Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for different workloads (ranging from small applications that store and retrieve data on the same computer, to millions of users and computers that access huge amounts of data from the Internet at the same time).

4.1 Microsoft Business Intelligence (BI) application architecture

Before we start digging into the details of performance, let's see how a typical Microsoft Business Intelligence (BI) application architecture would look. This will give an understanding of identifying and locating the performance issues/bottlenecks.

The below **Figure 4.1** shows a typical Microsoft [42] BI application architecture which has different layers shown from left to right. On the left layer we have source systems or a relational data warehouse, in the middle layer we have the Analysis Services cube pulling data from the source systems and storing it in an Analysis Services cube/OLAP store and on the right layer we have reporting applications which consume the data from the Analysis Services cube/OLAP cube.

Although a typical Microsoft BI application architecture is to have each layer on a different physical machine, that's not usually the case. Very often we will see these layers overlap; for example in one scenario we have the relational data warehouse, Analysis Services cube and reporting services/applications on the same machine whereas in another scenario we might have a relational data warehouse and the Analysis Services cube on one machine and reporting applications on another machine or relational data warehouse on one machine and Analysis Services cube, reporting services/applications on another machine or all on separate machines.

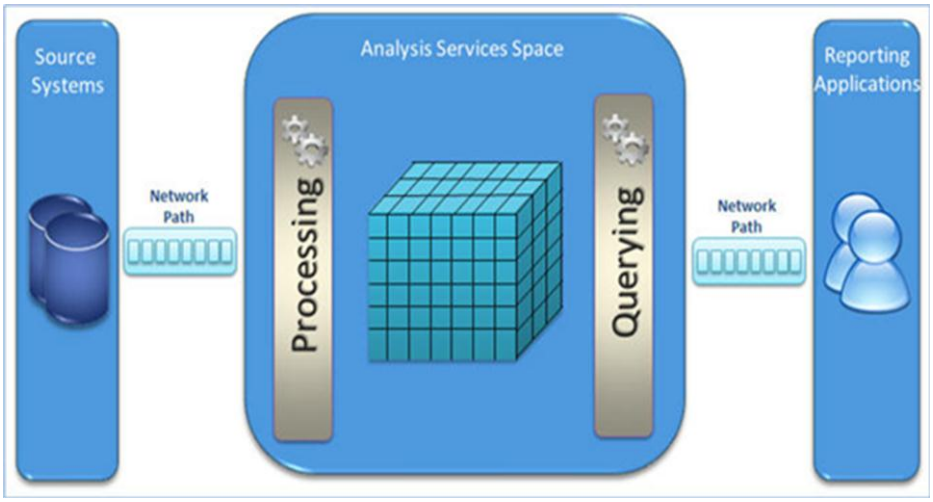

Figure 4.1 typical Microsoft BI application architecture

Whatever our system architecture or design approach is, we need to make sure that our OLAP query performance is very fast which the Microsoft BI platform is known for. But we can not overlook the processing performance as well, as this ensures data gets refreshed within the defined SLA (Service Level Agreement).

So basically when we talk of SSAS performance optimization, we need to take care of:

- Query Performance
- Processing Performance
- Proper and adequate utilization of hardware resources

Let me briefly talk about the SSAS internal architecture which will help us understand the different components in a SSAS environment, the operations they perform and how they work together as shown in **Figure 4.2** diagram.

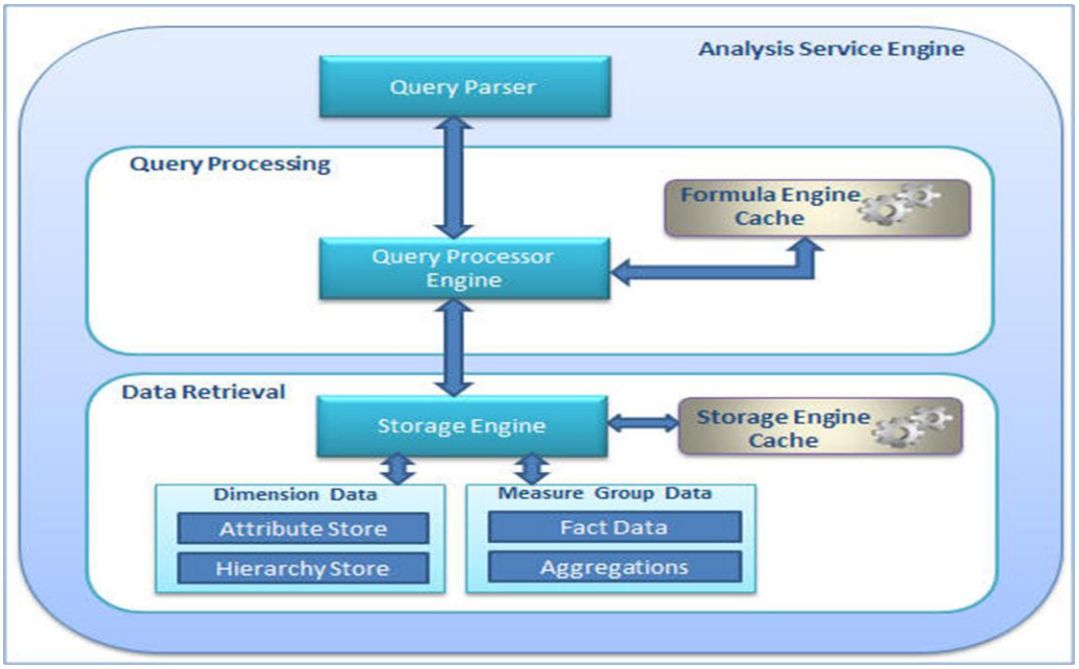

Figure 4.2 SSAS internal architecture

Query Parser

The Query Parser has an XMLA listener which accepts requests, parses the request and passes it along to the Query Processor for query execution.

Query Processor

Upon receiving the validated and parsed query from the Query Parser, the Query Processor prepares an execution plan which dictates how the requested results will be provided from the cube data and the calculations used. The Query Processor caches the calculation results in the formula engine cache (a.k.a Query Processor Cache) so it can be reused across users with the same security permissions on subsequent requests.

This summarizes the Query Processor operations:

- Makes a request for sub cube data from storage engine
  - Translation of request into sub cube data requests
- Produces result set by doing
  - Bulk calculation of sub cube
  - Cell-by-cell calculations

- Stores calculation results in formula engine cache with varying scope
    - Query scope – cache will not be shared across queries in a session
    - Session scope – cache will be shared across queries in a session
    - Global scope – cache can be shared across sessions if the sessions have the same security roles

Storage Engine

The Storage Engine responds to the sub cube data (a subset or logical unit of data for querying, caching and data retrieval) request generated by the Query Processor. It first checks if the requested sub cube data is already available in the Storage Engine cache, if yes then it serves it from there. If not then it checks if the aggregation is already available for the request, if yes then it takes the aggregations from the aggregation store and caches it to the Storage Engine cache and also sends it to Query Processor for serving the request. If not then it grabs the detail data, calculates the required aggregations, caches it to the Storage Engine and then sends it to Query Processor for serving the request.

This summarizes the Storage Engine operations:

- Creates Attribute Store (Key store, relationship store, bitmap indexes etc)
- Creates Hierarchy Store
- Creates Aggregation Store
- Storage Engine Cache
    - Loads data from storage engine cache as queries execute
    - CClears data from storage engine cache with cleaner thread (in case of memory pressure) or processing of partitions
- Aggregation Data
    - RResponds to request with aggregated values in storage
    - If new then summarizes lower level aggregated values on the fly as needed
- Fact Data
    - Scans MOLAP partitions and partitions segments in parallel
    - Uses bitmap indexes to scan pages to find requested data

While troubleshooting we need to understand which component is taking more time and needs to be optimized; such as the Query Processing Engine or Storage Engine. To understand this we can use SQL Server Profiler and capture certain events which will tell we the time taken by these components for a cold cache (empty cache, to learn more about cache warning refer to the next tip in this series):

- Storage Engine Time = Add elapsed time for each Query Subcube event
- Formula Engine = Total execution time (Query End event) – Storage Engine time (A)

If most of the time is spent in the Storage Engine with long running **Query Subcube** events, the problem is more likely with the Storage Engine. In this case we need to optimize the dimension design, design of the aggregations and create partitions to improve query performance (discussed in details in next tips in this series). If the Storage Engine is not taking much time then it is the Query Processor which is making things slow, in that case we need to focus on optimizing the MDX queries.

4.2. SSAS Developer and Admin Interfaces

Now that we have a feeling for how Analysis Services works, we need to figure out how to work with it. In this chapter, I'll review the ways of interacting with SSAS as a developer or administrator.Three interfaces are available: Business Intelligence Development Studio (BIDS), Management Studio, and PowerShell. Of these three, BIDS is perhaps the one we'll use most often.

Business Intelligence Development Studio

BIDS was introduced as the development platform for SSAS with SQL Server 2005 and Visual Studio 2005. For SQL Server 2008, BIDS is based on Visual Studio 2008[43]. As the *business intelligence* part indicates, BIDS is the development front end for all the BI services in SQL Server: Integration Services, Analysis Services, and Reporting Services.
Using Visual Studio as the platform for SQL Server projects means that cube developers have a familiar environment that has a common support base and broad community. It also reduces the footprint on the desktop when cube developers need to write code with Visual

Studio. Finally, this also means that the BI development platform is easily integrated with Team Foundation Server for project management and source control.

BIDS Is Visual Studio?

A common question seems to be "Since Visual Studio is the front end for SQL Server BI development, do we need an MSDN license for every DBA?" The answer is no.

When we install the SQL Server client tools, one of the options is to install Business Intelligence Development Studio. If we don't have Visual Studio installed, then installing BIDS will install the shell of Visual Studio with the BI project templates. **Figure 4.3** shows BIDS 2008 as installed without a prior installation of Visual Studio. If we have Visual Studio 2008 already installed, the BIDS installer will just add the BI projects to Visual Studio.

