

May 2009

Welcome to the May 2009 edition of IBM InfoSphere Information Server Developer's Notebook. This month we answer the question;

How can I use Information Server with my dimensional data model to my data warehouse or data mart?

Excellent question! On initial reaction, we almost want to say; How can't you use Information Server with your data mart, Information Server aiding in so many areas to a quick and successful data mart design, creation, delivery, and maintenance. While Information Server solved only the ETL role 8 or more years ago, today it performs many more functions including data discovery, data cleansing and base line analysis, data delivery, and much, much more.

Because so many topics present themselves with this broader question, we have to pick just one area to give focus to, and that topic is; populating a dimensional data model with regards to a slowly changing dimension.

Software versions

All of these solutions were *developed and tested* on (IBM) InfoSphere Information Server (IIS) version 8.1.1, using the Microsoft Windows XP/SP3 platform to support IIS client programs, and a RedHat Linux Advanced Server 5 (RHAS 5) 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server side components.

IBM InfoSphere Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM InfoSphere Information Server product contains the following major components;

WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, InfoSphere Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM InfoSphere Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

28.1 Terms and core concepts

As stated above, there are dozens of topics one can discuss in the area of IBM InfoSphere Information Server (IIS) regarding data mart and data warehouse design, creation, data cleansing, delivery and maintenance. In this edition of IBM InfoSphere Information Server Developer's Notebook (IISDN), we give focus to just one of these topics; that being, slowly changing dimensions. Slowly changing dimensions (SCD) is an information technology industry standard term, SCD is not a proprietary IIS topic.

Slowly changing dimensions, a primer

Generally we will say, there are relational modeled data sources (relational databases) and non-relational modeled data sources (hierarchical databases, XML data, other). Within just relational modeled data sources, there are two further quantifications of data modeling; E/R modeling (also known as third normal form modeling), and then dimensional modeling.

Generally, E/R modeled systems are associated with write intensive, read via indexed key, on line transaction processing (OLTP) or on line operational data store (ODS) style systems. One entity, for example Customer, begets one or more Order records, begets one or more Order Line Items. The demonstration database from the IBM/Informix Dynamic Server (IDS), modeled in E/R, third normal form, appears in Figure 28-1.

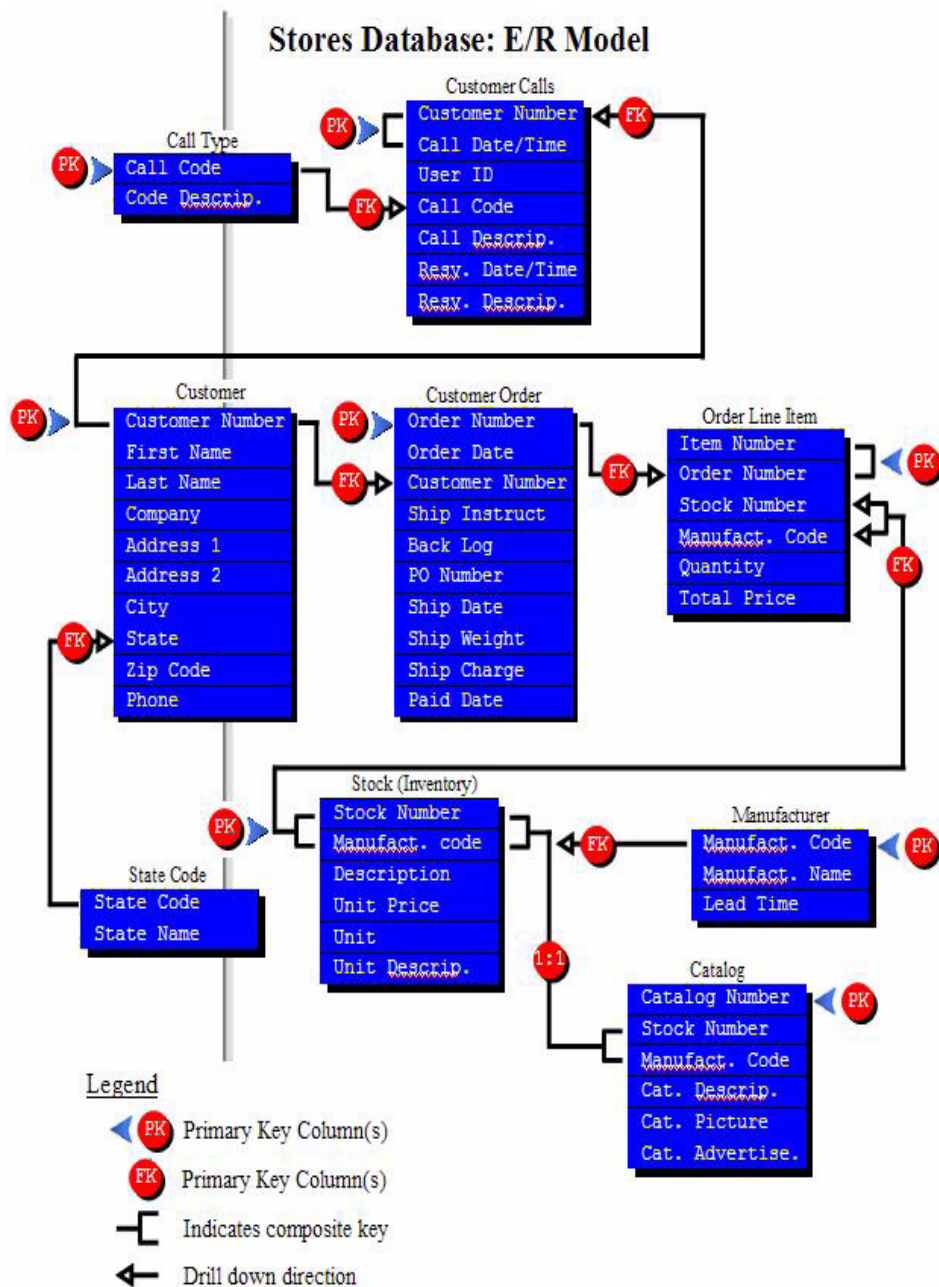


Figure 28-1 Stores database in E/R, OLTP style data model.

Generally, the Informix Stores database is designed to record new customer sales orders, optimized for on line transaction processing; write intensive, read via known (and indexed) pathways, few previously unanticipated data access routines.

Figure 28-2 displays the same basic database, modeled, however, in a dimensional form.

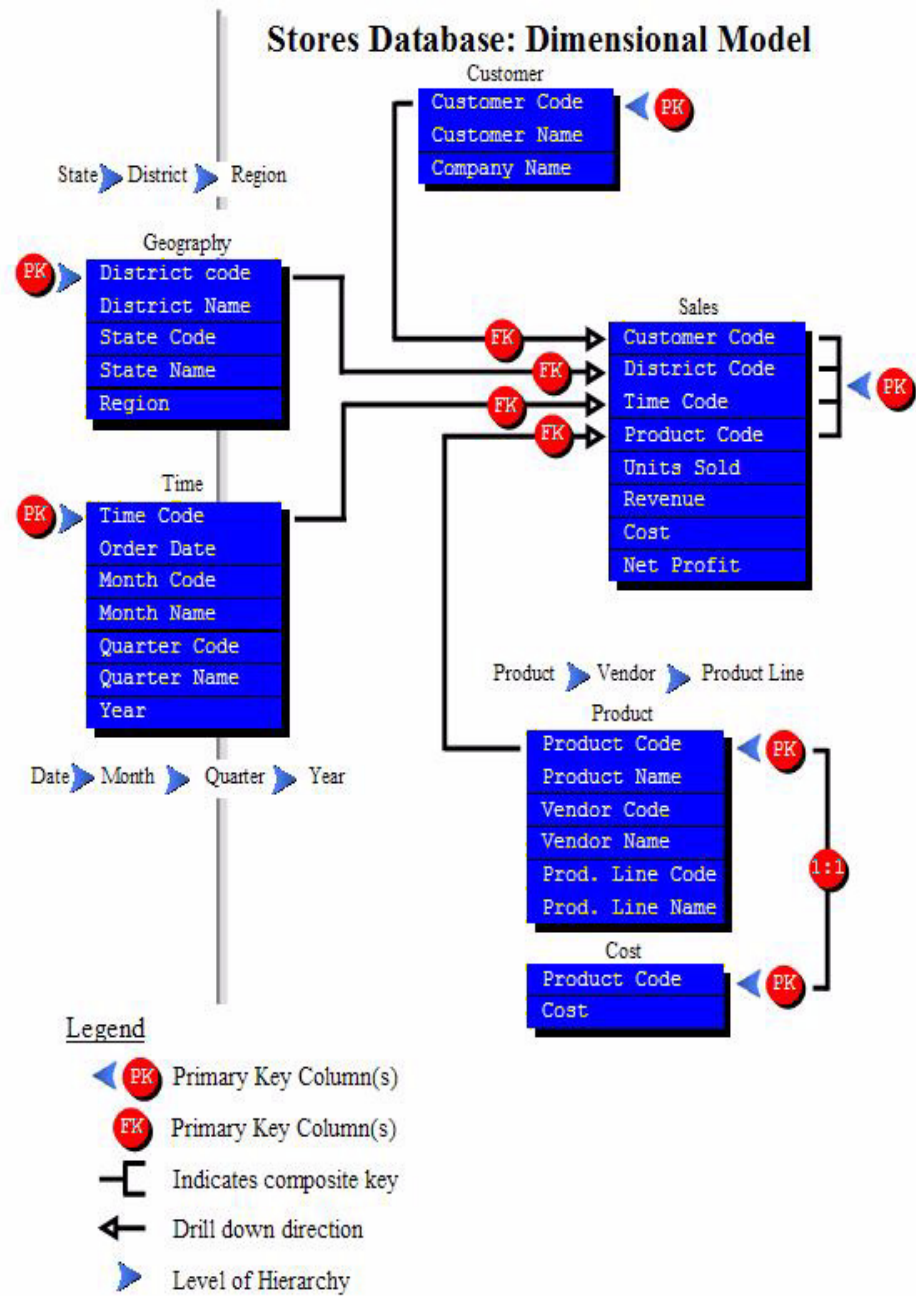


Figure 28-2 The same basic Stores database, modeled for dimensional use.

