be useful feel free to email me with requests. The ones that have high demand and are really useful I will give consideration to developing. Given the foundation of the sets of classes developed to support the Instant Data Warehouse new utilities are easy and fast to develop and I expect that new utilities will be added on a regular basis.

Best Regards

Peter Nolan

4. WHAT UTILITIES ARE AVAILABLE?

The current utilities that are available are as follows:

1. The Data Transfer Utility.
2. The Batch Maintenance Utility
3. The SQL Statement Processor Utility.
4. The SQL Server 2000 DDL Generator Utility.
5. The Oracle 8/9 DDL Generator Utility.

Each of these utilities will be discussed in the following sections.

5. WHAT COMPONENTS ARE COMMON ACROSS THE IDW?

The Instant Data Warehouse has a number of components that are common. They are:

1. Parameters and parameter management.
The Instant Data Warehouse uses a common command/parameter interface for all programs. This means that all programs are able to understand all parameters and no valid parameter passed to a program is rejected as 'invalid' because it is not needed. This means you don't have to worry much about any extra parameters and which ones to delete. Because of this I will document all parameters in one place and only note those parameters that are non-standard with the specific utility.
2. Error Messages and error message management.
It is not possible to turn off the error messages, as much as some people might want to. You can point the error messages to different outputs as you wish. Keeping track of error messages is important in that they will indicate problems within the Instant Data Warehouse. Because the error messages are common across all components the error messages for the utilities include the error messages for the Instant Data Warehouse.
3. Auditing and Audit trail management.
It is possible to turn auditing off. When auditing is turned on for a program the program will issue a message specifying the number of rows/records read/written for every file/table. This is particularly handy to make sure that all records made it from source to target. It also helps keep track of growth in the number of records flowing into the data warehouse.
4. ODBC access and file access.
All access to databases or files is via C++ classes. All database accesses are managed such that the calling program is unaware that it is talking to a database apart from the fact that it builds SQL to pass into the class. The same is true for most file access. Further, the fact that the file access uses the same data structures to read/write data files as the ODBC classes use for reading/writing ODBC data sources also means that the classes for database access and file access communicate extremely efficiently through the common data structure. This is a part of the reason for the ease of construction of utilities.

5.1. Tables Required in Target Database

If you choose to write error messages and audit messages two tables are required to exist in the target database to receive the messages. These are the dw_message_table and the dw_audit_table. Example DDL for each of these tables is provided below.

```
create table dwh4.dw_audit_table
(  program_name      varchar      (0256)    not null    default 'unknown'
,  field_name        varchar      (0256)    not null    default 'unknown'
,  total_rows        integer                               not null    default 0
,  integer_total      integer                               not null    default 0
,  dollar_total       money                               not null    default 0
,  date_time          datetime                               not null    default current_timestamp,
)
on CDWOR01FG01
;
```

```
create table dwh4.dw_message_table
(  program_name      varchar      (0256)    not null    default 'unknown'
,  function_name      varchar      (0256)    not null    default 'unknown'
,  message_number     varchar      (0010)    not null    default 'unknown'
,  message_severity   varchar      (0001)    not null    default 'unknown'
,  message_text       varchar      (4000)    not null    default 'unknown'
,  date_time          datetime                               not null    default current_timestamp,
)
on CDWOR01FG01
;
```

6. PARAMETERS

This section documents the valid parameters for all programs. In general parameters that are provided to a program that are not needed are ignored. Parameters that are required by a program and not provided produce an error condition and stop processing.

6.1. Definition of Parameters

Parameter Name	Description and valid values
DBConnectionInParameter	<p>This is the ODBC DSN Connection information that is required to issue the connection to the source ODBC DSN. This varies from database manager to database manager and you should consult your particular database provider for details of the contents of this parameter.</p> <p>For SQL Server 2000 a valid connection parameter is as follows: DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba; PWD=password;DATABASE=CDWOR01DB01;</p> <p>For Oracle 8 a valid connection parameter is as follows: DSN=PNORADB1;SERVER=ORAGDB1.PETERLAP;UID=system; PWD=manager;</p> <p>For Access 2000 a valid connection parameter is as follows: DSN=ACCESSDB1;SERVER=PETERLAP;UID=admin;PWD=; DATABASE=ACCESSDB1;</p>
InCatalogName	<p>This is the 'Catalog' name that forms the first part of the fully qualified table name from which data will be read. For most databases this is the database name itself. For example if connecting with the above SQL Server 2000 connection string the CatalogName would be 'CDWOR01DB01'. This parameter is ignored for MS Access.</p>
InSchemaName	<p>This is typically the 'owner' of the table. It forms the second component of the fully qualified table name from which data will be read. This parameter is ignored for MS Access.</p>
InTableName	<p>This is the table/view name that forms the third component of the fully qualified table name from which data will be read.</p> <p>For most databases the Data Transfer Utility will issue an SQL statement of 'Select * from InCatalogName.InSchemaName.InTableName' in order to read data. The current exception in MS Access which does not support three part names. For MS Access the select built is only 'Select * from InTableName'.</p>
DBConnectionOutParameter	<p>This is the ODBC DSN Connection information that is required to issue the connection to the target ODBC DSN. By using different connection strings for source and target you can move data from one database to another.</p> <p>If the DBConnectionOutParameter is not set and the DBConnectionInParameter is set the DBConnectionOutParameter defaults to the DBConnectionInParameter.</p>
OutCatalogName	<p>This is the 'Catalog' name that forms the first part of the fully qualified table name to which data will be written. If OutCatalogName is not set and InCatalogName is set then OutCatalogName will default to InCatalogName. This parameter is ignored for MS Access.</p>
OutSchemaName	<p>This is typically the 'owner' of the table. It forms the second component of the fully qualified table name to which data will be written. If OutSchemaName is not set and InSchemaName is set then OutSchemaName will default to InSchemaName. This parameter is ignored for MS Access.</p>
OutTableName	<p>This is the table/view name that forms the third component of the fully qualified table name to which data will be written. If OutTableName is not set and InTableName is set then OutTableName will default to InTableName.</p>
WorkFileName	<p>This is the fully qualified name of a file that the server has access to that can be opened for reading and writing. Data is unloaded to the work file and then loaded</p>