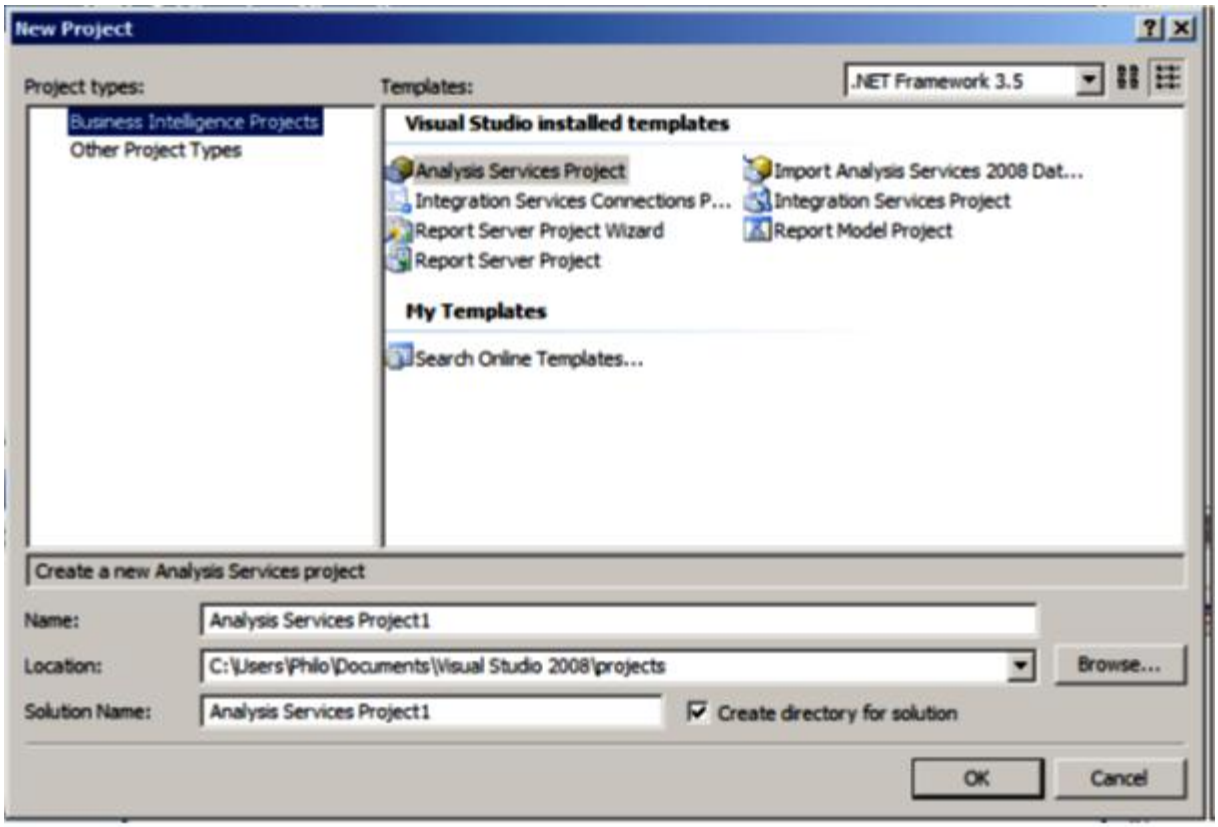

Figure 4.3 BIDS 2008 as a stand-alone install

Panes

If we haven't worked with Visual Studio before, let's take a quick tour. **Figure 4.3** shows BIDS 2008 with a cube open. The panes on the left (for measures and dimensions) are fixed in the cube designer, while the panes on the right (Solution Explorer and Properties) are Visual Studio standard panes and can be detached and moved around.

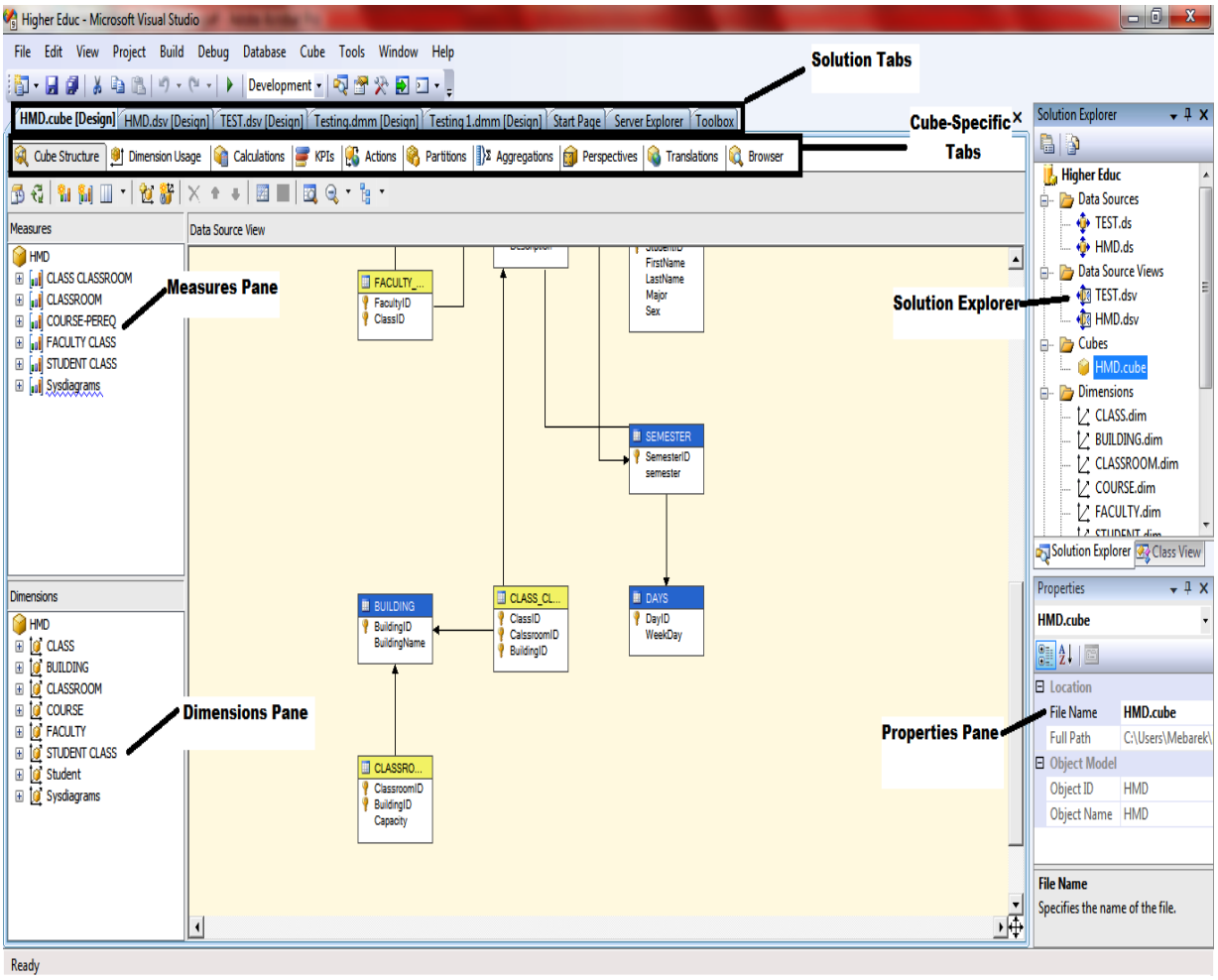

Figure 4.4 BIDS 2008

The solution tabs at the front are where we'll be able to select different objects as we open them (data source views, cubes, dimensions, and so forth). Underneath those tabs are a row of tabs specific to the object we have open. we can see the tabs for the cube in

Figure 4.4 for a dimension, we'd have just four tabs: Dimension Structure, Attribute Relationships, Translations, and the Browser.

we'll be diving into the designers in depth in future chapters. My goal in this chapter is just some basic familiarization with Visual Studio itself.

we can arrange the Visual Studio panes to suit our needs, or tuck them out of the way. Clicking the little thumbtack in the pane's title bar makes the pane autohide to a tab along the side of the studio.

Clicking the tab slides the pane back in. we can click the thumbtack again to pin the pane in place. we can also "tear off" a pane so it floats, or dock it somewhere else. Just grab the title bar and drag the pane, and it will detach.

Solution Explorer

The Solution Explorer **Figure 4.5** is the organizer for the objects that make up our OLAP solution. The objects generally map to files in the solution. Double-clicking an object opens it in the designer (except for Data Sources—double-clicking it just opens the Data Source Designer dialog box). Right-clicking the solution name and selecting Properties opens the solution property pages, which is where we can set the deployment location.

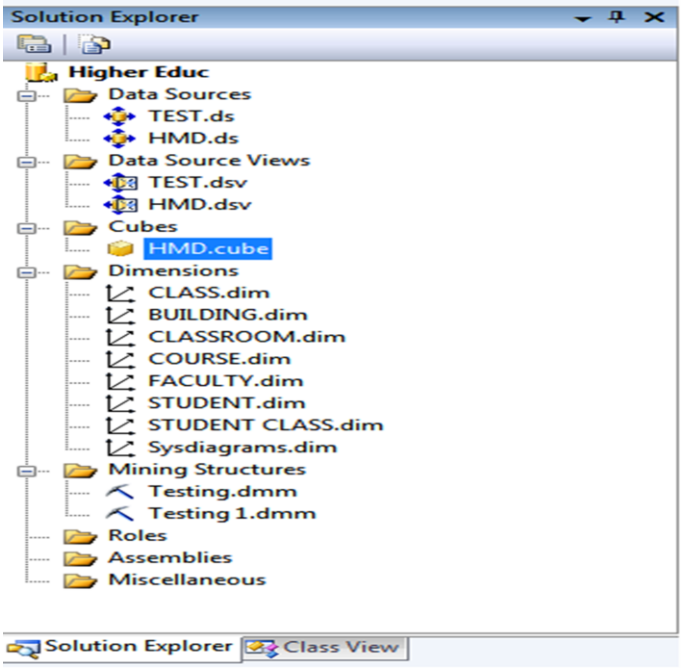

Figure 4.5 The Solution Explorer pane

4.3 Desinging Data warehouse

The Motivation for the Data Warehouse system

The day-to-day operations of the school rely heavily upon the OLTP system. The staffs make use of OLTP system for management information system. The need for strategic decision necessitates the development of the data warehouse. The OLTP can only handle few historical data which are not enough to make strategic decisions. With a data warehouse and OLAP, staff and management can perform roll-up and drill-down operations to student enrollments by year, by semester by course or any combination desired. The system will provide decision support that is flexible and user-friendly. The University would like to use the data-warehouse to answer questions such as: What is the trend of student admission, student's enrollments, Lecturers room schedule, number of annual enrollment etc. These types of questions need a lot of historical data to generate which the current system OLTP system cannot support.

The logical Design

The logical design of data-warehouse is defined by the dimensional data modeling approach. The dimensioning design process followed in this project adheres to the methodology described by Kimball and Ross (2002).

Unlike the Entity Relationship (ER) and Unified Modeling Language (UML) data modeling processes, the logical design of Data Warehouse (DW) is defined by the Dimensional data modeling approach. To minimize the join operations which slow down queries, normalization is not the guiding principle in DW design. A schema is a collection of database objects, including tables, views, indexes, and synonyms. There is a variety of ways of arranging schema objects in the schema models designed for data warehousing. The following are the two types of schemas commonly used in dimensional data modeling.

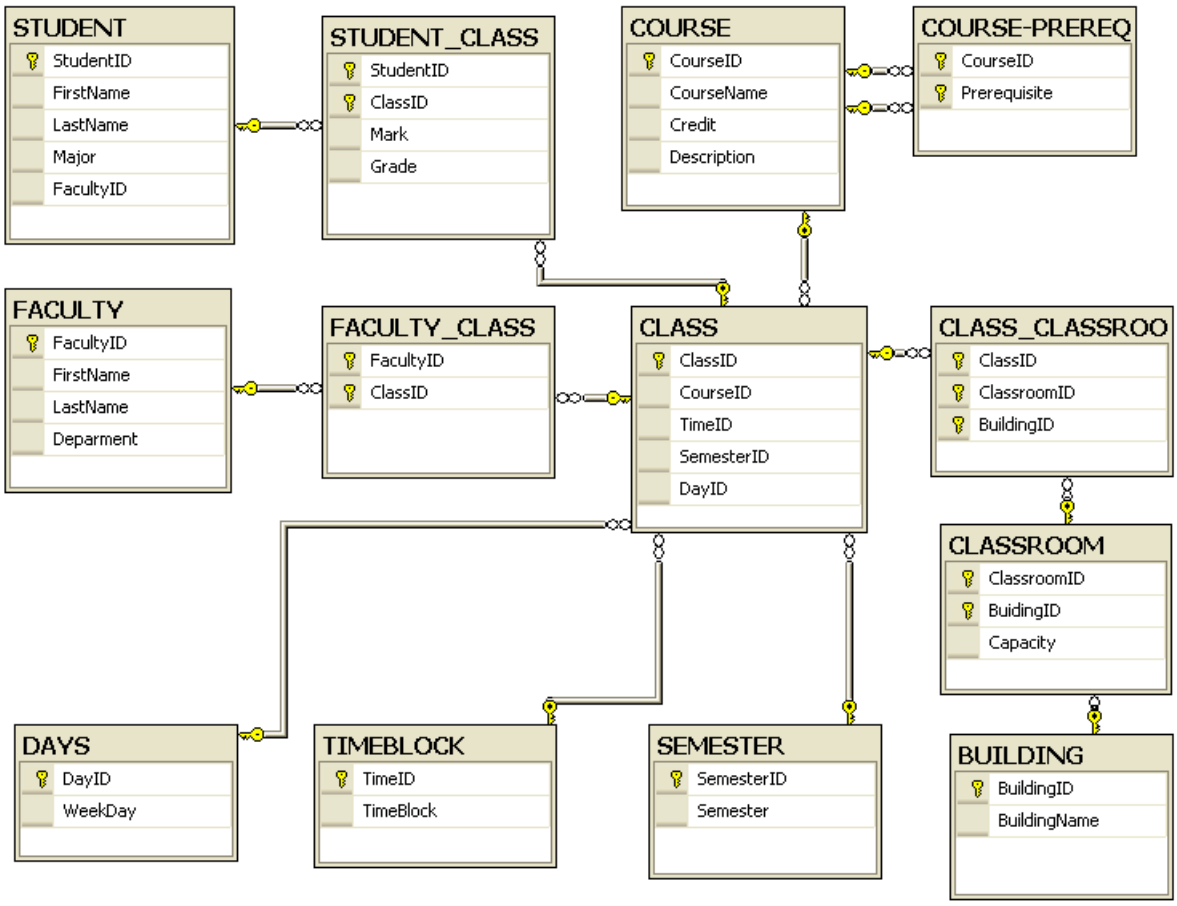

Figure 4.6 The logical design of OLTP for class registration database

Dimensional Data Modeling approach

The dimensional approach is quite different from the normalization approach followed when designing a database for daily operations. **Figure 4.7** shows the five dimension tables used in the project.