Dimensional data models support sweeping, entirely and previously unanticipated read pathways. Also, dimensional models support temporal data analysis, an area where OLTP style systems fall terribly short.

Note: An OLTP style system can, for example, tell you how many of a given stock number are on the shelf currently, or the current price for a given item.

An OLTP style system has difficulty, for example, telling you how many of a given stock number were on the shelf every day at 2 PM for the last 120 days. OLTP style systems lack in the area of history, giving preference to current states and conditions.

Dimensional models optimize for reading, and can easily tell you states and conditions over unit time.

Dimensionally modeled systems are centered on a single or very small number of Fact tables, which store the key/central information; for example, the event of a given sale, of a single item, to a given customer. Dimension tables surround the Fact table, and store repetitive information used for roll ups, or summation (trending) of data. Where the Fact table stores the sales information, the Dimension table stores, for example, the geographic data related to the customer; what City they are in, rolling up to a State, rolling up to a District, a Region, Country, and so on.

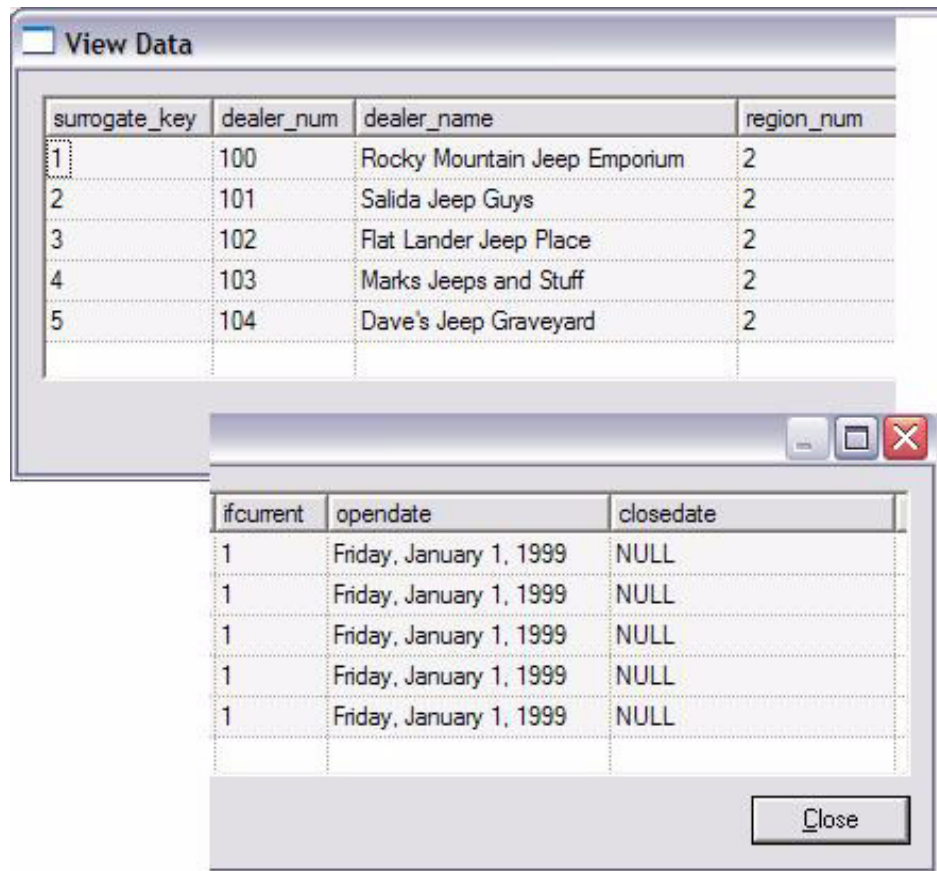
There are specific adjectives used to describe columns in a dimensional model. Columns in a Fact table may be non-additive, additive, or semi-additive. (Other adjectives are also present, but not discussed here.) While you can count occurrences of the state abbreviation, CO (for Colorado, U.S.), you can't add one CO abbreviation to another. (State abbreviation in this context is non-additive). Generally you can add columns related to revenue, as a total revenue, sub-total revenue, etcetera. (Revenue in this context is additive.) An example of a numeric column you can add, but only along certain indices (thus, semi-additive), might be Quantity-On-Hand. It makes sense to calculate the total number of Stereos you have on hand currently, or at one location, but why would you add them over unit time, that result yields a non-sequitur response (it doesn't make sense). If you had 50 stereos on hand in January, the same 50 in February, and the same 50 still in March, did you have 150 stereos on hand in the first quarter, or just 50? If you have 50 stereos on hand in Denver, and 50 in Boulder, do you have 100 in Colorado?

Columns in a Dimension Table have adjectives too. And the adjectives we are specifically discuss in this section of this document are, Type-1, Type-2, and Type-3 dimension table columns.

In Figure 28-2, we might say that the Company_Name column, in the Customer Dimension table, is a Type-1 column. If the Company Name changes, we update the Company Name column and are done with it. For example, at a time in history, the National House of Pancakes changed its name to the International House of Pancakes (IHop). Its still the same company, and we wish to report on it as such.

An example of a Type-2 column in a Dimension Table in Figure 28-2 might be, Product.Product_Line, with a given Product_Code belonging to a given Product_Line. We might have grouped Products like Cell_Phone_X, iPod_Y, Camera_Z, etcetera, into a Product Line called Personal Electronics. This year, however, we want to track sales of Cell_Phone_X in a new Product Line called, Personal Electronic Navigation Products, so that we may specifically see how we are performing in the area of GPS enabled personal electronics. We want the old association to exist for whatever reason, but also show the new relationship of a given product to a new product grouping. These are the unique challenges of reporting, and dimensionally modeled systems.

The data model we use for in this edition of IISDN tracks automobile sales. The single Dimension table for this model is displayed in Figure 28-3. (This table is wide enough that we split its display across two images.)



surrogate_key	dealer_num	dealer_name	region_num
1	100	Rocky Mountain Jeep Emporium	2
2	101	Salida Jeep Guys	2
3	102	Flat Lander Jeep Place	2
4	103	Marks Jeeps and Stuff	2
5	104	Dave's Jeep Graveyard	2

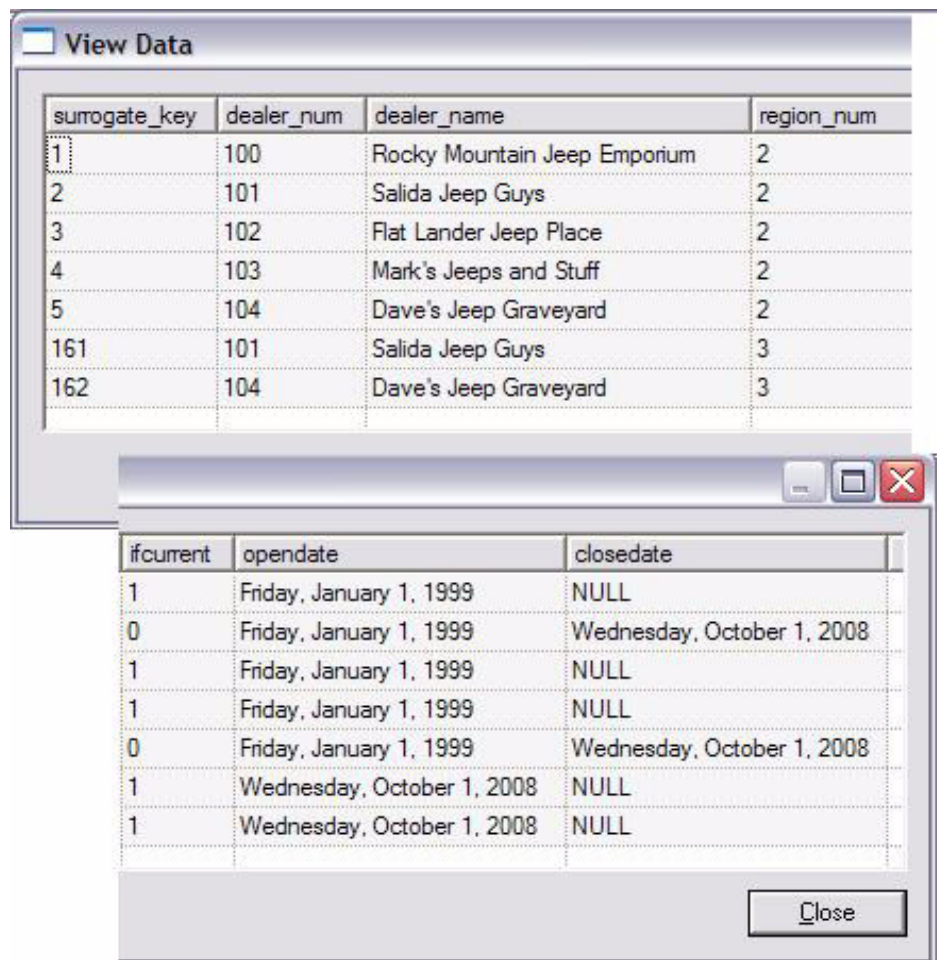
ifcurrent	opendate	closedate
1	Friday, January 1, 1999	NULL
1	Friday, January 1, 1999	NULL
1	Friday, January 1, 1999	NULL
1	Friday, January 1, 1999	NULL
1	Friday, January 1, 1999	NULL

Close

Figure 28-3 Dealer Dimension table.