	<p>into the target database. An open (read or write) is issued to this name. An example valid value is: d:\cdwor01\release\data\cdwf001.dat</p> <p>When loading data produced by CDWAT01/CL01 the parameter used to pass the file name to the program is this parameter. The file name to pass in is one of the file names that were used in the file name parameters below.</p>
CDWF001FileName	<p>This is the fully qualified name of a file that the server has access to that can be opened for reading and writing. This is the file that CDWAT01 writes attributed detail fact records to. It is only required for CDWAT01/CDWAG01. An example valid value is: d:\cdwor01\release\data\cdwf001.dat</p>
CDWF002FileName	<p>This is the fully qualified name of a file that the server has access to that can be opened for reading and writing. This is the file that CDWAG01 writes summarised fact records to. It is only required for CDWAG01. An example valid value is: d:\cdwor01\release\data\cdwf002.dat</p>
CDWF003FileName	<p>This is the fully qualified name of a file that the server has access to that can be opened for reading and writing. This is the file that CDWCL01 writes summary records which are to be inserted into the summary fact table to. It is only required for CDWCL01. An example valid value is: d:\cdwor01\release\data\cdwf003.dat</p>
CDWF004FileName	<p>This is the fully qualified name of a file that the server has access to that can be opened for reading and writing. This is the file that CDWCL01 writes summary records which are to be applied to the summary fact table as updates. It is only required for CDWCL01. An example valid value is: d:\cdwor01\release\data\cdwf004.dat</p>
SQLFileName	<p>This is the fully qualified name of a file that contains valid SQL which you want to process against a data source using the CDWU002 – SQL Batch Processor utility. The server must have read access to this file. It is only required for CDWU002. An example valid value is: d:\cdwor01\release\data\sqlcommand.dat</p>
InsertUpdateOption	<p>The insert update option indicates the behaviour of the insert/update process. The valid values and their meanings are as follows:</p> <ul style="list-style-type: none"> • InsertOnly (Default Value) Records are inserted into the target table. A unique index constraint violation is considered to be an error and processing will stop if a unique index constrain violation occurs. • InsertThenUpdate Records are inserted into the target table. If a unique index constraint violation occurs an update will be issued using the primary key of the table in the where clause for the update. • InsertThenDeleteInsert This function has been specifically added for Sybase IQ. The records are inserted into the target table. If a unique index constraint violation occurs a delete will be issued using the primary key of the table in the where clause for the delete. If the delete is successful then an insert will be issued. • UpdateThenInsert If the majority of rows are going to exist in the target database performing inserts first is a bit wasteful of processing time for large volumes. Update then insert allows you to issue an update first and then if the record is not found issue the insert. <p>This option is only valid for CDWU001.</p>
DataMovementOption	<p>This defines how the data is to be moved in this particular run of the program. The valid values and their meanings are as follows:</p> <ul style="list-style-type: none"> • Transfer (Default Value) Records are copied from the source table to the target table in the one invocation of the program. The work file is still used to contain data during the transfer. • Unload Records are unloaded from the 'In' Catalog,Schema.Table to the workfile. • Load Records are loaded from the work file to the 'Out' Catalog,Schema.Table. <p>This option is only required for CDWU001.</p>

LoadFactTable	This defines whether the table being loaded is a fact table produced by the Instant Data Warehouse. The default value is 'No' and the only valid value for the parameter is 'Yes'. The purpose of this option is to search through the column names in the work file looking for columns with a prefix of 'dk_' or 'DK_'. If it finds any column with such a prefix it changes the prefix to 'pk_' or 'PK_' to send it into the fact table. This option can be ignored unless you are loading fact tables. This option is only required for CDWU001.
DebugLevel	This defines the level of debugging for the run time code. It is expected to be an integer value. The code has a large number of print statements imbedded in it. If you pass a non-zero value as the DebugLevel these print statements will execute and log the program execution at a very detailed level. It is used to assist in debugging and support. You should not set this parameter unless requested to do so by technical support.
SourceDatabase	<p>This defines the type database from which data will be read. It is required because there are some differences in the ODBC statements that can be processed by each database. The valid values and their meanings are as follows:</p> <ul style="list-style-type: none"> • SQLSERVER2000 (Default Value) This specifies that the source database is SQL Server 2000. • MSACCESS This specifies that the source database is MS Access. Currently only MS Access 2000 has been tested though it is expected that Access 2002 will also work. • ORACLE8 This specifies that the source database is Oracle 8. • ORACLE9 This specifies that the source database is Oracle 9. <p>Note that to date only SQL Server 2000 and MS Access have been tested. Testing is in progress for Oracle 8 and 9.</p>
TargetDatabase	This defines the type database to which data will be written. Note that MS Access is supported as a target database however the tool is intended to be moving data into an RDBMS.
ErrorMessageOutput	<p>This defines where you would like error messages to be written. The valid values and their meanings are as follows:</p> <ul style="list-style-type: none"> • cerr (Default Value) This specifies that the error messages should be written the the standard C Error output whatever that might be on the server. • ErrorFile This specifies that the error messages should be written to error file that is specified in the ErrorMessageFileName parameter. • TargetDatabase This specifies that the error messages should be written to the database table accessed via the DBConnectionOutParameter. The table that will be written to is OutCatalogName.OutSchemaName.dw_message_table. <p>Note. If the Data Transfer Utility is unable to connect to the target database it will be unable to write error messages to the database if you specify TargetDatabase in this option. It is recommended that when performing initial testing error messages are written to cerr or the ErrorFile.</p>
ErrorMessageFileName	This is the fully qualified name of a file that the server has access to that can be opened for writing. This parameter is required if you specify that error messages should be written to the target database. An example valid value is: d:\cdwor01\release\data\errormessages.dat
Audit	This defines whether you would like audit records written for the processing that is performed. The audit records contain the name of the program, the function calling the audit processing, the column name or description of what is being audited, the number of rows being read, inserted, updated, deleted. The Audit process is required by the Instant Data Warehouse and so has been included in the Data Transfer Utility. The default value is 'No' and the only valid parameter value is 'Yes'. Audit messages are written to the table OutCatalogName.OutSchemaName.dw_audit_table.

As can be seen from the detailed documentation on parameters, the only specific parameters are those required to specifically identify files being written by the attribution, aggregation and consolidation processes. The way that many stars can be built is by having multiple invocations of each program using different parameters for the input table and output table.

7. THE DATA TRANSFER UTILITY

7.1. Why Have the Data Transfer Utility?

I have been working in data warehousing for some 12 years now. One of the things we have to do time and time again is to move data from one place to another, either as test data or in a re-runnable production environment. There are tools to do this, and the list of them seems to be getting longer. However, they all have their 'features' and 'shortcomings'. The ones with a lot of functionality are expensive and the ones that are cheap have little functionality.