Data Hierarchies in Dimensional tables

Each of the dimensions contains at least one hierarchy. The hierarchies allow users to analyze data aggregations using the OLAP. This allows related items to be grouped and summarized for high level analysis while retaining the ability to drill down to more specific product detail.

For analyzing the data warehouse. For example we can look for Faculty who taught a particular course.

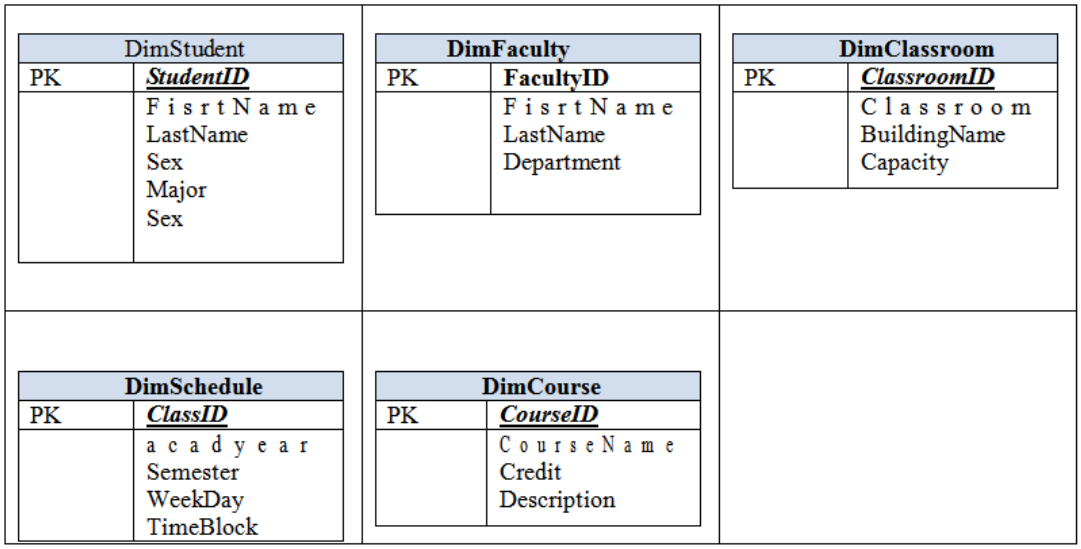

Figure 4.7 The Dimensional tables showing data hierarchies

Fact Table

Fact table contains dimension attributes and measures. Dimension attributes are FKs or other attributes called degenerate dimension (<<dd>>). Measures are the values to be aggregated when queries group rows together. The Fact table is composed of two types of attributes: dimension attributes and measures.

Figure 4.8 presents the fact table for this research. In the table, the registered field is used as a measure while the other keys are used to link the dimension tables.

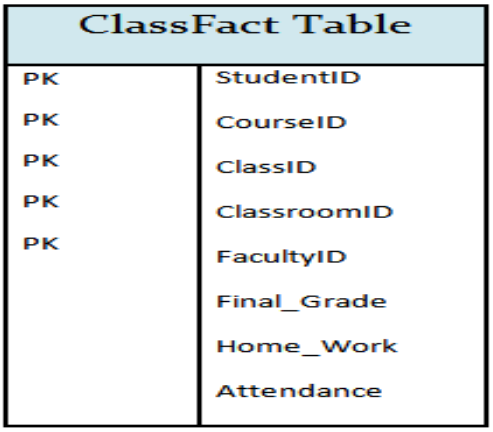

Figure 4.8 the Fact Table

The Star Schema:

Data Warehouse are commonly organized with one large central fact table, and many smaller dimensions tables. This configuration is termed a star schema. A star schema has been adopted in this research. Below is the diagram of the relationship between the fact table and the Dimensional tables.

In the **Figure 4.9** the relationships between the fact table and dimension tables are depicted in a star form.

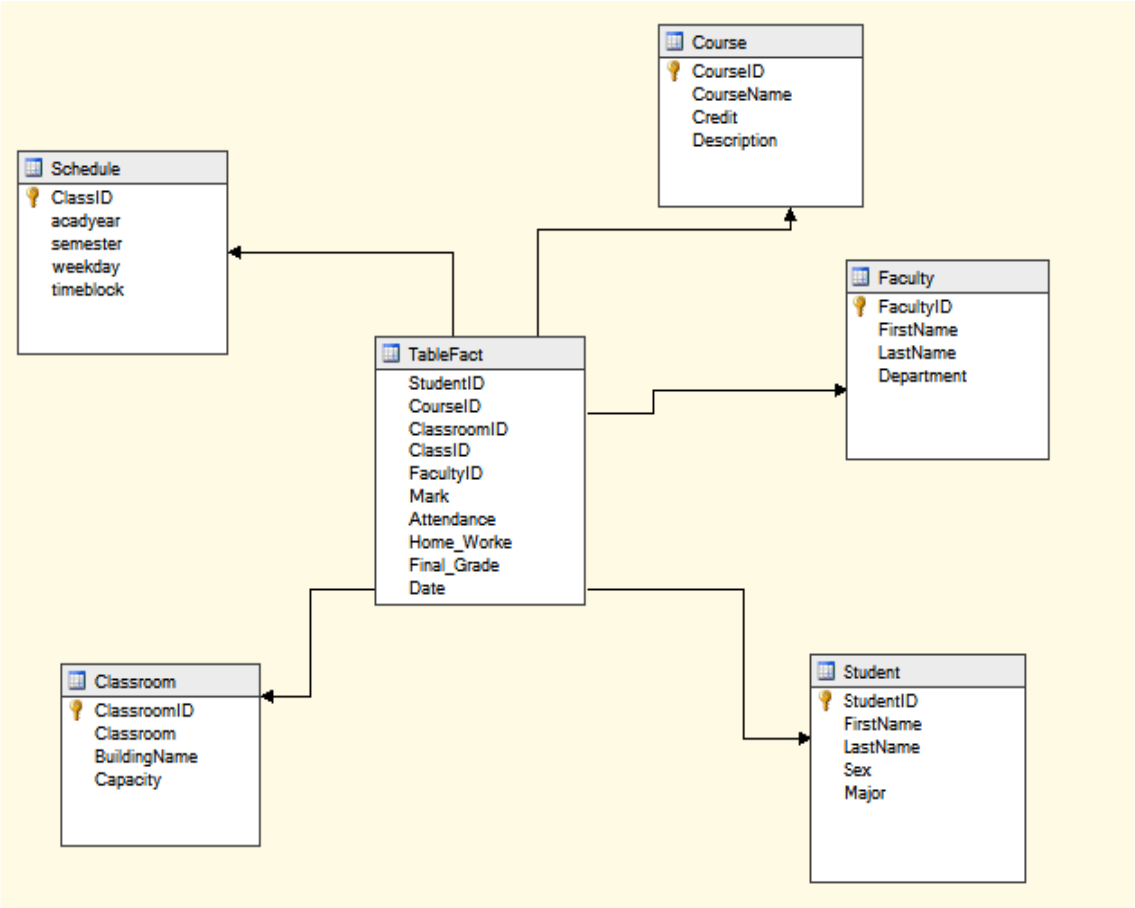

Figure 4.9 The Star schema

4.4 Implementing an Analysis Services Data warehouse

At the core of the SQL Server relational database management system (RDBMS) lies the database engine, which is optimized to support thousands of concurrent transactions while ensuring the integrity of the data and the management of resources. Applications accessing data in a well-designed relational database can swiftly retrieve and update the data—that is, as long as the queries used by the application don't try to aggregate large volumes of information. When such queries do have to be performed, a SQL Server Analysis Services cube often provides a more efficient solution than the typical relational database. An Analysis Services cube is a multidimensional structure that pre-aggregates data across multiple dimensions. the cube can store the pre-calculated aggregation. Unlike the SQL Server database engine, which supports online transaction processing (OLTP) of data in a relational database, **the Analysis Services engine supports online analytical processing** (OLAP) of data stored in a multidimensional cube—the main component in an Analysis Services database. An Analysis Services database is a logical collection of objects that include one or more cubes as well as other components that facilitate OLAP operations.

In this Chapiter, we discuss the basic components that make up an Analysis Services database and describe how to implement those components. To demonstrate these concepts, we created an Analysis Services project in SQL Server 2008 Business Intelligence Development Studio and named the project Final_Year. The project connects to the Final Master Degree. Sample data warehouse on a local instance of SQL Server 2008. By default, Business Intelligence Development Studio assigns the project name to the database; however, as we'll see later in the article, we can rename the database when we deploy it to an instance of SQL Server.

To implement an Analysis Services database, we need to take, at a minimum, the following steps:

1. Create a data source.
2. Create a data source view.
3. Create one or more dimensions.
4. Create a cube.
5. Deploy the database.

The rest of the Chapiter covers each of these steps and demonstrates how to carry them out. For the most part, we focus on *how* to perform these steps and offer some of the *why* along with my explanations.

Creating the Data Source

An Analysis Services data source defines the connection to a specific source of data. For example, it is common for a data source to define the connection to a SQL Server database. However,we can also configure a data source to connect to other sources, such as a Microsoft Access or Oracle database. When we configure a data source, we must specify a provider (such as SQL Server Native Client 10.0) and then supply the information necessary to establish a connection to the data source, using the specified provider. To create a data source in an Analysis Service project (in Business Intelligence Development Studio), right-click the Data Sources node in Solution Explorer, and then click new Data Source. This launches the Data Source wizard, which walks we through the steps necessary to create a data source. When we create a data source, we must define a data connection (unless one has already been created). To create a connection, For My work, we used the default provider, Native OLE DB/ SQL Server Native Client 10.0, and connected to a local instance of SQL Server 2008, specifying "Final Master Degree" as the database. After setting up the connection, we must configure the impersonation information. Figure shows the Impersonation Information screen of the Data Source wizard. For the example project in this article, we use the service account for my data source connection.