In Figure 28-3, we have 5 Dealers, each belonging to Region_Num 2, all records are Current (ifCurrent = 1), and all records were made effective on Jan 01, 1999 (displayed as OpenDate). None of the records are listed as closed, ineffective.

In Figure 28-4, we have the same 5 Dealers, some have changed Regions, some have changed Dealer Name.



surrogate_key	dealer_num	dealer_name	region_num
1	100	Rocky Mountain Jeep Emporium	2
2	101	Salida Jeep Guys	2
3	102	Flat Lander Jeep Place	2
4	103	Mark's Jeeps and Stuff	2
5	104	Dave's Jeep Graveyard	2
161	101	Salida Jeep Guys	3
162	104	Dave's Jeep Graveyard	3

ifcurrent	opendate	closedate
1	Friday, January 1, 1999	NULL
0	Friday, January 1, 1999	Wednesday, October 1, 2008
1	Friday, January 1, 1999	NULL
1	Friday, January 1, 1999	NULL
0	Friday, January 1, 1999	Wednesday, October 1, 2008
1	Wednesday, October 1, 2008	NULL
1	Wednesday, October 1, 2008	NULL

Close

Figure 28-4 Dealer Dimension table, with Type-1 and Type-2 changes.

In Figure 28-4, we see that the Dealer_Name of the original Dealer_Num 5, has changed from "Marks Jeeps and Stuff", to "Mark's Jeeps and Stuff". Compare Figure 28-3 to Figure 28-4, and look for the new apostrophe. Dealer_Name is, in this case, a Type-1 dimension column.

In Figure 28-4, Dealer_Num(s) 101 and 104 changed Region_Num, which is a Type-2 dimension column. In this case, we create a new record for each to reflect the new Dealer_Num, Region_Num assignment.

Type-2 dimension columns use two keys for their identity. In this case, we can join by Dealer_Num to display the same dealer, albeit with two past, and exclusive by time, Region_Num assignments. Type-2 dimension columns generally have a second identifying key, called a surrogate key, which is truly unique, and is thus guaranteed to allow us to join with a distinct Dealer_Num, in this case, over version, over unit time.

Type-2 dimension columns generally include an effective date, both start and end, as displayed in Figure 28-4 by the columns OpenDate and CloseDate.

Note: Whereas Type-1 dimension columns require no change to the expected dimension table column structure (the updated column value is made in place, updating an existing column), Type-2 dimension columns are normally associated with extra indicator columns; a surrogate key column (we need to effectively version the previously existing primary key column value), and effective begin/end date indicator columns, and even an 'If current' type column.

A change to a Type-2 dimension column creates a new record in the database table, that with the newly updated Type-2 dimension column values; hence, the need for a surrogate key. If you had two versions of the "CO/Colorado" record where CO is the expected primary key into that table, How would that work? You can't have two primary keys with the same value- The surrogate key provides uniqueness, where the CO value column becomes known as the 'business key'.

Type-3 dimension columns do not create a new/second record upon update of their dimension column value. Type-3 dimension columns often store a current value column and prior version column in the same row, in the same table.

If you wish to read more on the topic of slowly changing dimensions, we recommend,

Wikipedia.com always has good summary articles. See,

http://en.wikipedia.org/wiki/Slowly_changing_dimension

Also, Brian Caufield of IBM wrote a really good IBM Developer Work's article on slowly changing dimensions, and then integration and use with IBM InfoSphere Information Server. See,

<http://www.ibm.com/developerworks/data/tutorials/dm-0903datastageslowlychanging/>

Lastly, a really good and free book on the topic of dimensional modeling, written by Daniel M. Farrell, Et.al., can be viewed and/or downloaded at,

<http://www.redbooks.ibm.com/abstracts/sg247138.html?Open>

Keeping track of slowing changing dimension, data

Slowly changing dimensions are an information technology industry standard concept, used commonly in data marts and data warehouses, and are not directly related to IBM InfoSphere Information Server. If you had to maintain slowly changing dimensions yourself, manually, that would entail a lot of work; you have lookups, dimension column value comparisons, new surrogate key generation, beginning and ending date calculation and more.

Fortunately, IBM InfoSphere Information Server has a single, high function Stage to do all of this work for you. This being just one of the many examples of the differentiating functionality of Information Server over similar products; the benefit being higher end user productivity.

Figure 28-5 displays a sample IBM Information Server DataStage component Job that maintains a slowly changing dimension. A code review follows this image.

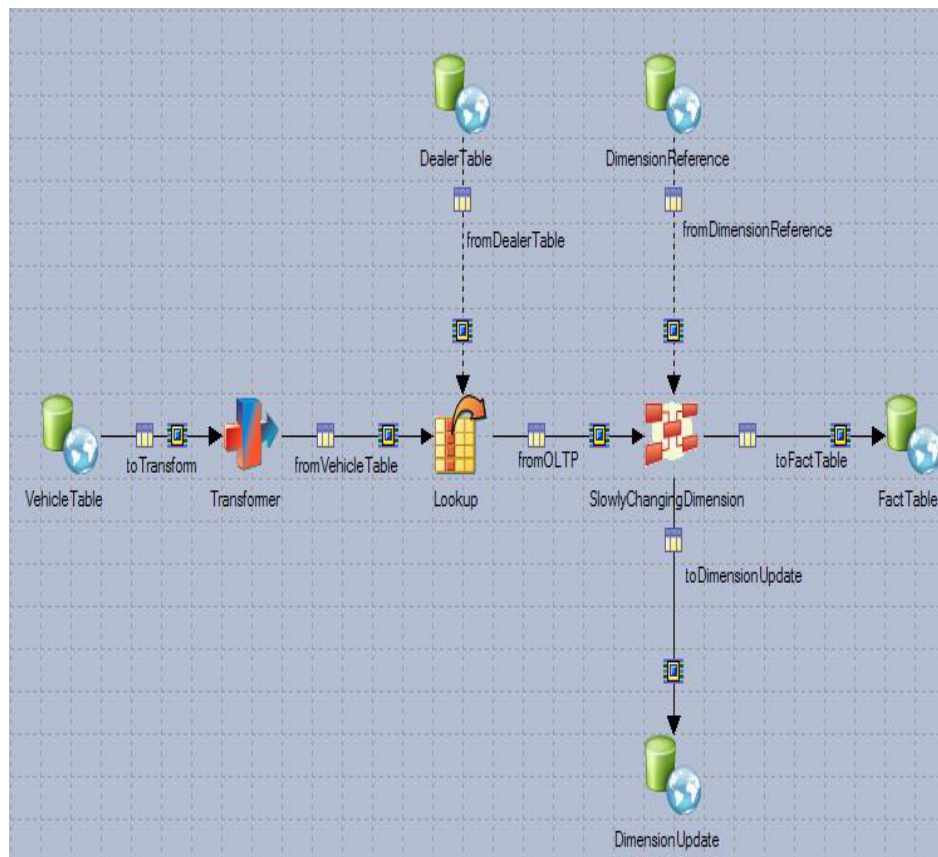


Figure 28-5 Sample DataStage Job with Slowly Changing Dimensions.

A code review related to Figure 28-5 is offered below;

- The DataStage Stage specifically meant to serve the maintenance of a slowly changing dimension is displayed in Figure 28-5 with the Stage name, SlowlyChangingDimension.
This Stage is located in the Processing drawer of the Palette view, and requires 4 links, 2 input, 2 output, making the minimum number (count) of Stages for a DataStage Job using the Slowly Changing Dimension Stage to be 5, (1 + 2 + 2).
- Two Stages are configured to access the Dimension table, one for input (read), and one for output; generally a write in the form of a SQL UPSERT.
The Dimension table read Stage is called DimensionReference, and its Link is called the Dimension Reference Link. The Dimension table write

Stage is called DimensionUpdate, and its Link is called the Dimension Update Link.

- One Link goes to the Fact table, and is also generally written in the form of a SQL UPSERT.
- Two additional Stages present themselves in Figure 28-5 that are not required, but are at least common. They are;

- Transformer

A Type-2 Dimension column has an associated effective date. While the Slowly Changing Dimension Stage allows us to hard code an effective date, or use a DataStage Job Parameter to receive this value, we use the Transformer to calculate an effective date value row by row.

Note: Another thing we like to do with Transformers is data type casting.

In the DataStage Job displayed in Figure 28-5, we are reading and writing from several database tables. In each of those database related Stages, we read or write the data in the exact data type or format as expected by the associated SQL table and column.

Some database servers perform implicit data type conversion. Huh? You can SQL INSERT a CHAR data type into an INTEGER column, and all will be well, assuming that the actual data value can convert without error. IBM Informix is such a database server.