One thing that many of them require is quite a bit of setup work to make it possible to move data from one place to another. For example, a lot of these jobs insist that you specify the movement from source table column to target table column on a column by column basis using the graphical front end tool.

For me, this was fine while I was setting up a re-runnable production job, but when I just wanted to grab hold of some test data I seem to constantly be fetching data using MS Access and then trying to push it into the target database. I always wanted something easier and this data transfer utility is it.

Lastly, the data transfer utility is used by the Instant Data Warehouse for loading fact tables. As such it has been built with all the 'extras' that are required within the Instant Data Warehouse.

I hope you find this utility useful.

7.2. *What Does the Data Transfer Utility do?*

As the name implies the Data Transfer Utility moves data from one place to another. Those places are ODBC Data Source Names in either a Windows (win32) or Unix environment or any data source that can be seen from a win32/Unix environment.

The functions performed are as follows:

1. Unload from an ODBC DSN to a self describing work file.
2. Load from the self describing work file into an ODBC DSN.

When loading you can specify the following behaviour.

1. Insert Only.
Insert only considers a unique index constraint violation to be an error and will stop processing if a unique index constraint violation occurs.
2. Insert then update.
In this case a unique index constraint violation causes an update using the table primary key to be used in the where clause.
3. Insert then delete insert.
This function has been specifically added for Sybase IQ. In Sybase IQ updates are much slower than a delete/insert. And since there is no logging in Sybase IQ the delete/insert does not cause excessive logging. Since it is available for Sybase IQ it has been made available for all databases.
4. Update then insert.
If the majority of rows are going to exist in the target database performing inserts first is a bit wasteful of processing time for large volumes. Update then insert allows you to issue an update first and then if the record is not found issue the insert.

7.3. *How Does the Data Transfer Utility Determine Primary Keys?*

To implement updates and deletes it is obvious the Data Transfer Utility requires knowledge of the primary keys of a table. However, in a data warehouse environment it is generally the case that views are being used as target 'tables'. Also, it is normal that tables have multiple unique index constraints on them that need to be used when performing select/update/delete processing on those tables.

Thus, using the underlying table 'primary key' would prove to be an inadequate method of determining the primary key of a table. This is one reason why ETL tools always allow the definition of keys inside the repository description of a table.

One of the design goals of the Instant Data Warehouse was to minimise the volume of 'metadata' required to control the processing being performed. Thus it is most likely that the Data Transfer Utility will target a 'view' of a table rather than the table itself.

In the Data Transfer Utility primary keys on a target table are indicated by the prefix 'pk_' (or 'PK_') in a column name.

If you specify that the Data Transfer Utility can perform a delete or an update at least one column of the target view/table must be prefixed with 'pk_' or the generated insert/update statement will not contain a valid column in the 'where clause'. This has proven to be a common mistake when first using the Data Transfer Utility.

7.4. *Tables Required in Target Database*

None.

7.5. Examples of invoking the Data Transfer Utility

As mentioned the program name is CDWU001 and all parameters are passed via the standard console parameters so common for DOS/Unix commands. The general syntax just being the program name and the parameters stored in a .bat or .cmd file. Note that you cannot use a new line between parameters. The commands below are formatted to make them easier to read.

An example command to transfer the table CDWOR01DB01.DWH4.customer_dim to CDWOR01DB01.DWH4.customer_dim_test_load1 is as follows:

```
CDWU001.exe
DBConnectionInParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=password;
DATABASE=CDWOR01DB01
InCatalogName=CDWOR01DB01
InSchemaName=DWH4
InTableName=customer_dim
DBConnectionOutParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=passwor
d;DATABASE=CDWOR01DB01
OutCatalogName=CDWOR01DB01
OutSchemaName=DWH4
OutTableName=customer_dim_test_load1
InsertUpdateOption=InsertThenDeleteInsert
DataMovementOption=Transfer
WorkFileName=D:\CDWOR01\RELEASE\data\testdata.DAT
ErrorMessageOutput=TargetDatabase
SourceDataBase=SQLSERVER2000
TargetDataBase=SQLSERVER2000
ErrorMessageFileName=D:\CDWOR01\RELEASE\data\testerrormessages.DAT
LoadFactTable=Yes
DebugLevel=0
Audit=Yes
```

However, the following command will also work.

```
CDWU001.exe
DBConnectionInParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=password;
DATABASE=CDWOR01DB01
InCatalogName=CDWOR01DB01
InSchemaName=DWH4
InTableName=customer_dim
OutTableName=customer_dim_test_load1
InsertUpdateOption=InsertThenDeleteInsert
DataMovementOption=Transfer
WorkFileName=D:\CDWOR01\RELEASE\data\testdata.DAT
ErrorMessageOutput=TargetDatabase
```

Notice how many fewer parameters can be provided when use is made of the defaulting behaviour of the parameters. Unfortunately, there is no getting around passing the ODBC DSN connection parameter. It could have been placed in a file however, it seems better to have it as a parameter so that the program can be called from command files accessing a variety of different sources. Since the password must be present in the file obviously read access to these command files should be restricted.

The target table in this case is customer_dim_test_load1. The first part of the DDL to create the table is as follows. Notice that the primary key is named pk_customer_dim. The Data Transfer Utility would also work if this were a view over the target table with a column named pk_customer_dim.

```
create table dwh4.customer_dim_test_load1
(   pk_customer_dim          integer          not null primary key
,   cust_code                varchar          (0006)    not null
```

8. BATCH MAINTENANCE UTILITY

8.1. Why Have the Batch Maintenance Utility?

One of the common errors in loading data into a data warehouse is the accidental loading of the same data twice, usually due to the human error of restarting the batch at the beginning of processing rather than at the point of failure. It's an easy mistake to make.

Another requirement which is often overlooked is that for any time variant information, such as type 2 dimensions, there must be a close date on the record being closed and open date on the new record. Many people use the system date. This causes problems in the event that there are multiple days worth of processing on the one day.

For example, if there is a failure in the data warehouse load on a Friday night it will not usually be corrected until the Monday morning. After the correction, the data for Friday, Saturday and Sunday must be processed. If the system date is used the date from and date to ranges on time variant information will not accurately reflect the date that the record was changed.

What is usually required, in a data warehouse that will be loaded on a 'daily' basis is a batch date parameter that can inform the Data Warehouse of the date that is to be used as the 'batch date'.

8.2. What Does the Batch Maintenance Utility Do?

The Batch Maintenance Utility performs three functions.