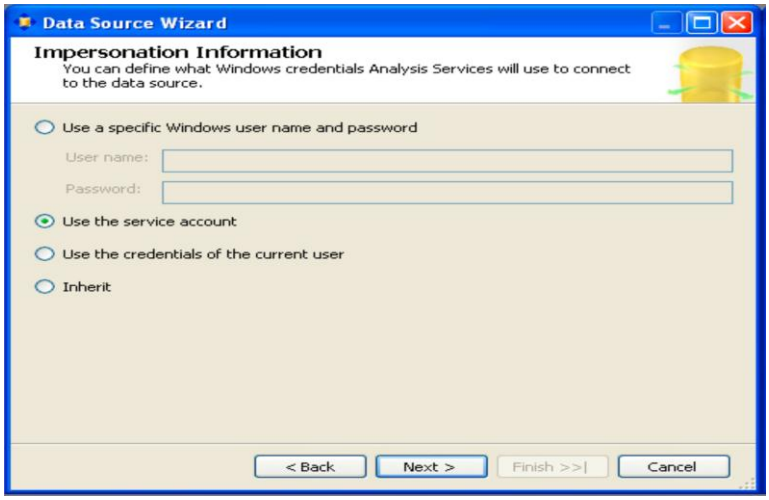

Figure 4.10: Specifying the impersonation information in for a data source

Creating the Data Source View

The next step in setting up an Analysis Services database is to create a data source view. A data source view provides an abstraction layer between the data source and other components in the database, such as dimensions and cubes. The data source view contains the metadata for objects selected from the data source and the relationships between those objects, providing a logical model of the underlying data source. we can enhance a data source view with elements that extend the data source. For example, we can add calculated columns or provide friendly names for tables and views. An Analysis Services database can include one or more data source views.

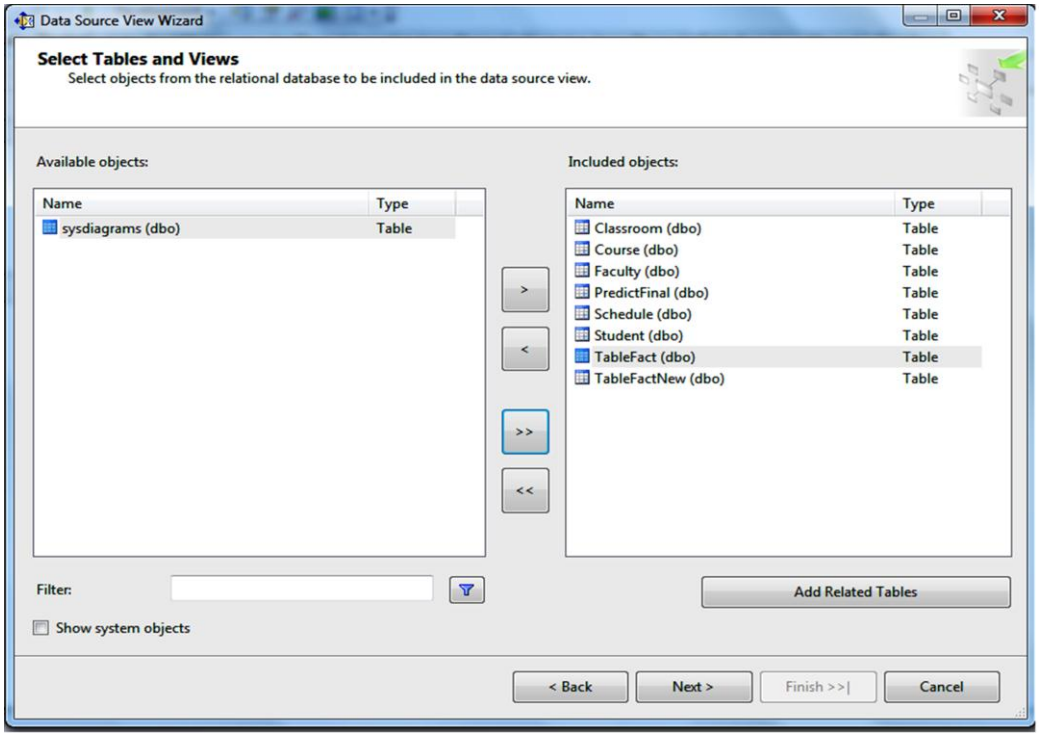

Figure 4.11: Selecting the tables and views for a data source view

As we can see in Figure 3, the design surface includes the Diagram Organizer pane, which lists the initial diagram (<All Tables>). The diagram itself is displayed in the main window. Notice that the design surface also includes the Tables pane, which lists the tables included in the data source view.

we can add diagrams to our data source view that are based on those tables, and we can use any of the tables in any of the diagrams. For example, we created, one diagram based on Student Informations. **Figure 4.12** shows the diagram we created for Student.

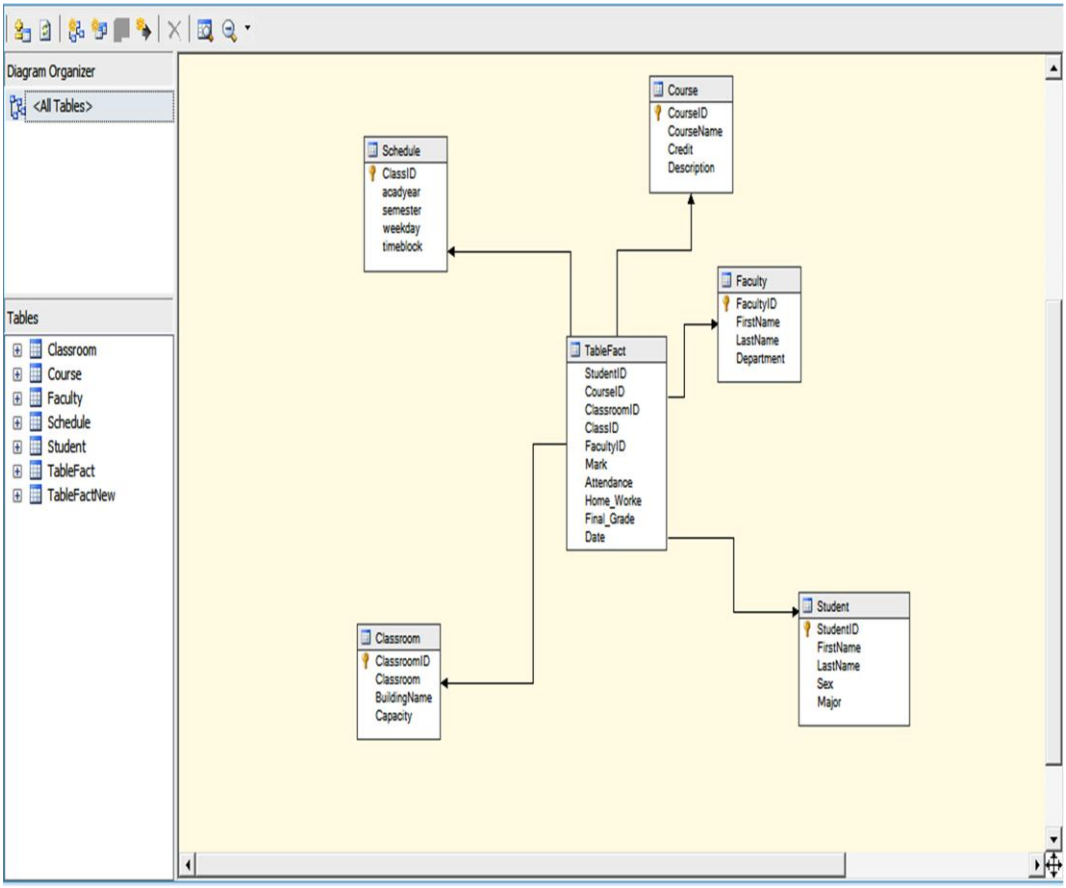

Figure 4.12 Viewing the Student fact table diagram

- **Explore Student Data**

MEBAREK-PC.Final_...e - dbo.TableFact | MEBAREK-PC.Final_...ree - dbo.Student | MEBAREK-PC.F

StudentID	FirstName	LastName	Sex	Major
1	MEBAREK	GHODBANE	M	INFO
2	SALAH	GHODBANE	M	INFO
3	ABDO	GODBANE	M	INFO
4	ZAKI	BEL	M	INFO
5	AMINA	SAF	F	INFO
6	IMEN	SAF	F	INFO
7	MAJED	GHODBANE	M	INFO
8	AMIRA	DODO	F	INFO
9	SAMIRA	BOUB	F	INFO
10	SALIHA	BABA	F	INFO
11	IMEN	SOUS	F	INFO
12	AMIRA	DIDI	F	INFO
13	OSAMA	SOSO	M	INFO
14	ABDELNOUR	MICH	M	INFO
15	KHADIJA	MICH	F	INFO
16	SOUSUO	SOUSSS	F	INFO
NULL	NULL	NULL	NULL	NULL

- **Explore Faculty Data**

MEBAREK-PC.Final_...ree - dbo.Faculty | MEBAREK-PC.Final_...e - dbo.TableFact

FacultyID	FirstName	LastName	Department
1	DATA	MINING	INFO
2	TETO	SETO	ST
3	RAR	FAFA	BOILOGIE
NULL	NULL	NULL	NULL

- **Explore Classroom Data**

MEBAREK-PC.Final_...e - dbo.Classroom | MEBAREK-PC.Final_...ree - dbo.Faculty

ClassroomID	Classroom	BuildingName	Capacity
1	16	M	60
2	22	S	80
3	3	C	100
NULL	NULL	NULL	NULL

- **Explore Course Data**

MEBAREK-PC.Final...gree - dbo.Course | MEBAREK-PC.Final_...e - dbo.Classroom

CourseID	CourseName	Credit	Description
1	DATA_MINING	9	DATA MININIG
2	SEM	6	SEMULATION
3	WEB	9	ONTOLOLGY WEB
4	ENGLISH	3	ENGLISH
5	Art_Int	7	FRANCER
NULL	NULL	NULL	NULL

- **Explore Schedule Data**

MEBAREK-PC.Final...ee - dbo.Schedule | MEBAREK-PC.Final...gree - dbo.Course | MEBAREK-PC.Fin

ClassID	acadyear	semester	weekday	timeblock
1	2011	FIRST	MONDAY	8-9
2	2011	SECONDE	SUNDAY	11-12
3	2011	FIRST	SATURDAY	2-3
4	2011	FIRST	WEDNESDAY	3-4
NULL	NULL	NULL	NULL	NULL

- **Explore Schedule Data**

MEBAREK-PC.Final_...e - dbo.TableFact | MEBAREK-PC.Final...ee - dbo.Schedule | MEBAREK-PC.Final...gree - dbo.Course

StudentID	CourseID	ClassroomID	ClassID	FacultyID	Mark	Attendance	Home_Worke	Final_Grade
14	1	1	1	1	A	Medium	Average	1
14	2	1	1	1	B	high	Bad	1
14	3	1	1	1	A	Low	Average	0
14	4	1	1	1	C	Medium	Bad	1
14	5	1	1	1	A	high	Excellent	0
15	1	1	1	1	C	Low	Bad	0
15	2	1	1	1	C	Low	Excellent	1
15	3	1	1	1	C	Medium	Average	1
15	4	1	1	1	A	high	Bad	0
15	5	1	1	1	A	Low	Excellent	0
16	1	1	1	1	B	Medium	Average	1
16	2	1	1	1	C	high	Average	0
16	3	1	1	1	B	Low	Excellent	1
16	4	1	1	1	C	Medium	Bad	1
16	5	1	1	1	A	high	Excellent	0
5	1	1	1	1	B	high	Average	0
5	2	1	1	1	C	Low	Excellent	1
5	3	1	1	1	B	Medium	Average	0
5	4	1	1	1	A	high	Excellent	1
5	5	1	1	1	A	Low	Average	1
6	1	1	1	1	C	Medium	Excellent	0
6	2	1	1	1	C	high	Bad	1
6	3	1	1	1	B	Low	Excellent	0
6	4	1	1	1	B	Medium	Bad	0

Creating the Cube

The fundamental components of a cube are dimensions and measures. A dimension in a cube is referred to as a *cube dimension* and is an instance of a database dimension. Measures are the facts that are being described by the dimensions. For example, the measure can be aggregated along dimensions. Measures are grouped together into logical containers called measure groups. The measure groups tie together dimensions within the structure of the cube. Let's create a cube so we can better see how this works.