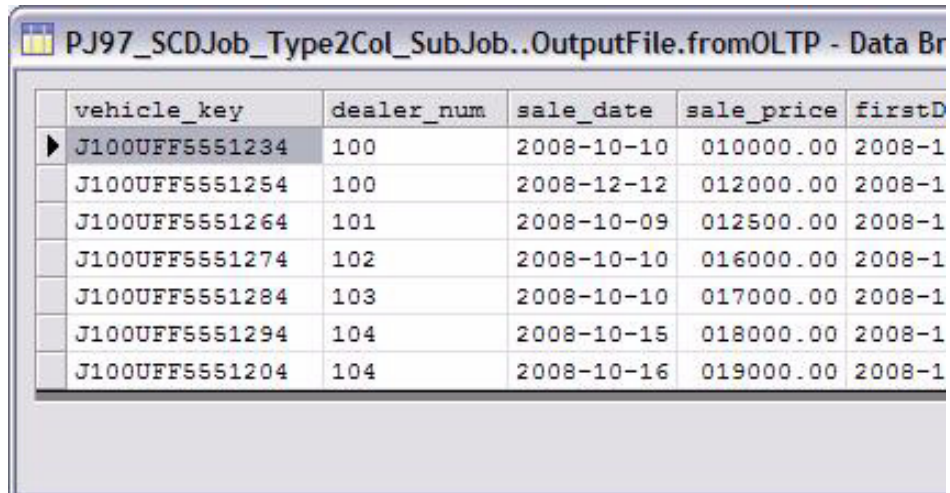
Other database servers do not support implicit data type conversion. Oracle and DB2/UDB come to mind.

The net here becomes; we read and write data from SQL tables without conversion in the reading or writing Stage. We use a Transform Stage as an intermediary to convert data types and related properties, for example, format and length.

- Lookup-

As is the most often use case, an Slowly Changing Dimension application receives line item data (Fact Table data), and needs to enrich it from demographic data (Dimension Table data) first, before entering the Slowly Changing data comparison.

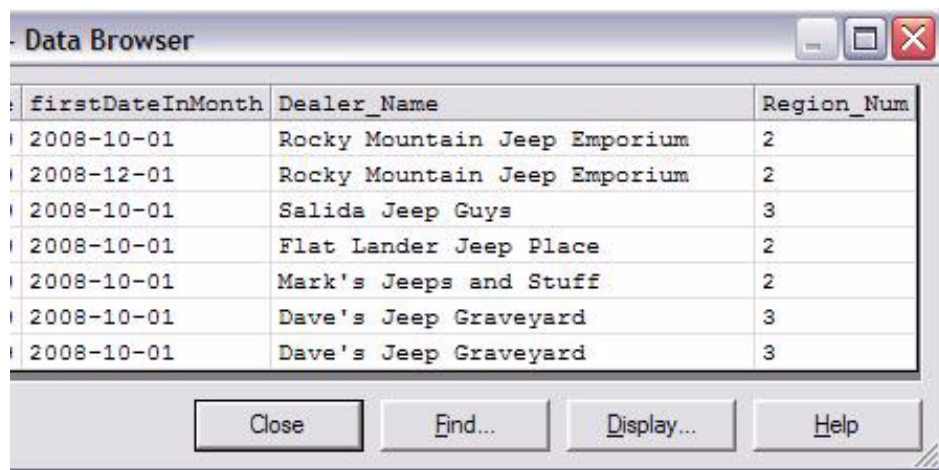
In the example instructions below, we give focus to the Slowly Changing Dimension Stage, and then any downstream/dependent Stages. We begin the instructions from the sample data set displayed in Figure 28-6 and Figure 28-7.



The screenshot shows a window titled "PJ97_SCDJob_Type2Col_SubJob..OutputFile.fromOLTP - Data Br". It contains a table with the following data:

vehicle_key	dealer_num	sale_date	sale_price	firstD
J100UFF5551234	100	2008-10-10	010000.00	2008-1
J100UFF5551254	100	2008-12-12	012000.00	2008-1
J100UFF5551264	101	2008-10-09	012500.00	2008-1
J100UFF5551274	102	2008-10-10	016000.00	2008-1
J100UFF5551284	103	2008-10-10	017000.00	2008-1
J100UFF5551294	104	2008-10-15	018000.00	2008-1
J100UFF5551204	104	2008-10-16	019000.00	2008-1

Figure 28-6 Sample input data set, page 1 of 2.



The screenshot shows a window titled "Data Browser". It contains a table with the following data:

firstDateInMonth	Dealer_Name	Region_Num
2008-10-01	Rocky Mountain Jeep Emporium	2
2008-12-01	Rocky Mountain Jeep Emporium	2
2008-10-01	Salida Jeep Guys	3
2008-10-01	Flat Lander Jeep Place	2
2008-10-01	Mark's Jeeps and Stuff	2
2008-10-01	Dave's Jeep Graveyard	3
2008-10-01	Dave's Jeep Graveyard	3

At the bottom of the window are buttons for "Close", "Find...", "Display...", and "Help".

Figure 28-7 Sample input data set, page 2 of 2.

The DataStage Job we used to make the sample data is displayed in Figure 28-8, and is just a subset, the preface to the DataStage Job displayed in Figure 28-5.

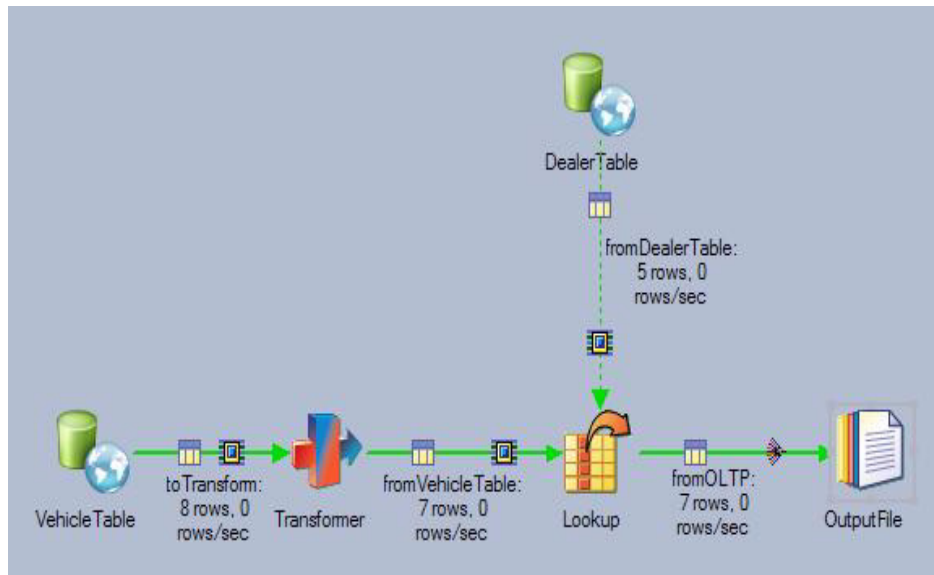


Figure 28-8 Sample Job to make input data set data, do not make.

The actual (minimal) DataStage Job we make in this document, and the Job for which instructions are supplied, is displayed in Figure 28-9.

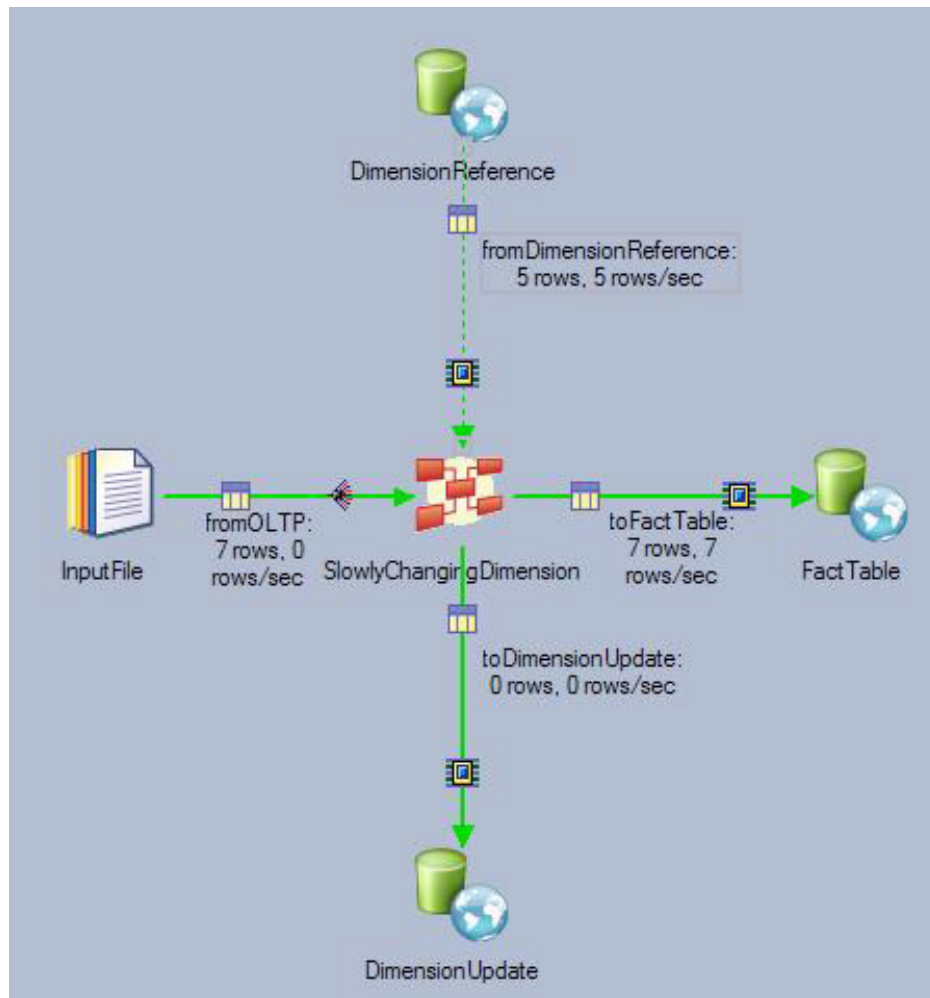


Figure 28-9 The DataStage Job we make in this document.