1. It checks to see if there is a current batch being run. If there is a current batch being run it ends with an error message and an error condition of 1.
2. If there is no current batch running it updates the default batch date by incrementing it one day as a default. If there is some other change that needs to be made to the batch date it can be made manually.
3. At the end of the batch the Batch Maintenance Utility signals that the batch is completed successfully. Thus allowing a new batch to be processed.

By placing the program CDWBM01 at the beginning of any specific batch for a specific schema and testing for a return code of 0 prior to starting your batch for that schema you protect your processing into that schema from every being doubled up.

By placing program CDWBM02 at the end of processing for any specific batch for a specific schema you will signal the completion of the batch for that schema.

8.3. Tables Required in the Target Database

The dw_batch_control table is required to be in the target schema for the batch maintenance utility to perform successfully. In the very first instance, to start processing you must enter an initial record in this table to signal that a batch has 'completed'. This table being empty is considered an 'error' condition because it will only ever be empty prior to any batch being submitted.

The batch_complete_flag is set to 0 when a batch is started and 1 when a batch is completed. Records are never deleted from this table. Thus you will build up a history of the batch processing over the months and years.

```
create table dwh4.dw_batch_control
(
    pk_batch_number          integer      not null
  , batch_date              datetime    not null
  , batch_complete_flag     integer     not null
  , constraint pk_dw_batch_control primary key
    ( pk_batch_number
    )
    on CDWOR01FG01
)
on CDWOR01FG01
;
```

8.4. Examples of Invoking the Batch Maintenance Utility

The Batch Maintenance Utility that is called at startup time is called CDWBM01. It can be invoked as described below. Note that you are expected to provide only 'out' connection information as the dw_batch_control table is expected to be in the target data warehouse.

```
CDWBM01.exe
DBConnectionOutParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=password;DATABASE=CDWOR01DB01
OutCatalogName=CDWOR01DB01
OutSchemaName=DWH4
OutTableName=dw_batch_control
ErrorMessageOutput=TargetDatabase
DebugLevel=9
Audit=Yes
```

The Batch Maintenance Utility that is called at completion time is called CDWBM02. It can be invoked as described below.

```
CDWBM02.exe
DBConnectionOutParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=password;DATABASE=CDWOR01DB01
OutCatalogName=CDWOR01DB01
OutSchemaName=DWH4
OutTableName=dw_batch_control
ErrorMessageOutput=TargetDatabase
DebugLevel=9
Audit=Yes
```

9. SQL STATEMENT PROCESSOR UTILITY

9.1. Why Have the SQL Statement Processor Utility?

Is there anything more annoying than having to cut/paste DDL from a file to a window to run a piece of DDL? Especially if you have to run lots of small pieces of DDL, which is often the case during development. I have always preferred to keep each piece of DDL in a separate file and then be able to run DDL as a simple command line interface. So after editing a piece of DDL I could just go to the command line and run it. Or, if I have changed 10 pieces of DDL I can make up a small command file and run all 10 quickly and simply.

All the databases give you some kind of utility to do this. However, they vary from database to database and I for one feel the error messages from these utilities when they are not working are very hard to use to fix whatever the problem is. So, once I had built the C++ Classes, one of which is able to execute any SQL statement that is supported by the database, it seemed like a great idea to write a small SQL Statement Processor. This way I will only ever have one command interface for all my DDL ever again. A file I can edit and a command I can run. Hooray!!!

9.2. What Does the SQL Statement Processor Utility Do?

The SQL Statement Processor reads a file and executes SQL Statements found within the file.

Note there are some limitations in the sql statement reader.

- Statements are separated by the semi-colon character (;).
- You may not have components of two commands on one line. When it sees a semi colon it believes this line is the end of the SQL Statement. If you have the start of the next statement on the same line it will also be passed to the statement processor, which will cause an error.
- No validity checking is performed on any statements. They are simply passed to the ODBC data source specified as a parameter and executed.
- If you place a semi colon near the end of the file and then enter blank lines after the semi colon it will interpret this as a blank statement and it will and execute the blank statement. It is recommended that the final semicolon in the file is on the last line.

9.3. Tables Required in the Target Database

No tables are required in the Target Database to run the SQL Statement Processor.

9.4. Examples of Invoking the SQL Statement Processor Utility

The SQL Statement is passed into the program using the SQLFileName parameter. Also, since the statement processor is executing statements against a database it was decided to use the 'out' parameters.

```
CDWU002.exe
DBConnectionOutParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=password;DATABASE=CDWOR01DB01
OutCatalogName=CDWOR01DB01
OutSchemaName=DWH4
OutTableName=dw_batch_control
SQLFileName=D:\CDWOR01\RELEASE\data\sqlfiletest1.dat
ErrorMessageOutput=TargetDatabase
DebugLevel=9
Audit=Yes
```

10. THE SQL SERVER 2000 DDL GENERATOR UTILITY

Note that the Oracle DDL Generator Utility is almost identical to the SQL Server 2000 DDL Generator Utility. The only point the Oracle DBA should be aware of is that the table and index filegroups in SQL Server 2000 translate into table/index tablespaces in Oracle.

10.1. Why Have The SQL Server 2000 DDL Generator Utility?

Is there anything more tedious than trying to maintain SQL DDL by hand? Perhaps only trying to maintain a lot of views of tables using one of the 'productivity tools' for database design. And who has never had the problem of getting the wrong number of columns in the create view statement when compared to the select statement supporting the view?

I recall on my very, very first data warehousing project back in 1991 the very first utility I wrote was a small product that allowed me to define all my columns only once and then to specify which columns where in which tables. It generated the DDL and the associated documentation from this little set of files. Saved me hours and hours of tedious ddl changes. Unfortunately it was written in rexx/2, a language that only ran on OS/2 at the time. And I have never bought the version that runs on windows.

I always wanted to be able to have that utility again on windows, and so now I have re-written it in C++.

10.2. What Does The SQL Server 2000 DDL Generator Utility Do?

The SQL Server 2000 DDL Generator Utility reads a simple model stored in two tables in the target database and from them generates the DDL defined by the data in the tables. It's not meant as a replacement for 'repository' products. It is meant as a quick and easy way of generating DDL for portions of the data warehouse. Especially making sure that the views line up etc.