To specify which tables to use, select them in the **Select Measure Group Tables** screen of the Cube wizard, as shown in Figure. As we can see in the figure

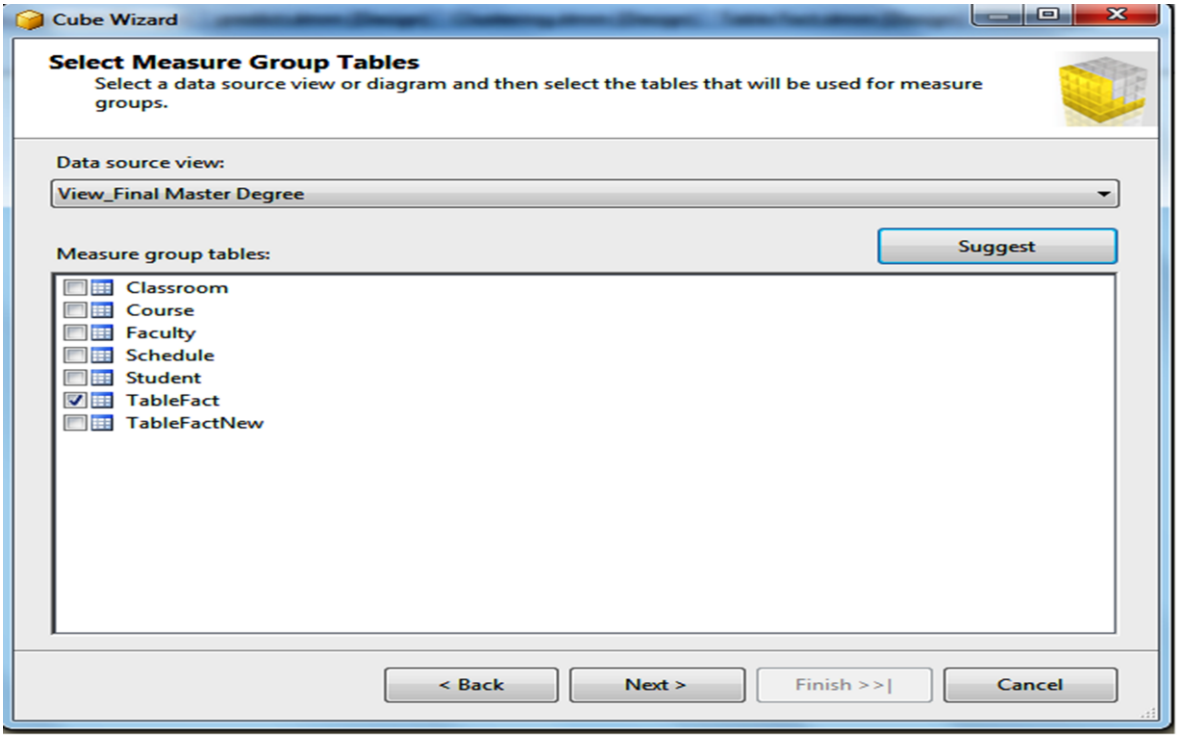

Figure 4.13: Selecting the measure group tables for Cube Final Master Degree

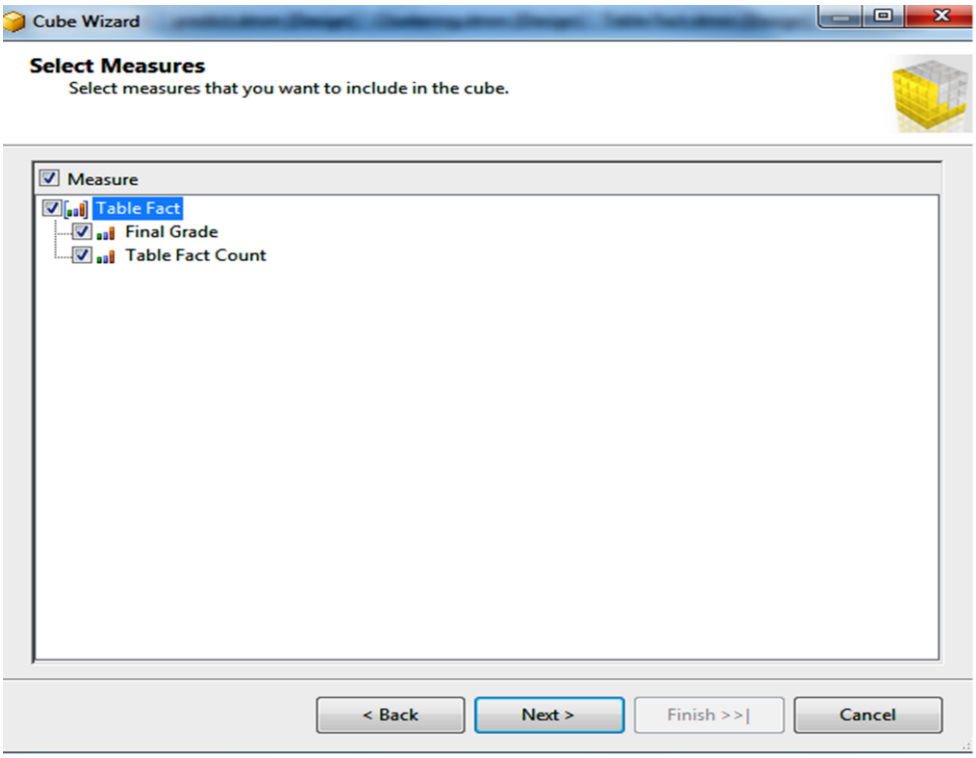

Figure 4.14: Selecting the measures for Cube_Final Master Degree

The next step in creating a cube is to select which database dimensions we want to include as cube dimensions, as shown in **Figure 4.15**. Because we created the database dimensions specifically for this cube.

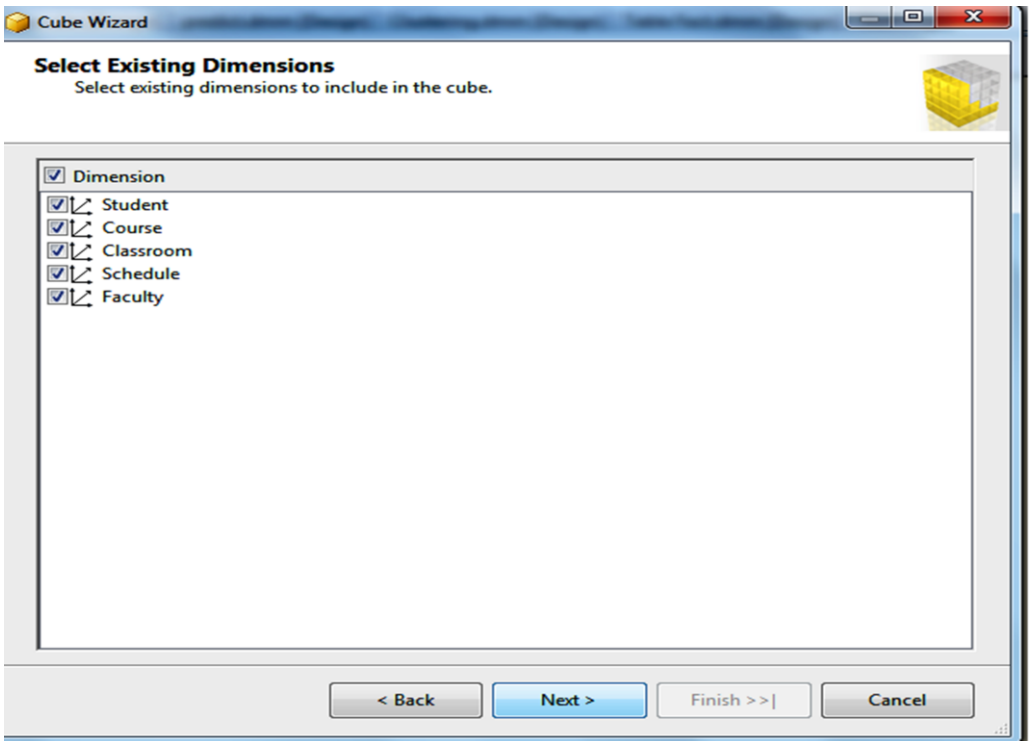

Figure 4.15 Selecting the dimensions for Cube_Final Master Degree

The final step in creating the cube is to decide on a name. (I chose Cube_Final Master Degree.cube.) After that, when we complete the wizard, Business Intelligence Development Studio generates the cube and opens it in the designer surface **Figure 4.16** shows we what the Cube Designer would look like for the Final_Year project.

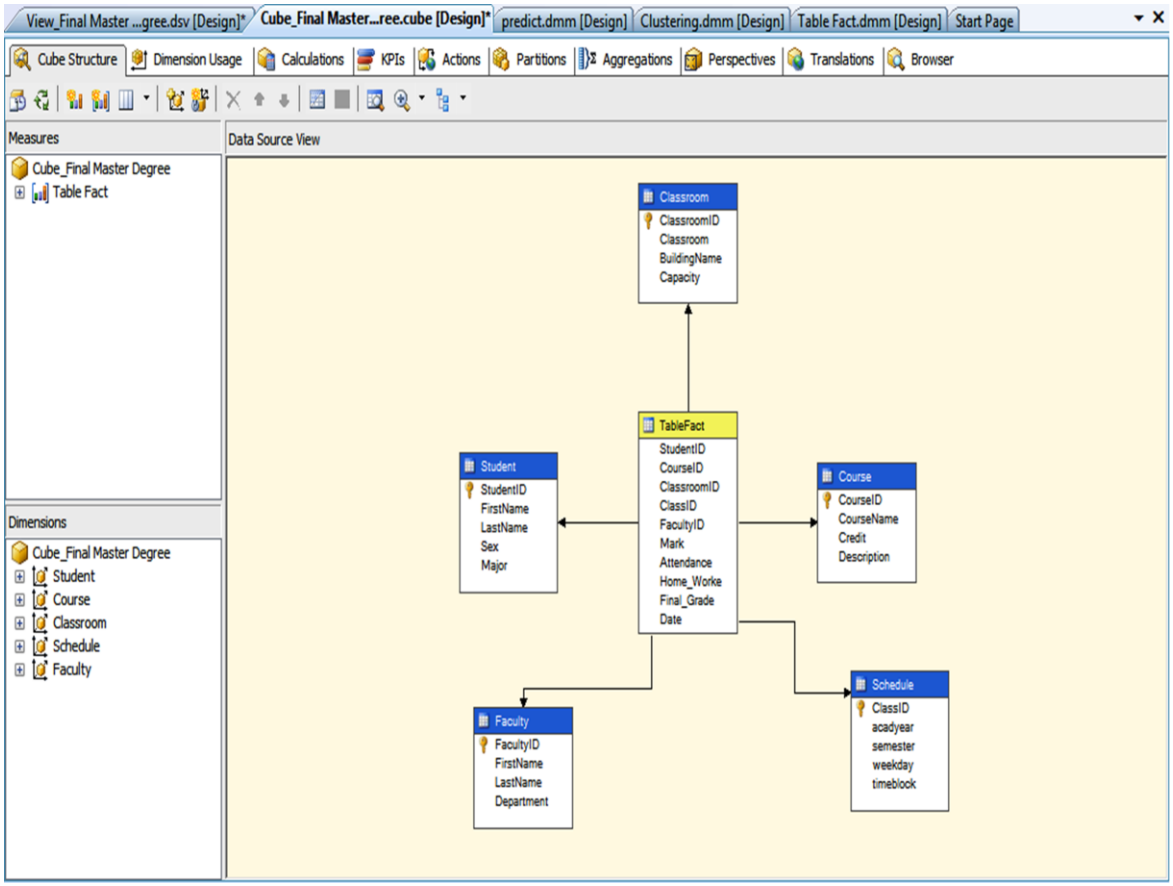

Figure 4.16 Viewing the AW_Sales cube in Cube Designer

Notice that the Cube Structure tab displays the Measures pane, which allows us to view the available measure groups and their measures. we can also create new measures and measure groups as well as delete them. Cube Designer also includes the Dimensions pane, which lists the cube dimensions and their attributes. In addition to viewing the dimensions and attributes, we can add and delete them as well as edit their properties.