The full SQL command file to make the required tables and data is displayed below in Example 28-1.

Example 28-1 SQL DDL/DML to make sample tables and data.

```
CREATE DATABASE ids_iisdn WITH LOG;
GRANT dba TO public;

CREATE TABLE OLTPSide.Dealer_Table
(
  Dealer_Num    INTEGER NOT NULL PRIMARY KEY,
```



```
--
Dealer_Name    CHAR(32),
Region_Num     INTEGER
) LOCK MODE ROW;

INSERT INTO Dealer_Table Values(100,
    "Rocky Mountain Jeep Emporium",2);
INSERT INTO Dealer_Table Values(101,
    "Salida Jeep Guys      ",3);
INSERT INTO Dealer_Table Values(102,
    "Flat Lander Jeep Place  ",2);
INSERT INTO Dealer_Table Values(103,
    "Mark's Jeeps and Stuff  ",2);
INSERT INTO Dealer_Table Values(104,
    "Dave's Jeep Graveyard   ",3);

CREATE TABLE OLTPSide.Vehicle_Table
(
    Vehicle_Key    CHAR(32) NOT NULL PRIMARY KEY,
    --
    Dealer_Num     INTEGER,
    Sale_Date      DATE,
    Sale_Price     DECIMAL(8,2)
) LOCK MODE ROW;

INSERT INTO Vehicle_Table Values(
    "J100UFF5551234",100,"10/10/2008",10000.60);
INSERT INTO Vehicle_Table Values("
    J100UFF5551244",100,NULL      ,NULL  );
INSERT INTO Vehicle_Table Values(
    "J100UFF5551254",100,"12/12/2008",12000.60);
INSERT INTO Vehicle_Table Values(
    "J100UFF5551264",101,"10/09/2008",12500.60);
INSERT INTO Vehicle_Table Values(
    "J100UFF5551274",102,"10/10/2008",16000.60);
INSERT INTO Vehicle_Table Values(
    "J100UFF5551284",103,"10/10/2008",17000.60);
INSERT INTO Vehicle_Table Values(
    "J100UFF5551294",104,"10/15/2008",18000.60);
INSERT INTO Vehicle_Table Values(
    "J100UFF5551204",104,"10/16/2008",19000.60);

CREATE TABLE DataMart.DealerDimension_Table
(
    Surrogate_Key  INTEGER NOT NULL PRIMARY KEY,
    --
    Dealer_Num     INTEGER,
    Dealer_Name    CHAR(32),
    Region_Num     INTEGER,
```

```
--
ifCurrent      INTEGER,
openDate       DATE,
closeDate      DATE
) LOCK MODE ROW;

CREATE TABLE DataMart.DealerDimension_Table2
(
  Dealer_Num     INTEGER NOT NULL PRIMARY KEY,
  Dealer_Name    CHAR(32),
  CurrentRegion_Num INTEGER,
  --
  effectiveDate  DATE,
  lastRegionNum  INTEGER
) LOCK MODE ROW;

CREATE TABLE DataMart.SalesFact_Table
(
  Vehicle_Key    CHAR(32) NOT NULL PRIMARY KEY,
  --
  DealerDimension_SK INTEGER,
  --
  Dealer_Num     INTEGER,
  Sale_Date      DATE,
  Sale_Price     DECIMAL(8,2)
) LOCK MODE ROW;

CREATE TABLE DataMart.SalesFact_Table2
(
  Vehicle_Key    CHAR(32) NOT NULL PRIMARY KEY,
  --
  Dealer_Num     INTEGER,
  Sale_Date      DATE,
  Sale_Price     DECIMAL(8,2)
) LOCK MODE ROW;
```

The SQL command file to reset between runs of your new Slowly changing Dimensions DataStage Job is displayed below in Example 28-2.

Example 28-2 SQL DDL/DML to reset data between tests.

```
DELETE FROM DealerDimension_Table
WHERE 1 = 1;

INSERT INTO DealerDimension_Table Values(
  1,100,"Rocky Mountain Jeep Emporium",
  2,1,"01/01/1999",NULL);
```

```
INSERT INTO DealerDimension_Table Values(
  2,101,"Salida Jeep Guys      ",
  2,1,"01/01/1999",NULL);
INSERT INTO DealerDimension_Table Values(
  3,102,"Flat Lander Jeep Place ",
  2,1,"01/01/1999",NULL);
INSERT INTO DealerDimension_Table Values(
  4,103,"Marks Jeeps and Stuff  ",
  2,1,"01/01/1999",NULL);
INSERT INTO DealerDimension_Table Values(
  5,104,"Dave's Jeep Graveyard  ",
  2,1,"01/01/1999",NULL);
```

```
DELETE FROM DealerDimension_Table2
WHERE 1 = 1;
```

```
INSERT INTO DealerDimension_Table2
SELECT
  Dealer_Num      ,
  Dealer_Name     ,
  Region_Num      ,
  openDate        ,
  "
FROM
  DealerDimension_Table;
```

```
DELETE FROM SalesFact_Table
WHERE 1 = 1;
```

```
DELETE FROM SalesFact_Table2
WHERE 1 = 1;
```

28.2 Complete the following examples

In this section of this document, we create one new IBM Information Server (IIS) DataStage component Job that demonstrates use of the Slowly changing Dimension Stage. The following pre-requisite conditions must be met in order to follow these instructions;

- You need an ASCII text sample data file with content and structure equal to that as displayed in Figure 28-6 and Figure 28-7.
- You need to configure your IIS DataStage Project to access a SQL data source.

Instructions for configuring an IIS DataStage Project are found in a prior edition of this document; namely, the December/2007 edition, Configuring SQL Data Sources.

- You need to create a series of SQL tables and seed with data equal to that as displayed in Example 28-1.
1. Create a new IBM Information Server (IIS) DataStage component Parallel Job.
 - Save the Job with a given name and in a given folder.
 - Drag and Drop 5 Stages from the Palette View to the Parallel Canvas.
 - From the File Drawer of the Palette View, 1 Sequential File Stage.
 - From the Processing Drawer, 1 Slowly Changing Dimension Stage.
 - From the Database Drawer, 3 ODBC Connector Stages. Figure 28-9.

Note: Links have an associated Link Type, and correct operation of the Slowly Changing Dimension Stage is highly dependent on Links being of the correct type.

The Link Type *defaults* to the type of Stage it originates from or terminates at, and the order in which it was created (the order in which it was attached to a given Stage).

There are 4 Link Types going to or coming from the Slowly Changing Dimension (SCD) Stage, and having a ready grasp of these Link Types (names), will aid in your configuring and understanding of the SCD Stage.

- Link the 5 Stages you dragged and dropped above; 5 Stages equals 4 Links.

Linking is best done with a Right-Click, Drag and Release, from the source to target Stage. *If you make your Links in the given order stated below, your Links will need less changes;*

- Sequential File Stage (InputFile) to Slowly Changing Dimension (SCD) Stage.
- ODBC Connector Stage (DimensionReference) to SCD Stage.

This Link forms a Reference Link, like we commonly use for any Lookup Stage.
- SCD Stage to ODBC Connector Stage (FactTable).

This Link is called our 'Output Link', and is a critically different Link Type than the next Link we create which also outputs.

- SCD Stage to ODBC Connector Stage (DimensionUpdate).

This Link forms our Dimension Update Link, a Link Type that is specific to the SCD Stage.

- Reposition and rename all Stages and Links as displayed in Figure 28-9.

Note: If you've ever configured the Sequential File and ODBC Connector Stages before, then the only new, and thus hard, Stage left to configure is the Slowly Changing Dimension (SCD) Stage.

We will configure each of the supporting Stages first, then finish with configuring SCD Stage.

2. Configure and test the Sequential File Stage entitled, InputFile.

This Stage is done when you can read the sample data as displayed in Figure 28-6 and Figure 28-7.

3. Configure the ODBC Connector Stage entitled, DimensionReference.

Initially the SQL table sourced by this Stage is empty. This Stage is done when you can Click the View Data Button and receive no error.

As a source Stage, supporting a Reference Link, this Stage is configured to perform a SQL SELECT against the DealerDimension_Table.

4. Configure the ODBC Connector Stage entitled, DimensionUpdate.

Initially the SQL table targeted by this Stage is empty. This Stage is done when you can Click the View Data Button and receive no error.

As a target Stage, supporting a Dimension Update Link, this Stage is normally configured to perform a SQL UPSERT; an INSERT then UPDATE, or UPDATE then INSERT, whichever is most likely to occur.

There are 2 relevant TABS in the configuration of this Stage, and both are displayed below as Figure 28-10 and Figure 28-11.