10.3. Tables Required in the Target Database

The following tables must be in the target database:

- dw_ddl_gen_table
- dw_ddl_gen_columns

Dw_ddl_gen_table

The DDL for the dw_ddl_gen_table table is as follows:

```
create table dwh4.dw_ddl_gen_table
(
  process_row          varchar      (001)    not null default 'n'
, table_name           varchar      (255)    not null default 'unknown'
, view_name            varchar      (255)    not null default 'unknown'
, table_type_ind        varchar      (001)    not null default 'U'
, table_file_group      varchar      (030)    not null default 'unknown'
, index_file_group      varchar      (030)    not null default 'unknown'
, table_ddl_file        varchar      (255)    not null default 'unknown'
, view_ddl_file         varchar      (255)    not null default 'unknown'
)
on CDWOR01FG01
;
```

The dw_ddl_gen_table defines the tables/views that will be generated for this run of the DDL generator. Each record in dw_ddl_gen_table correlates to one table and one view being generated. The DDL generator always produces the table DDL and the view DDL and places the output into separate files so that you can inspect the DDL prior to running it against the database.

Field Name	Description
process_row	This has the values 'y' or 'n' depending on whether you want the DDL generator to generate the DDL for this table on this run of the DDL generator.
Table_name	This is the name of the table that will be generated by the DDL generator.
View_name	This is the name of the view that will be generated by the DDL generator.
Table_type_ind	This field defines the type of table that will be generated. I recommend you fill in the column as follows: 'd' – Dimension table 'f' – Fact table 'c' – Control table 's' – Staging table 'e' – Extract table 'w' – Work table Note that this field is used by the DDL generator to place the where level_col = 'detail' clause on the tables defined as 'd'imension tables.
Table_file_group	This is the SQL Server file group that you would like the physical table to be placed in. You can place the primary index and the data in different file groups if you wish. If you want to partition your fact tables across different file groups you will need to generate the DDL using your usual methods.
index_file_group	This is the SQL Server file group that you would like the primary index to be placed in. You can place the primary index and the data in different file groups if you wish.
Table_ddl_file	This is the fully qualified name of the file to which you would like to write the DDL for the table.
View_ddl_file	This is the fully qualified name of the file to which you would like to write the DDL for the view.

Some example values for this table are:

Proc Row?	Table	View Name	Type	Table File Group	Index File Group	Table DDL	View DDL
y	f_order_fact	f_order_fact_view	f	cdwor01fg01	cdwor01fg01	Filename	Filename
y	f_order_fact_summ	f_order_fact_summ_view	f	cdwor01fg01	cdwor01fg01	Filename	Filename

dw_ddl_gen_columns

```
create table dwh4.dw_ddl_gen_columns
( table_name          varchar      (030)    not null default 'unknown'
, table_column_name    varchar      (030)    not null default 'unknown'
, view_column_name     varchar      (030)    not null default 'unknown'
, db_data_type         varchar      (030)    not null default 'unknown'
, col_number_in_table  integer              not null default 0
, col_number_in_pk     integer              not null default 0
, flag_column_nullable varchar      (001)    not null default 'N'
)
on CDWOR01FG01
;
```

The dw_ddl_gen_columns table defines the columns within each of the tables. Each record in dw_ddl_gen_columns correlates to column in a table and view.

Field Name	Description
table_name	This is the same table name as specified in dw_ddl_gen_table. This record is looked up by the DDL generator after it has identified the table as one of the tables for which to generate DDL.
table_column_name	This is the name of the column in the table. It will be used in both the create table statement and the create view statement that is generated.
view_column_name	This is the name of the column in the view defined to be created in dw_ddl_gen_table. This name is only used in the generation of the DDL for the view.
db_data_type	This is the data type of the column that will be passed to the database in the DDL. It must be a valid data type in value DDL format for the database manager to process. For example, 'integer', varchar (10) and money are all valid data types for SQL Server.
col_number_in_table	This is the number of the column within the table. Strictly speaking it is not necessary in the relational environment. However, it is always useful to have columns ordered as you would like to see them. This column can just be an ascending number to place the columns in the table in the order that you would like them. The DDL generator sorts the columns into this order to produce the create table and create view DDL.
col_number_in_pk	This is the position of the column in the primary key. If a column is not a part of the primary key the column should be set to 0. Note that the DDL generator is expecting that every table has a primary key. If you want to use the DDL generator to generate the DDL for tables without primary keys you will have to edit the DDL after it is produced. The main reason for this is that it is a rare case indeed in a data warehouse that a table does not have a primary key. All dimension tables have the level column and dim char ky fld as primary keys and all summary fact tables have their combination of integer keys as primary keys. It is recommended that detail level fact tables have primary keys to provide a level of protection against double processing of a file into a fact table.
flag_column_nullable	This column is set to 'y' if the column is nullable and 'n' if the column is to be defined as not null.

Some example values for this table are:

Table Name	Table Column Name	View Column Name	DB Data Type	Col Number in Table	Col Number in PK	Null?
f_order_fact	pk_time_key	pk_time_key	integer	1	1	n
f_order_fact	pk_prod_key	pk_prod_key	integer	2	2	n
f_order_fact	pk_cust_key	pk_cust_key	integer	3	3	n
f_order_fact	pk_vendor_key	pk_vendor_key	integer	4	4	n
f_order_fact	invoice_qty	invoice_qty	integer	5	0	n
f_order_fact	invoice_amt	invoice_amt	money	6	0	n
f_order_fact	discount_amt	discount_amt	money	7	0	n
f_order_fact	product_code	product_code	varchar (6)	8	0	n
f_order_fact	customer_code	customer_code	varchar (6)	9	0	n
f_order_fact	vendor_code	vendor_code	varchar (6)	10	0	n
f_order_fact	invoice_num	invoice_num	varchar (6)	11	0	n
f_order_fact	invoice_remark	invoice_remark	varchar (30)	12	0	n
f_order_fact	order_date	order_date	datetime	13	0	n
f_order_fact	payment_date	payment_date	datetime	14	0	n
f_order_fact	delivery_date	delivery_date	datetime	15	0	n
f_order_fact_summ	pk_time_key	pk_time_key	integer	1	1	n
f_order_fact_summ	pk_prod_key	pk_prod_key	integer	2	2	n
f_order_fact_summ	pk_cust_key	pk_cust_key	integer	3	3	n
f_order_fact_summ	pk_vendor_key	pk_vendor_key	integer	4	4	n
f_order_fact_summ	invoice_qty	invoice_qty	integer	5	0	n
f_order_fact_summ	invoice_amt	invoice_amt	money	6	0	n
f_order_fact_summ	discount_amt	discount_amt	money	7	0	n

10.4. Example of Invoking The SQL Server 2000 DDL Generator Utility

Note that the SQL Server 2000 DDL Generator does not require an OutTableName because the names of the tables are set in the code.

```
CDWU003.exe
DBConnectionOutParameter=DSN=CDWOR01DB01;SERVER=PETERLAP\SQL200001;UID=dba;PWD=password;DATABASE=CDWOR01DB01
OutCatalogName=CDWOR01DB01
OutSchemaName=DWH4
ErrorMessageOutput=TargetDatabase
DebugLevel=9
Audit=Yes
```

11. ERROR MESSAGES

This section documents the error messages that can be issued by the Instant Data Warehouse.