The main window in Cube Designer displays the data source view on which the cube is based. The other tabs provide details about other components we can add to the cube, although those components are beyond the scope of this article. However, we do plan to discuss them in later.

That's all there is to setting up a basic cube. Once we've put all the pieces together, we're about ready to deploy that database, so let's move on to the next step.

Deploying the Multidimensional Database

Before we deploy our database, we must ensure that we send it to the correct instance of SQL Server, which means we have to set the deployment properties. as shown in **Figure 4.17**. As the figure shows, I'm deploying the database to a local instance of SQL Server.

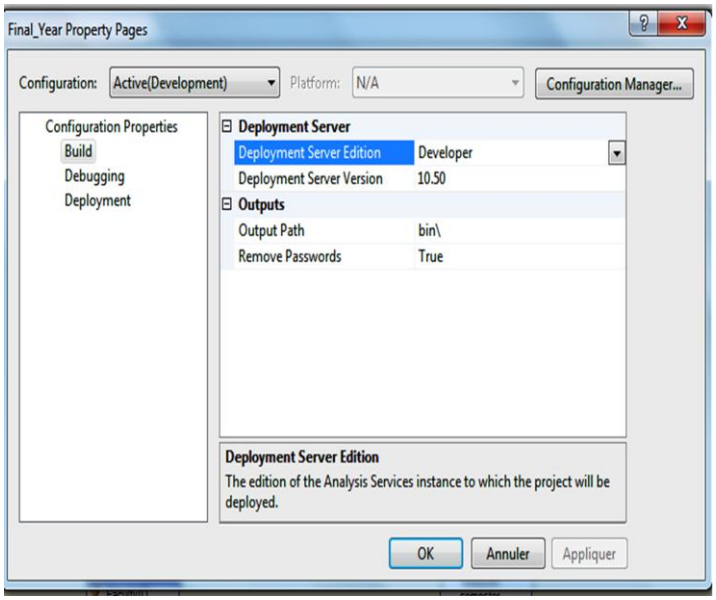

Figure 4.17 Configuring the deployment properties for the Final Master Degree database

After we've configured the properties, we can deploy the database to the SQL Server instance. To deploy the database, right-click the project name in Solution Explorer, and then click **Deploy**. During the deployment process, Business Intelligence Development Studio displays the progress of the deployment process and shows any relevant messages in the Output pane at the bottom of the window. If the database is deployed with no problems, we'll receive a message saying that the deployment is complete. we can verify that the database has been deployed by using SQL Server Management Studio to connect to Analysis Services on the specified instance of SQL Server. The database should be listed beneath the **Databases** node and should include the data source, data source view, dimensions, and cube. After we've deployed the database, we can browse the cube in either Management Studio or Business Intelligence Development Studio. In Development Studio, click the Browser tab of Cube Designer to view the work area available for rendering cube data, as shown in **Figure 4.18**.

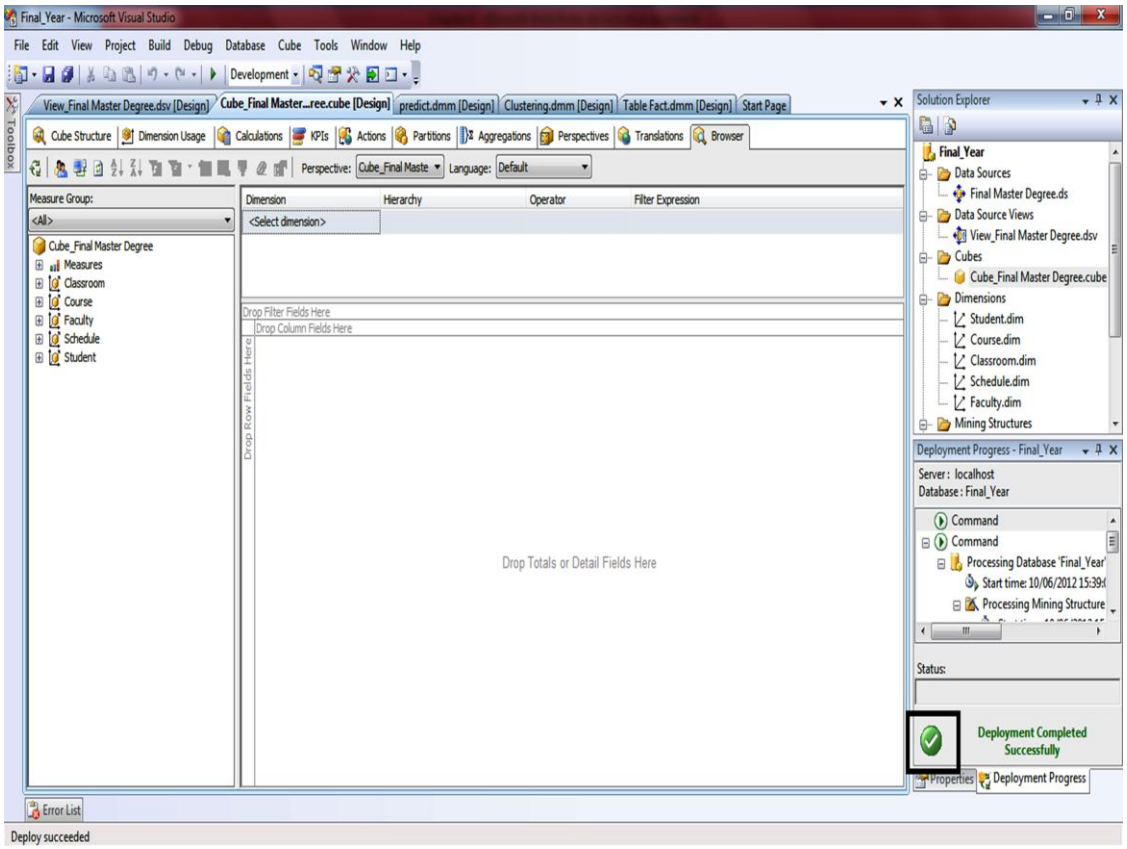

Figure 4.18 Viewing the Browser tab in Cube Designer

4.5 implementing data mining structures

Why Mine Data?

Data mining provides a powerful decision support mechanism. Data mining is used in virtually every industry by marketing departments, customer relationship management groups, network (e-mail) security, finance/credit departments, human resources (employee performance), and others.

The following example offer a few ways that we have seen data mining used:

Education: An independent school district mined statewide standardized testing scores and other demographic data to predict students at risk of underperforming or failing these tests at the grade and subject levels. These efforts resulted in a new initiative by the district to group at-risk students together, not only in smaller classes, but in smaller classes focused on the particular subjects these students were struggling in. Along with this, teacher

assignments were reevaluated to enable teachers who performed better with at-risk students to head these classes. Not only did test scores show improvement, but discipline-related issues decreased as well.

Microsoft Decision Trees

The Microsoft Decision Trees algorithm creates predictions for classification and association. A decision tree resembles the structure of a B-tree. As part of the data-mining business process, decision trees are often run against individual segments from a Clustering model.

A Microsoft Decision Tree model requires the following:

Input: Input data can be either discrete or continuous.

Key column: Like Naïve Bayes and Clustering, a unique identifier is required for each record in the training set.

Prediction column: This is the prediction attribute. It can be either discrete or continuous for this algorithm. If we use a discrete attribute, the tree can be well represented by a histogram.

Microsoft Naïve Bayes

The Microsoft Naïve Bayes algorithm creates classification predictions. The Naïve Bayes algorithm is simply a true-or-false mechanism, and does not account for any types of dependencies that may exist in the data we are analyzing. In order to create a Naïve Bayes model, we need to prepare our data and create a training data set. This need for a training data set marks Naïve Bayes as one of the *supervised* mining algorithms.

A Microsoft Naïve Bayes model requires the following training data:

Input: Input data columns must be discretized, because Naïve Bayes is not designed to be used with continuous data, and our inputs should be fairly independent of each other. One method of discretization is known as *banding,* or *binning*, data. Two common examples of banding data include age and income. Age is often discretized as Child, Teenager, Adult, and Senior Citizen. It may also be banded as < 11, 12–17, 18–54, and 55+. Income is often banded in brackets, such as < 20,000, 20,000–50,000, and 50,000+.

Key column: A unique identifier is required for each record in the training set. Two examples of unique identifiers are a surrogate key and a globally unique identifier (GUID).

A surrogate key in the EDW is usually a SQL Server Identity column, which is basically a numeric counter, guaranteed to be unique by the database engine.

Prediction column: This is the the prediction attribute. Will the customer buy more of our products? Is the applicant a good credit risk?

In the Select Data Source View dialog box, shown in **Figure**, choose Accessory Campaign from our Available Data Source Views, and click next.