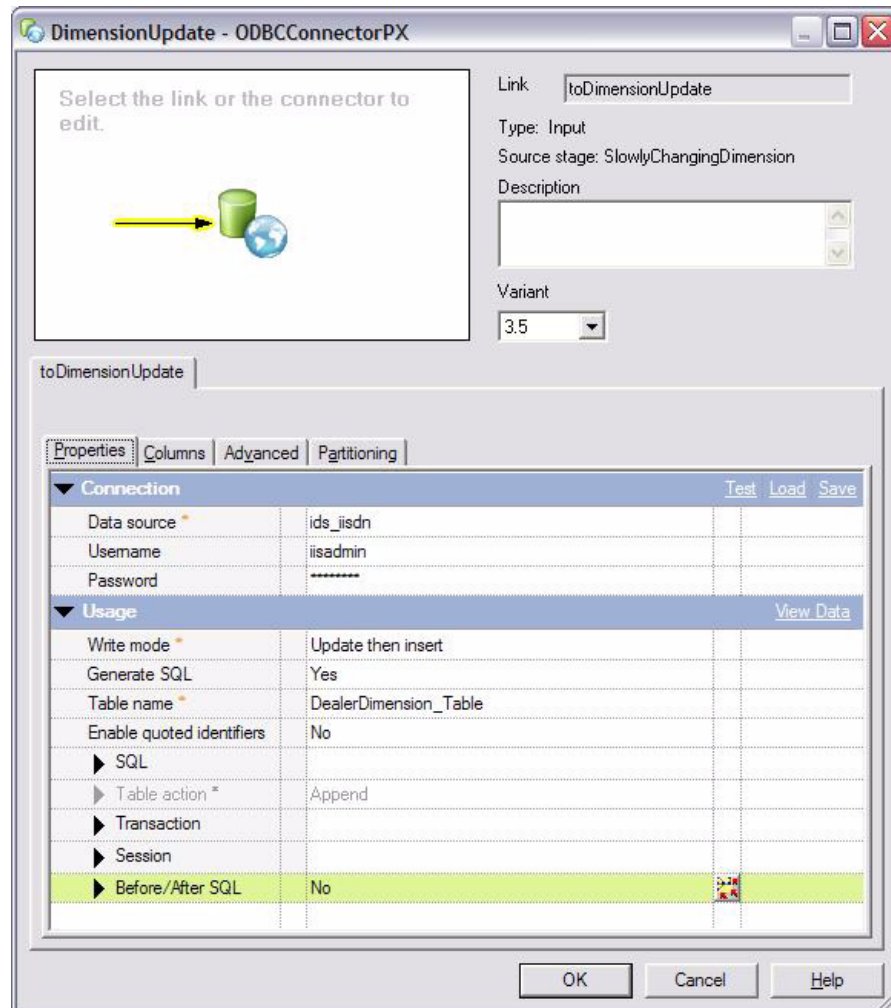


Figure 28-10 Properties TAB of ODBC Connector Stage, DimensionUpdate.

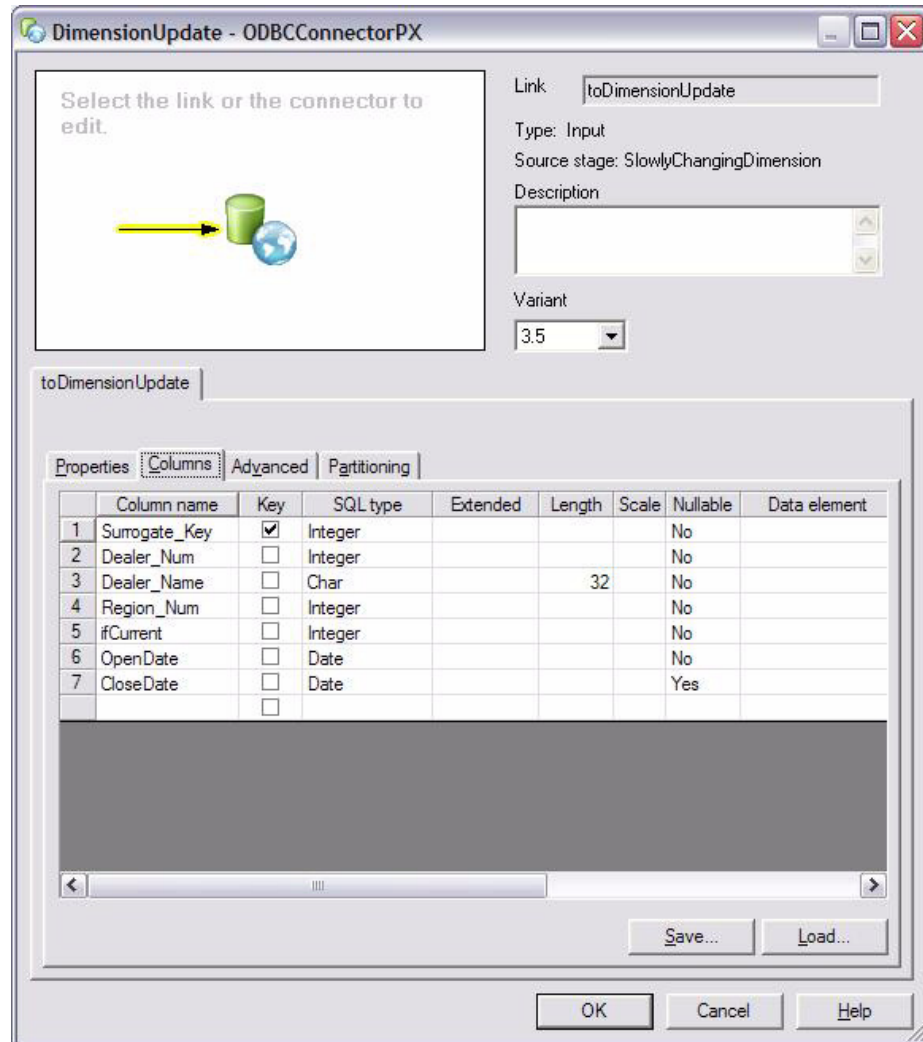


Figure 28-11 Columns TAB of ODBC Connector Stage, DimensionUpdate.

5. Configure the ODBC Connector Stage entitled, FactTable.

Initially the SQL table targeted by this Stage is empty. This Stage is done when you can Click the View Data Button and receive no error.

As a target Stage, supporting a Dimension Update Link, this Stage is normally configured to perform a SQL UPSERT; an INSERT then UPDATE, or UPDATE then INSERT, whichever is most likely to occur.

Again, we offer 2 figures, Figure 28-12 and Figure 28-13.

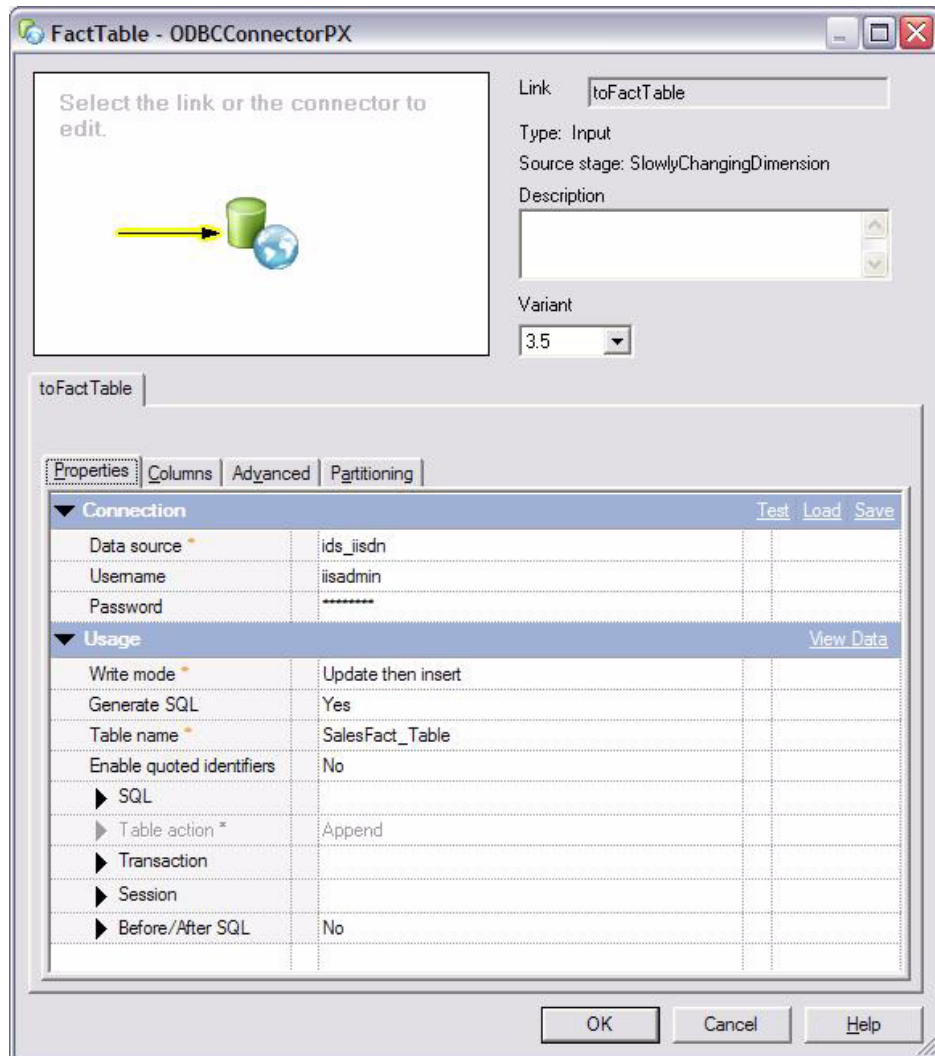


Figure 28-12 Properties TAB of ODBC Connector Stage, FactTable.