Message Number	Message Text
CDWI0001	Program starting.
CDWI0002	Program submitted with the following parameters
CDWI0003	Program ending.
CDWI0004	The following SQL Statement was successfully executed against the data warehouse database:
CDWI0005	The number of rows affected by the preceding SQL statement is as follows:
CDWS0001	Program Terminating.
CDWS0002	Could not connect to DBConnectionInParameter. Value of DBConnectionInParameter is as follows:
CDWS0003	The CDWF001FileName input parameter was not set. CDWAG01 requires that this parameter is set because it is the input file to the program.
CDWS0004	There was no data found in the input table InTable. The select statement used is as follows:
CDWS0005	There was an open error for the input table. Refer to ODBC Messages in the error message log. The select statement used is as follows:
CDWS0006	The number of result columns from the select from InTable is equal to zero. The select statement used is as follows:
CDWS0007	Could not open WorkFileName for writing. Value of WorkFileName is as follows:
CDWS0008	Could not prepare WorkFileName for writing. Value of WorkFileName is as follows:
CDWS0009	Could not write to WorkFileName. Value of WorkFileName is as follows:
CDWS0010	Could not connect to DBConnectionOutParameter. Value of DBConnectionOutParameter is as follows:
CDWS0011	There was an open error opening the output table. Refer to ODBC Messages in the error message log. The select statement used is as follows:
CDWS0012	The number of result columns from the validation of OutTable is equal to zero. The select statement used is as follows:
CDWS0013	The preparation for the insert statement for OutTable failed. The fully qualified name of the OutTable is as follows:
CDWS0014	The preparation for the update statement for OutTable failed. The fully qualified name of the OutTable is as follows:
CDWS0015	The preparation for the delete statement for OutTable failed. The fully qualified name of the OutTable is as follows:
CDWS0016	Could not open WorkFileName for reading. Value of WorkFileName is as follows:
CDWS0017	Could not prepare WorkFileName for reading. Value of WorkFileName is as follows:
CDWS0018	Could not read from WorkFileName. Value of WorkFileName is as follows:
CDWS0019	Error encountered when transferring data from WorkFileName to Insert Database pointer to perform insert. Contact technical support. The fully qualified table name for the insert table is as follows:
CDWS0020	The insert encountered a clash of primary keys or unique index and the program parameter specifies 'insert only' for rows. If you want key/index clashes to be resolved automatically change the InsertUpdateOption. The fully qualified table name for the insert table is as follows:
CDWS0021	Error encountered when transferring data from Insert database pointer to the Update database pointer to perform update. Contact technical support. The fully qualified table name for the update table is as follows:
CDWS0022	Error encountered when performing the update after a failed insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the update table is as follows:
CDWS0023	Error encountered when performing the delete after a failed insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the delete table is as follows:
CDWS0024	Error encountered when performing the insert after a failed insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CDWS0025	Error encountered when transferring data from Insert database pointer to the Delete database

	pointer to perform delete. Contact technical support. The fully qualified table name for the delete table is as follows:
CDWS0026	Error encountered when transferring data from WorkFileName to Update database pointer to perform update. Contact technical support. The fully qualified table name for the update table is as follows:
CDWS0027	Error encountered when transferring data from Update database pointer to the Insert database pointer to perform insert. Contact technical support. The fully qualified table name for the insert table is as follows:
CDWS0028	Error encountered when performing the insert after a failed update. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CDWS0029	Error encountered when performing update. Refer to ODBC Messages in the error message log. The fully qualified table name for the update table is as follows:
CDWS0030	The parameter defining the WorkFileName is not set. The WorkFileName must be set to a file that the program can open for reading and writing.
CDWS0031	Error encountered when performing the validate of the aggregation control table. Refer to ODBC Messages in the error message log. The select statement used to validate the aggregation control table is as follows:
CDWS0032	Error encountered when performing the open of the aggregation control table. Refer to ODBC Messages in the error message log. The select statement used to open the aggregation control table is as follows:
CDWS0033	An unexpected end of data was encountered when reading the aggregation control table. There must be at least one record in the aggregation control table specifying a summary level for the fact table specified in the input parameters. The select statement used to read rows from the aggregation control table is as follows:
CDWS0034	Could not open CDWF001FileName for reading. Value of CDWF001FileName is as follows:
CDWS0035	Could not prepare CDWF001FileName for reading. Value of CDWF001FileName is as follows:
CDWS0036	Memory allocation error. The program was unable to allocate memory to hold data in memory.
CDWS0037	Could not open CDWF002FileName for writing. Value of CDWF002FileName is as follows:
CDWS0038	Could not prepare CDWF002FileName for writing. Value of CDWF002FileName is as follows:
CDWS0039	Function f2010_initialise_for_aggregate_control_record_processing failed. Review Error Log for more messages.
CDWS0040	Function f2020_process_CDWF001_producing_sort_work_1 failed. Review Error Log for more messages.
CDWS0041	Function f2030_process_sort_work_1_producing_sort_work_2 failed. Review Error Log for more messages.
CDWS0042	Function f2040_process_sort_work_2_producing_CDWF002 failed. Review Error Log for more messages.
CDWS0043	Error encountered when performing the delete of all records in the sort work 1 table. Refer to ODBC Messages in the error message log. The sql executed to perform the delete is as follows:
CDWS0044	Error encountered when performing the delete of all records in the sort work 2 table. Refer to ODBC Messages in the error message log. The sql executed to perform the delete is as follows:
CDWS0045	The preparation for the insert statement for the sort work 1 table failed. Refer to ODBC Messages in the error message log. The sort work 1 table name is as follows:
CDWS0046	The file CDWF001FileName is empty. Program CDWAG01 should not be executed if there has been no data input to CDWAT01. Value of CDWF001FileName is as follows:
CDWS0047	Function f2090_send_data_to_sort_work_1 failed. Review Error Log for more messages.
CDWS0049	Error encountered when performing the insert into the sort work 1 table. Refer to ODBC Messages in the error message log. The sort work 1 table name is as follows:
CDWS0050	Error encountered when performing the insert select sort sum group by statement to summarise data from sort work table 1 and place it into sort work table 2. Refer to ODBC Messages in the error message log. The insert select statement executed is as follows:
CDWS0051	The close of the prepared insert statement for the sort work 1 table failed. Refer to ODBC Messages in the error message log. The sort work 1 table name is as follows:
CDWS0052	Error encountered when performing the open of the sort work 2 table. Refer to ODBC Messages in the error message log. The select statement used to open the sort work 2 table is as follows:
CDWS0053	Error encountered when performing the read of the sort work 2 table. Refer to ODBC Messages in the error message log. The select statement used to read the sort work 2 table is as follows:
CDWS0054	The CDWF001FileName input parameter was not set. CDWAT01 requires that this parameter is set because it is the output file from the program.
CDWS0055	Error encountered when performing the validate of the input table. Refer to ODBC Messages in the error message log. The select statement used to validate the input table is as follows:

CDWS0056	Error encountered when performing the open of the input table. Refer to ODBC Messages in the error message log. The select statement used to open the input table is as follows:
CDWS0057	No input data for CDWAT01 to process. This is defined to be an error. Please do not submit the fact table attribution process to run if there is no input data available
CDWS0058	Error encountered when performing the validate of a dimension table. Refer to ODBC Messages in the error message log. The select statement used to validate the dimension table is as follows:
CDWS0059	Could not open CDWF001FileName for writing. Value of CDWF001FileName is as follows:
CDWS0060	Could not prepare CDWF001FileName for writing. Value of CDWF001FileName is as follows:
CDWS0061	Function f3100_attribute_record failed. Review Error Log for more messages.
CDWS0062	Error encountered when writing CDWF001FileName. Value of CDWF001FileName is as follows:
CDWS0063	The CDWF002FileName input parameter was not set. CDWAG01 requires that this parameter is set because it is the output file from the program.
CDWS0064	The CDWF002FileName input parameter was not set. CDWCL01 requires that this parameter is set because it is the input file to the program.
CDWS0065	The CDWF003FileName input parameter was not set. CDWCL01 requires that this parameter is set because it is an output file from the program.
CDWS0066	The CDWF004FileName input parameter was not set. CDWCL01 requires that this parameter is set because it is an output file from the program.
CDWS0067	Error encountered when performing the validate of the summary fact table. Refer to ODBC Messages in the error message log. The select statement used to validate the summary fact table is as follows:
CDWS0068	Could not open CDWF002FileName for reading. Value of CDWF002FileName is as follows:
CDWS0069	Could not prepare CDWF002FileName for reading. Value of CDWF002FileName is as follows:
CDWS0070	The file CDWF002FileName is empty. Program CDWCL01 should not be executed if there has been no data input to CDWAT01. Value of CDWF002FileName is as follows:
CDWS0071	The preparation for the insert statement for the summary fact table failed. CDWAG01 prepares an insert statement to discover the characteristics of the summary fact table. The fully qualified name of the summary fact table is as follows:
CDWS0072	The preparation for the insert statement for the summary fact table failed. CDWCL01 prepares an insert statement to discover the characteristics of the summary fact table. The fully qualified name of the summary fact table is as follows:
CDWS0073	The preparation for the select statement for the summary fact table failed. Refer to ODBC Messages in the error message log. CDWCL01 prepares a select statement to read the summary fact table. The fully qualified name of the summary fact table is as follows:
CDWS0074	Could not open CDWF003FileName for writing. Value of CDWF003FileName is as follows:
CDWS0075	Could not prepare CDWF003FileName for writing. Value of CDWF003FileName is as follows:
CDWS0076	Could not open CDWF004FileName for writing. Value of CDWF004FileName is as follows:
CDWS0077	Could not prepare CDWF004FileName for writing. Value of CDWF004FileName is as follows:
CDWS0078	Error encountered when writing CDWF003FileName. Value of CDWF003FileName is as follows:
CDWS0079	Error encountered when writing CDWF004FileName. Value of CDWF004FileName is as follows:
CDWS0080	Error encountered when performing the open of the input table for the dimension table data. Refer to ODBC Messages in the error message log. The select statement used to open the input table is as follows:
CDWS0081	The number of columns returned by the select statement that opened the input table for the dimension table data returned zero columns. The select statement used to open the input table is as follows:
CDWS0082	Function f3090_update_aggregate_keys_control failed. Review Error Log for more messages.
CDWS0083	Error encountered when performing the validate of the input table for dimension table data. Refer to ODBC Messages in the error message log. The select statement used to validate the input table is as follows:
CDWS0084	Error encountered when performing the validate of the base type 2 dimension table. Refer to ODBC Messages in the error message log. The select statement used to validate the base dimension table is as follows:
CDWS0085	Error encountered when performing the validate of the output dimension table. Refer to ODBC Messages in the error message log. The select statement used to validate the output dimension table is as follows:
CDWS0086	Error encountered when performing the validate of the dimension table key definitions table. Refer to ODBC Messages in the error message log. The select statement used to validate the dimension table key definitions table is as follows:
CDWS0087	Error encountered when performing the validate of the dimension table type 2 column definitions table. Refer to ODBC Messages in the error message log. The select statement used to validate

	the dimension table type 2 column definitions table is as follows:
CDWS0088	Error encountered when performing the validate of the aggregate keys control table. Refer to ODBC Messages in the error message log. The select statement used to validate the aggregate keys control table is as follows:
CDWS0089	Error encountered when performing the open of the aggregate keys control table. Refer to ODBC Messages in the error message log. The select statement used to open the aggregate keys control table is as follows:
CDWS0090	Error encountered when performing the validate of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch control table is as follows:
CDWS0091	Error encountered when performing the validate of the dw month control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw month control table is as follows:
CDWS0092	Error encountered when performing the open of the dimension table key definitions table. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table key definitions table is as follows:
CDWS0093	The level of aggregate in 'agg_level' field in the dimension table key definitions table exceeds the maximum number of levels of aggregates in the Instant Data Warehouse. Please change the 'agg_level' field on the dimension table key definitions table.
CDWS0094	Error encountered when performing the open of the dimension table type 2 column definitions table. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table type 2 column definitions table is as follows:
CDWS0095	The preparation for the select statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CDWS0096	Error encountered when performing the open of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch control table table is as follows:
CDWS0097	Error encountered when performing the open of the dw month control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw month control table table is as follows:
CDWS0098	The preparation for the insert statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CDWS0099	The preparation for the first update statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CDWS0100	The preparation for the second statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CDWS0101	Function f3030_get_integer_keys failed. Review Error Log for more messages.
CDWS0102	Function f3060_update_rows failed. Review Error Log for more messages.
CDWS0103	Function f3050_insert_rows failed. Review Error Log for more messages.
CDWS0104	Function f3061_close_dimension_record failed. Review Error Log for more messages.
CDWS0105	Function f3061_close_dimension_record failed. Review Error Log for more messages.
CDWS0106	Function f3062_update_dimension_record failed. Review Error Log for more messages.
CDWS0107	Error encountered when performing the prepared select on the output dimension table to look up the level keys. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CDWS0108	Error encountered when performing the prepared insert on the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CDWS0109	Error encountered when performing the prepared update on the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CDWS0110	Error encountered when performing the prepared update to close out the row in the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CDWS0111	Error encountered when performing the prepared update on the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:

CDWS0112	The preparation for the update statement for the aggregate keys control table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the aggregate keys control table is as follows:
CDWS0113	Error encountered when performing the prepared update on the aggregate keys control table. Refer to ODBC Messages in the error message log. The fully qualified table name of the aggregate keys control table is as follows:
CDWS0114	Error encountered when performing the insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CDWS0115	An attempt has been made to execute the program after the Evaluation Period has expired. Contact peter@peterolan.net to discuss obtaining a new evaluation copy of the Instant Data Warehouse.
CDWS0116	Error encountered when performing the validate of the dim table load control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dim table load control table is as follows:
CDWS0117	Function f1000_initialise_part2 failed. Review Error Log for more messages.
CDWS0118	The dimension table load to memory function could not open the dimension table. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table is as follows:
CDWS0119	The dimension table load to memory function determined that there are zero rows in the dimension table. Zero rows in a dimension table is not allowed. The select statement used to open the dimension table is as follows:
CDWS0120	The dimension table load to memory function determined that there are less than 2 columns returned by a select * statement against a dimension table. This is not allowed. The view of the dimension table must return at least the dimension table character key field and one integer key. The select statement used to open the dimension table is as follows:
CDWS0121	The dimension table load to memory function has been unable to allocate memory for the pointer to store the pointers to the dimension character keys(ptr_ptr_dim_char_ky_fld). More memory is needed, or turn off loading into memory for the dimension table. The select statement used to open the dimension table is as follows:
CDWS0122	The dimension table load to memory function has been unable to allocate memory to store the integer keys(ptr_ws_dim_int_keys). More memory is needed, or turn off loading into memory for the dimension table. The select statement used to open the dimension table is as follows:
CDWS0123	The dimension table load to memory function has been unable to allocate memory store a dimension character key(ptr_dim_char_ky_fld). More memory is needed, or turn off loading into memory for the dimension table. The select statement used to open the dimension table is as follows:
CDWS0124	The dimension table load to memory function encountered an unexecpted error. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table is as follows:
CDWS0125	Function PrepareDimensionSelect() failed. Review Error Log for more messages.
CDWS0126	Function PerformPreparedDimensionSelect() failed. Review Error Log for more messages.
CDWS0127	Error encountered when performing the validate of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch control table is as follows:
CDWS0128	Error encountered when performing the open of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch control table is as follows:
CDWS0129	The dw batch control indicates that there is an incomplete batch. You may not start a new batch until the previous batch is completed. The select statement used to check the dw batch control table is as follows:
CDWS0130	Error encountered when performing the insert of the new dw batch control table record. Refer to ODBC Messages in the error message log. The insert statement used to insert the new dw batch control record is as follows:
CDWS0131	Error encountered when performing the update to close out the dw batch control table record. Refer to ODBC Messages in the error message log. The update statement used to close out the dw batch control record is as follows:
CDWS0132	Error encountered when performing an SQL Statement in CDWU002. Refer to ODBC Messages in the error message log. The SQL statement issued is as follows:
CDWS0133	The SQLFileName input parameter was not set. CDWU001 requires that this parameter is set because it is the input file to the program."
CDWS0134	Could not open SQLFileName for reading. Value of SQLFileName is as follows:

CDWS0135	Error encountered when performing the validate of the dw ddl gen table table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw ddl gen table is as follows:
CDWS0136	Error encountered when performing the validate of the dw ddl gen columns table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw ddl gen columns table is as follows:
CDWS0137	Error encountered when performing the open of the dw ddl gen table table. Refer to ODBC Messages in the error message log. The select statement used to open the dw ddl gen table is as follows:
CDWS0138	Function f1000_initialise_part1 failed. Review Error Log for more messages.
CDWS0139	Error encountered when performing the open of the dw ddl gen columns table. Refer to ODBC Messages in the error message log. The select statement used to open the dw ddl gen columns table is as follows:
CDWS0140	Could not open Output File for Table DDL for writing. Value of Output File for Table DDL is as follows:"
CDWS0141	Could not open Output File for View DDL for writing. Value of Output File for View DDL is as follows:
CDWS9001	An ODBC Error Occurred. Details Following.

11.1. Notes on Error Messages

The Instant Data Warehouse performs many checks during processing to determine if there are any error conditions that have occurred. All error conditions detected are reported via the error message classes. When running any program you can specify where the error messages are written to:

1. The C error file which is the console.
2. A file accessible by the server. All messages are always appended to the file if the file already exists.
3. To the error messages table in the data warehouse.

It is recommended that all error messages are written to the error message table in the data warehouse in a production environment because this is the easiest place for support staff to find error messages.

However, during initial setup and testing one of the errors that you may have is the inability to connect to the data warehouse database. In which case it will not be possible to write an error message to the database!!

The ODBC database access classes do not issue any error messages themselves. Any error conditions raised by ODBC are sent back to the calling program so that the calling program can add information to the error condition (such as function in which the error was found) prior to writing it to the error output specified.

This introduces a complication for those functions in the ODBC classes that emulate database functions by loading the dimension tables into memory and then searching the in memory arrays. There are a variety of problems that can arise during the execution of this code, the most notable of which is running out of memory.

In any case where the ODBC database access classes detect an error which is not an ODBC error the class issues an error message that is 'reported as' an ODBC error. You can identify these messages because the SQL State and SQL Native Error values are set to 9999, which you will not find in the ODBC error message manual. Currently these are messages CDWS0116 to CDWS0124. However, more messages which are reported as ODBC error are likely to be added in the future.