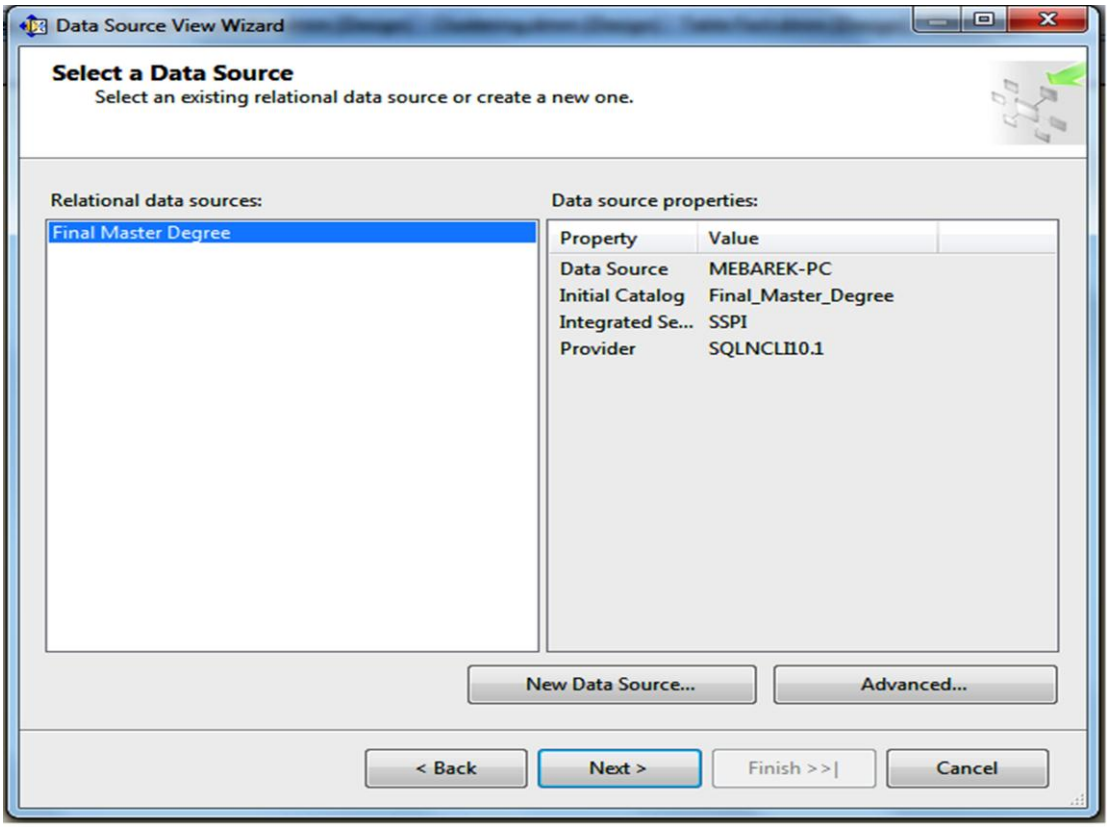

Figure 4.19 Selecting the Final Master Degree data source view

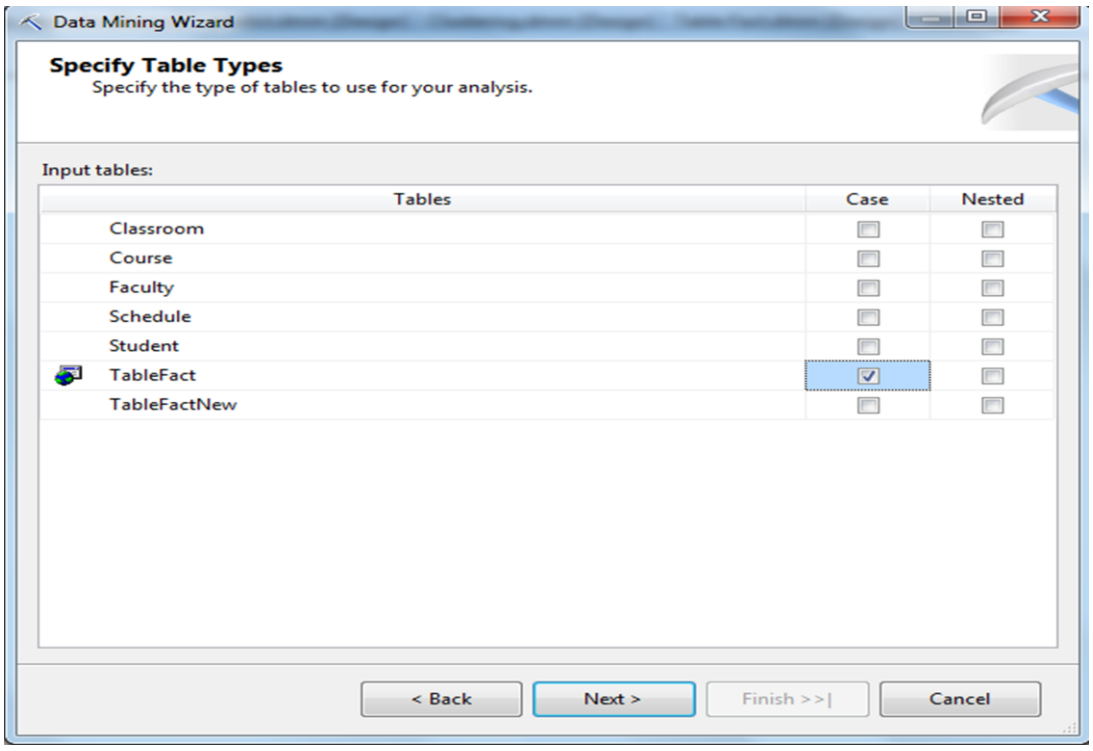

Figure 4.20 Choosing TableFact as My Case table

fields: Mark, Home_Work, Attendance. This model requires one Predictable column; choose Final Grdae. our selections should mimic **Figure 4.21**. When we are finished, click next.

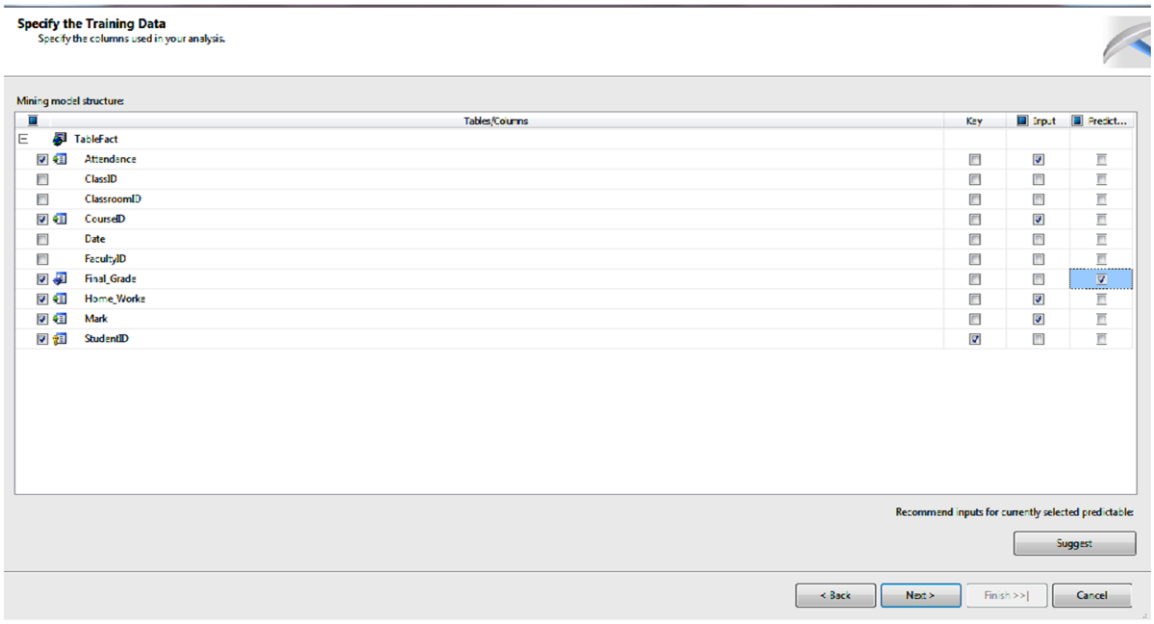

Figure 4.21 Choosing training data

The Create Testing Set dialog box, shown in **Figure 4.22**, is where we will specify some inputs regarding how we would like our model to be trained. For this, leave the Percentage of Data for Testing option set to 15 percent and Maximum Number of Cases in Testing Data blank. Click Next.

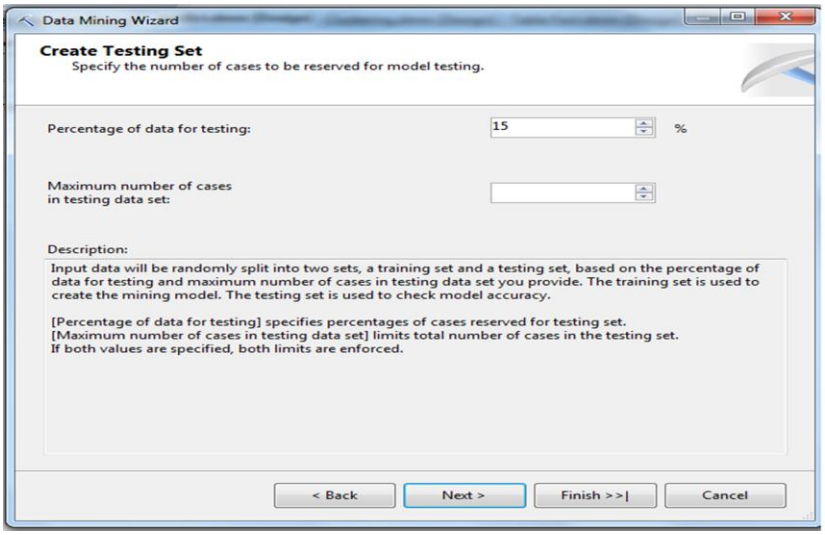

Figure 4.22. Specifying column content and type

Using the Data Mining Model Designer

The Data Mining Model Designer will be our main work area, now that we have finished defining our model with the Data Mining Wizard. The Data Mining Model Designer consists of the following tabs:

Mining Structure: This is where we modify and process our mining structure.

Mining Models: Here we create or modify models from our mining structure.

Mining Model Viewer: This view enables we to explore the models we have created.

Mining Accuracy Chart: Here we can view various mining charts. Later in this chapter, we will use this tab to look at and review a lift chart.

Mining Model Prediction: Using this view, we will create and review the predictions our mining model asserts.

The Mining Structure View

The Mining Structure view is separated into two panes, as shown in **Figure 4.23**. The leftmost pane displays our mining structure columns, and our data source view is shown on the right. we will also process our mining model here, using the Process the Mining Structure button on the toolbar. Click the Process the Mining Structure button (leftmost button in view toolbar) now to begin processing.

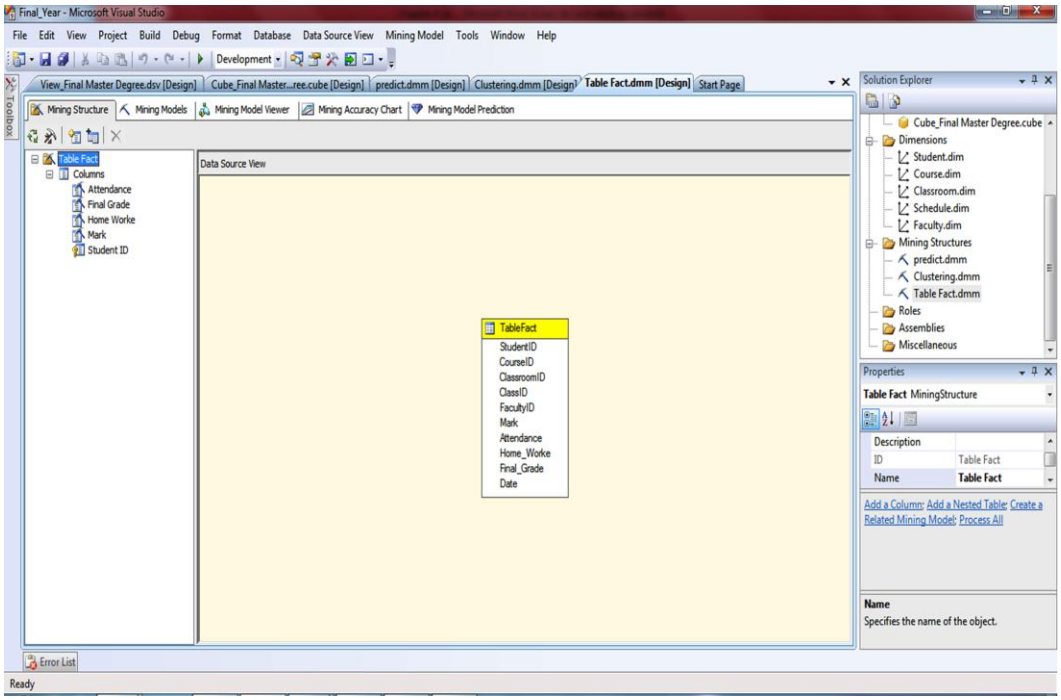

Figure 4.23 The Mining Structure tab

The Mining Model Viewer View

The Mining Model Viewer is a container that supports several viewers. Choosing Microsoft Tree Viewer from the Viewer drop-down will load our Final Grade into the Tree Viewer. After the Tree Viewer has loaded, ensure that Final Grade is selected in the Tree drop-down, which is just below the Viewer drop-down list. In this section, we will show us the Microsoft Tree Viewer, the Mining Legend, and this model's drill-through capability.