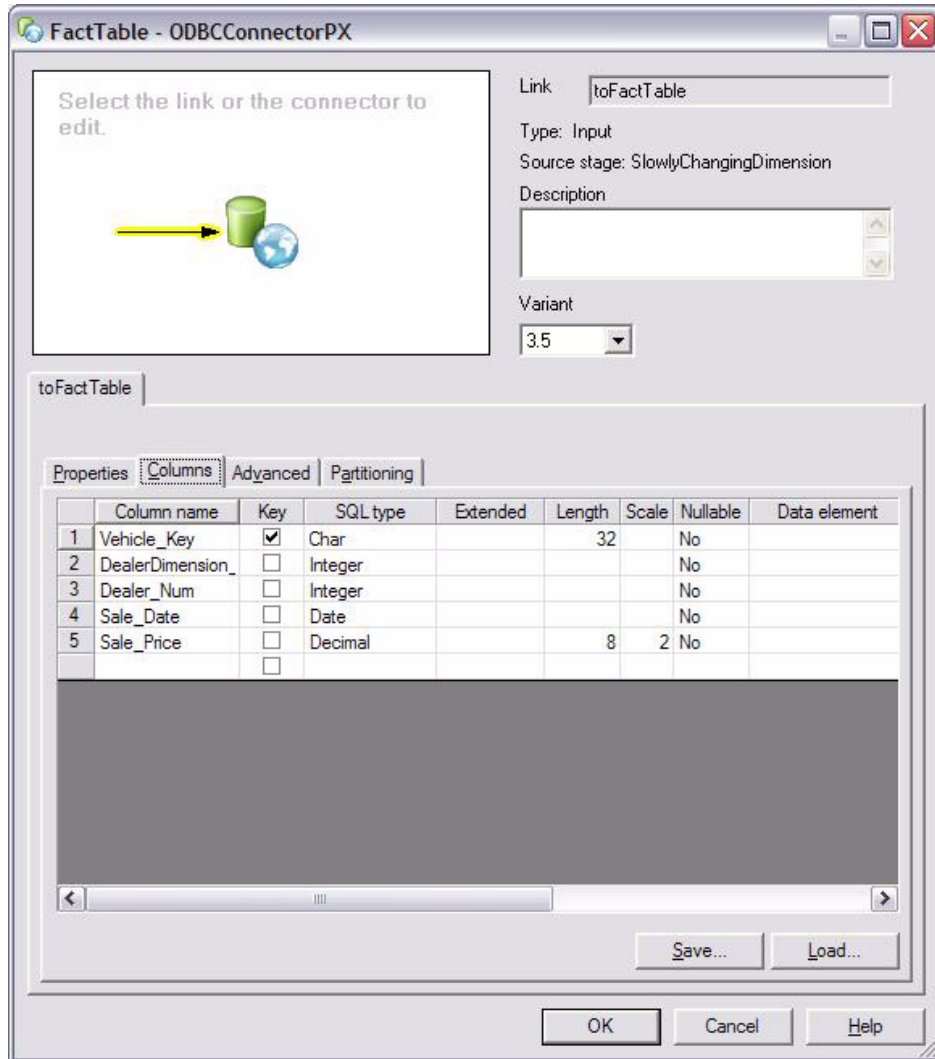


Figure 28-13 Columns TAB of ODBC Connector Stage, FactTable.

6. Configure the Slowly Changing Dimension (SCD) Stage.

the only remaining task then, is to configure the SCD Stage, save, compile and test.

- a. On the Stage -> General TAB, ensure that the Drop Down List Box entitled, 'Select output link', is set to 'toFactTable'.

This should already be set correctly, as the result of the order in which you created your Links above.

If this control is not set correctly, change it, and be extra cautious in your settings below. You are done with this step when your settings equal those as displayed in Figure 28-14.

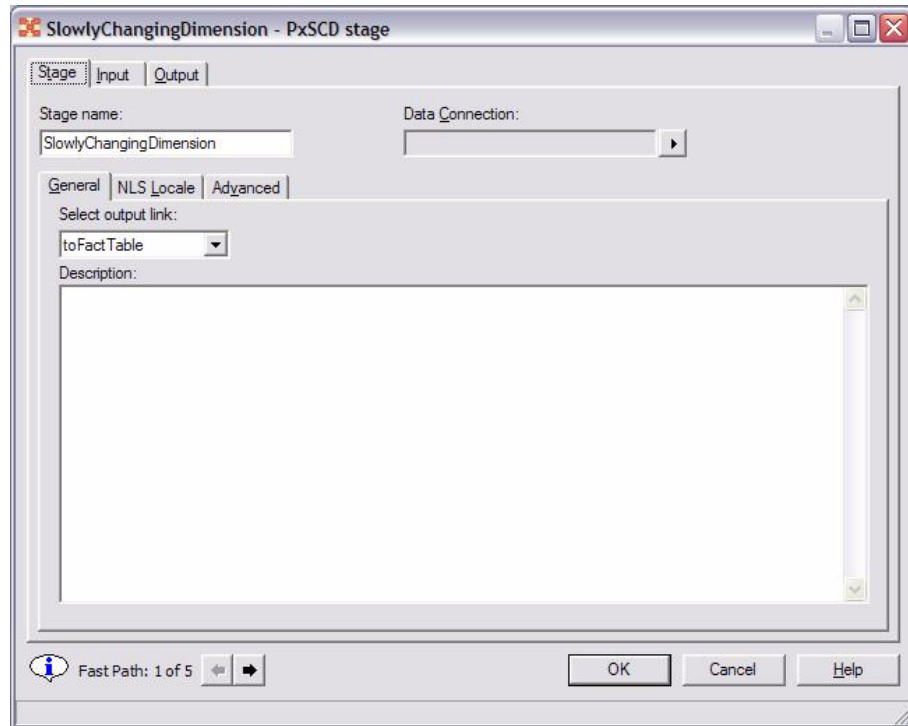


Figure 28-14 Stage -> General TAB to the SCD Stage.

- b. On the Input -> Lookup TAB, manage the settings to equal those as displayed in Figure 28-15.
 - The Drop Down List Box entitled, InputName, should already equal fromDimensionReference.
 - The left side of the paneled display should already contain 7 input columns.

The right side of the paneled display is affected via Drop Down List Box and other controls.
 - You are done with this step when your settings equal those as displayed in Figure 28-15.

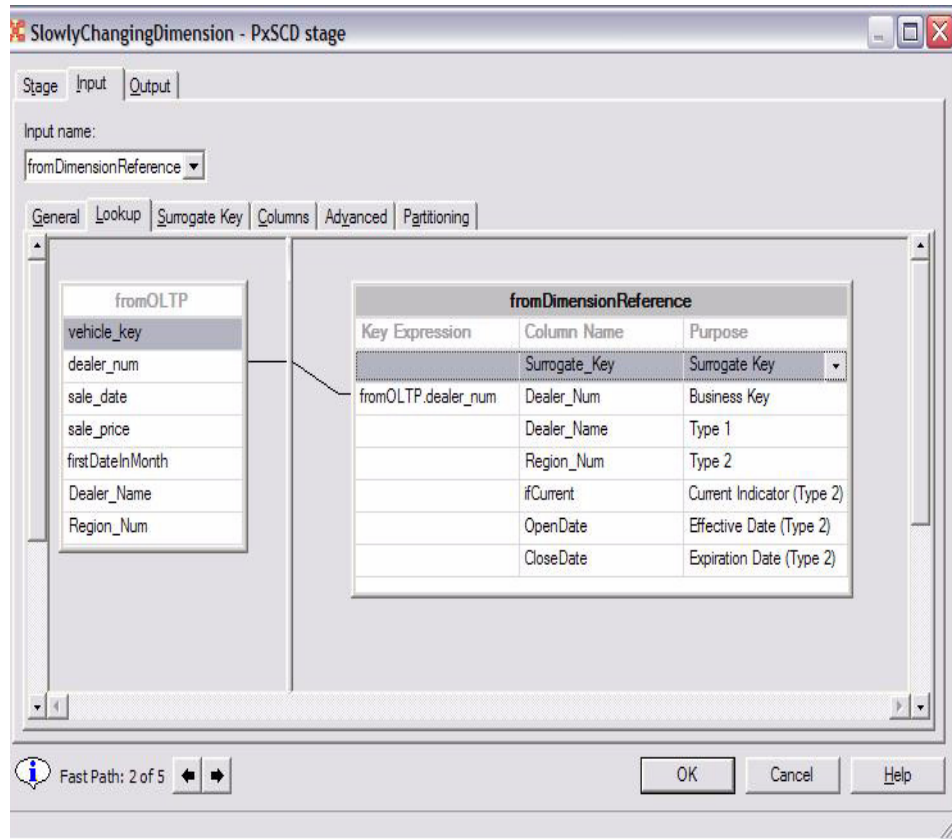


Figure 28-15 Input -> Lookup TAB to the SCD Stage.

- c. On the Input -> Surrogate Key TAB, manage the settings to equal those as displayed in Figure 28-16.

Above we discussed the concept of a surrogate key column, and how it related to the ideas of a slowly change dimension, specifically to a Type-2 Dimension column.

This TAB is where we specify the device used to generate a Surrogate Key Value. For simplicity (and speed), we use a binary data file on disk that IBM Information Server will manage to track the next highest key value to assign.

This file must exist before this DataStage Job will run successfully.

Note: The next edition of this document details how to make use of database resident object to generate a surrogate key value.

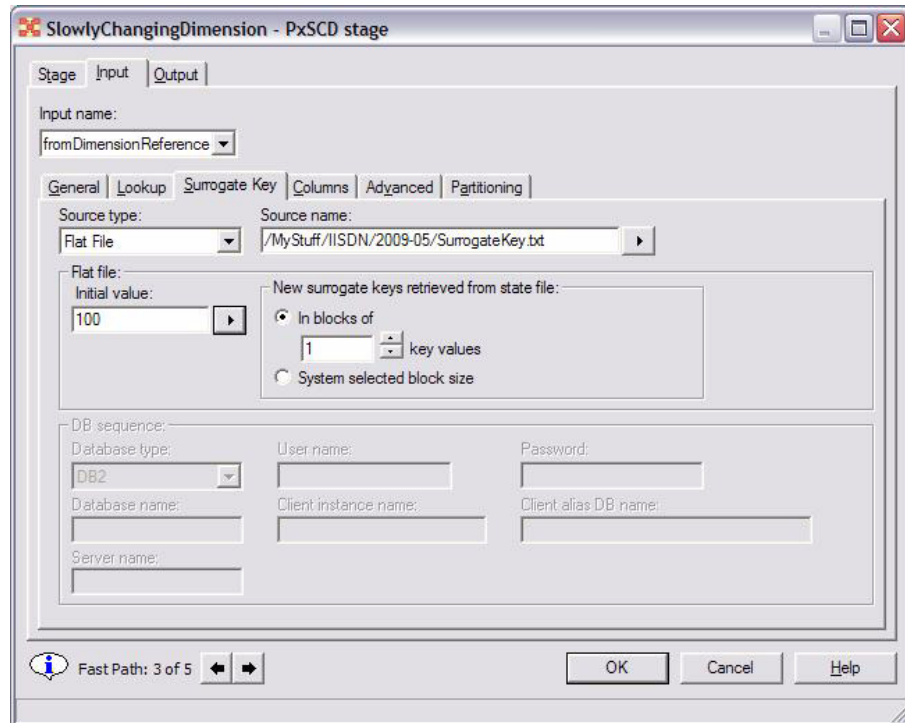


Figure 28-16 Input -> Surrogate Key TAB to the SCD Stage.

- d. The Input -> Columns TAB (not pictured) has two input Links, as controlled by the Drop Down List Box entitled, 'Input name'.

Both Input Links should display a single key column. For the Input Link entitled, fromOLTP, this should be the vehicle_key column. And for the Input Link entitled, fromDimensionReference, it should be Dealer_Num.

If these values are not correct, they are fixed on the source Stage feeding this SCD Stage.

- e. The Output -> Dim Update TAB is set next.
 - i. Again there are 2 Links to edit, from the Drop Down List Box entitled, 'Output name'.

- ii. From the Output Link entitled, toDimensionUpdate, manage the settings to equal those as displayed in Figure 28-17.

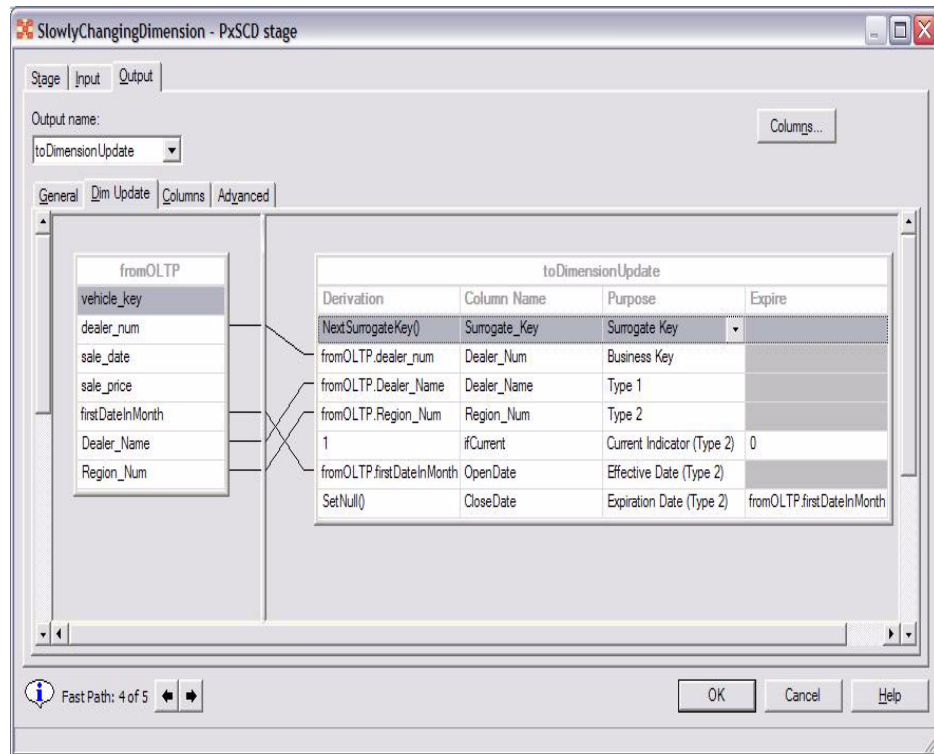


Figure 28-17 Output -> Dim Update TAB to the SCD Stage.

- iii. From the Output Link entitled, toFactTable, manage the settings to equal those as displayed in Figure 28-18.

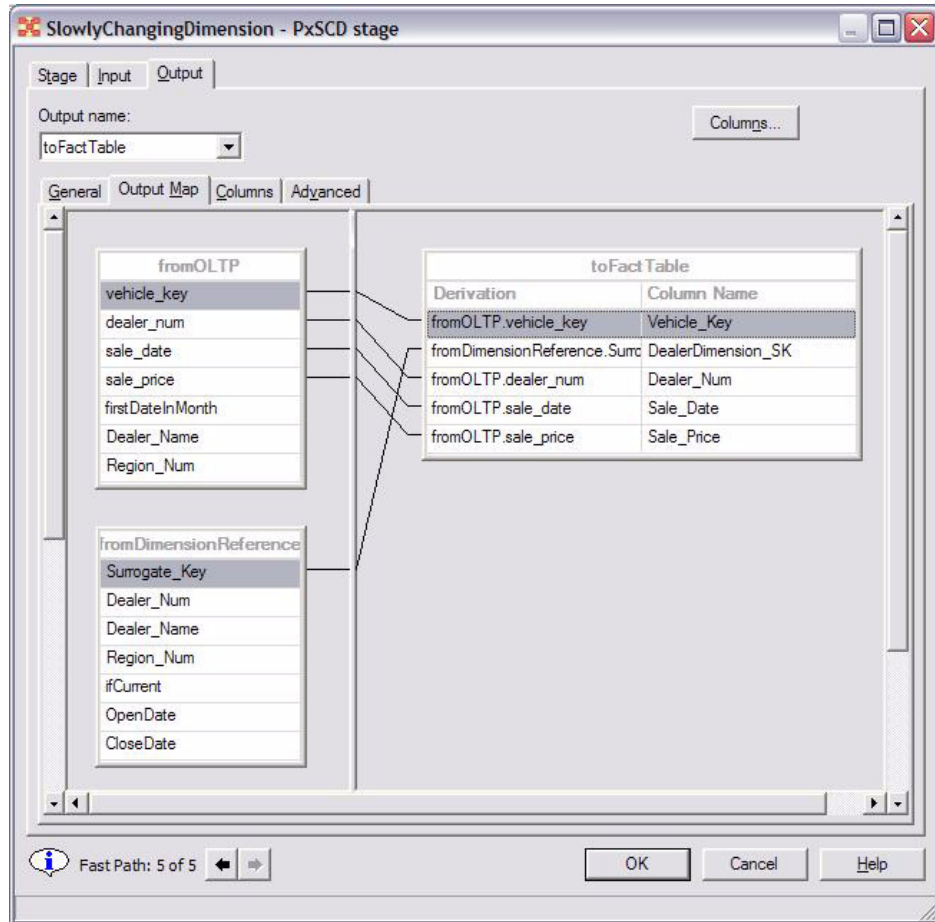


Figure 28-18 Output -> Dim Update TAB to the SCD Stage.

- f. The Output -> Columns TAB (not pictured) has two Links.
 - i. Be certain that the toFactTable Vehicle_Key column is Checked as a Key column.
 - ii. Be certain that the toDimensionUpdate Surrogate_Key column is Checked as a Key column.

7. Save, Compile and Test.

Per the suggested sample data, the Fact Table will always report 7 rows received. On this Stage we perform an Upsert, so all 7 rows are always applied, be the SQL UPDATES or SQL INSERTS.

Per the suggested sample data, the Dimension Table will always report zero rows on second and subsequent tests, this is because Updates have already been applied.

In order to reset for test, you need to execute a SQL script similar to that as displayed in Example 28-2.

28.3 In this document, we reviewed or created:

We introduced the data mart / data warehouse concept of slowly changing dimensions, OLTP (E/R) and dimensional modelling. And we detailed use of an IBM Information Server (IIS) DataStage component Stage to manage slowly changing dimensions (SCD) with ease. (Its a lot of work to create a DataStage Job to do that same work without the SCD Sage.)

Persons who help this month.

Emily Drew, Brian Caufield, and Carmen Perez.

Additional resources:

Legal statements:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating trademarks that were owned by IBM at the time this information was published. A complete and current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

Other company, product or service names may be trademarks or service marks of others.

Special attributions:

The listed trademarks of the following companies require marking and attribution:

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel trademark information

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Other company, product, or service names may be trademarks or service marks of others.