Exploring the Decision Tree and Naïve Bayes

The decision tree is built as a series of splits, from left to right, as shown in **Figure 4.24** . The leftmost node is all, which represents the entire Students. The background color of each node is an indication of that node's population; the darker the color, the greater the population.

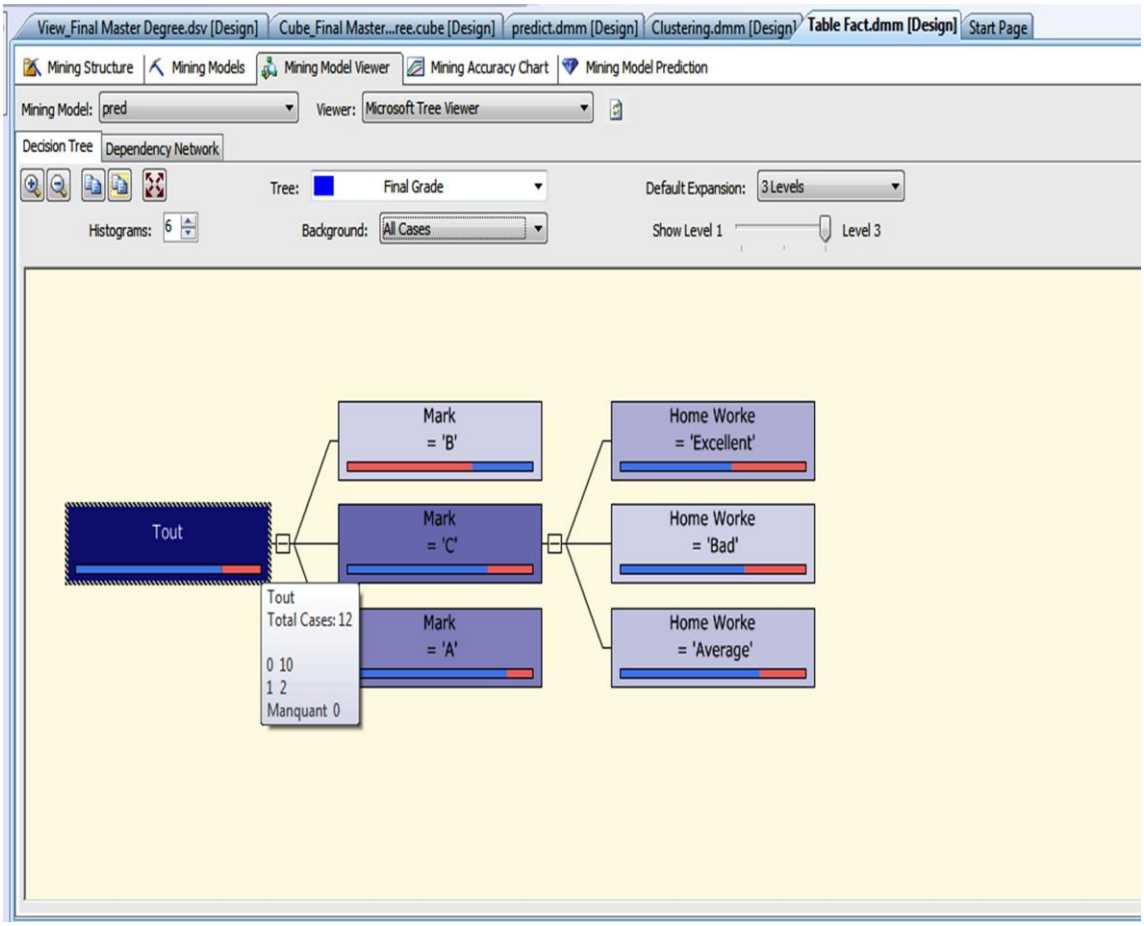

Figure 4.24 the Decision Tree

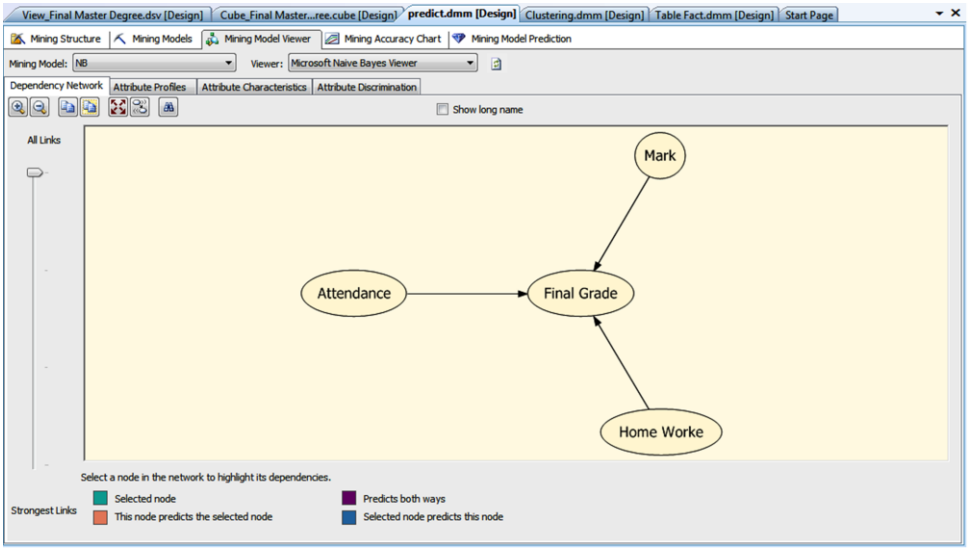

Figure 4.25 Naïve Bayes

Each node in the bar under the node condition is a histogram that represents our Students. The blue portion of the bar is our True state, and the pink our False state. Click the All node and select Show Legend. After the legend appears, dock it in the lower-right corner, as shown in **Figure 4.26**. Doing this makes it easier to watch the values as ou navigate the tree.

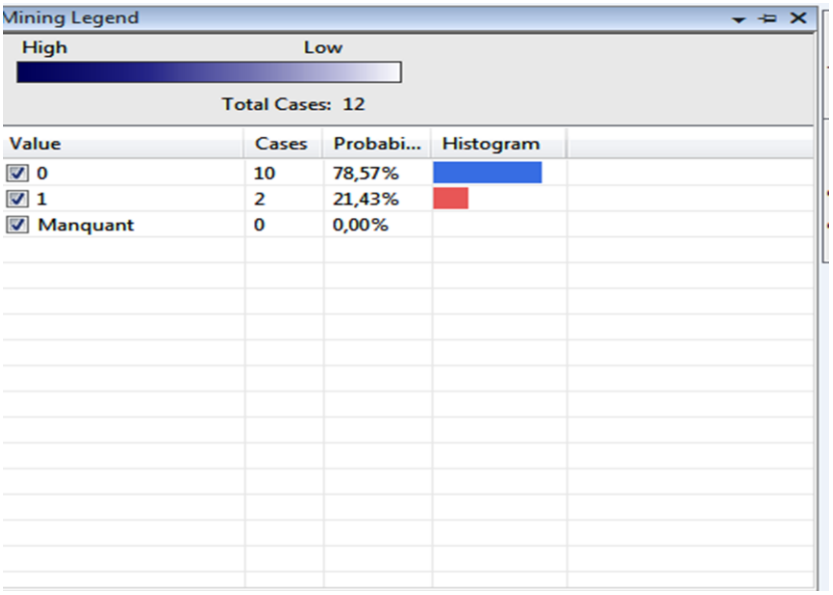

Figure 4.26 The Mining Legend, docked below the Solution Explorer

The Mining Legend has four columns of information about your model. The first column, Value, shows 0, 1, and Missing. 0 is our false, or Fail Students case, while 1 is our true, or pass Student case. The Missing value of 0 is good to see, as it indicates that our data is clean and fully populated. Our Cases column displays the population of each case, and the Probability calculates this distribution as a percentage. Finally, the Histogram column mimics the node's histogram.

The Mining Accuracy Chart View

The Mining Accuracy Chart tab is where you will validate your mining models. For your Decision Tree model, and Naïve Bayes you will be using the Input Selection and Lift Chart tabs. In the Input Selection tab's Predictable Column Name column, select Final Grade. The lift chart, shown in **Figure 4.27**, will help you determine the value of your model. The lift chart evaluates and compares your targets by using an *ideal line* and a *random line*. The percentage amount difference between your *lift line* and your random line is referred to as *lift* in the marketing world.

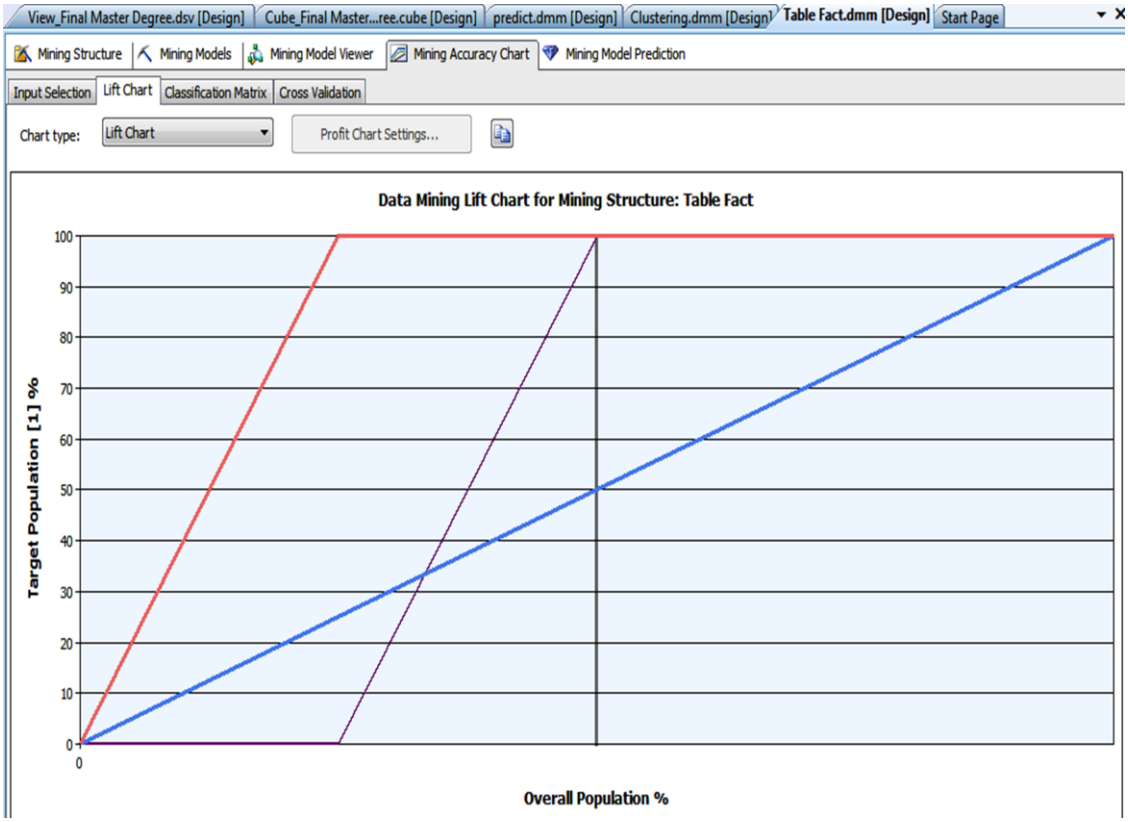

Figure 4.27 The Final Grade lift chart

The Mining Model Prediction View

Now we create a prediction. Selecting the Mining Model Prediction tab displays the Prediction Query Builder in Design view. Initially, there will be no selections in the Select Input Tables(s) area, or in the Source/Field grid at the bottom of the designer.

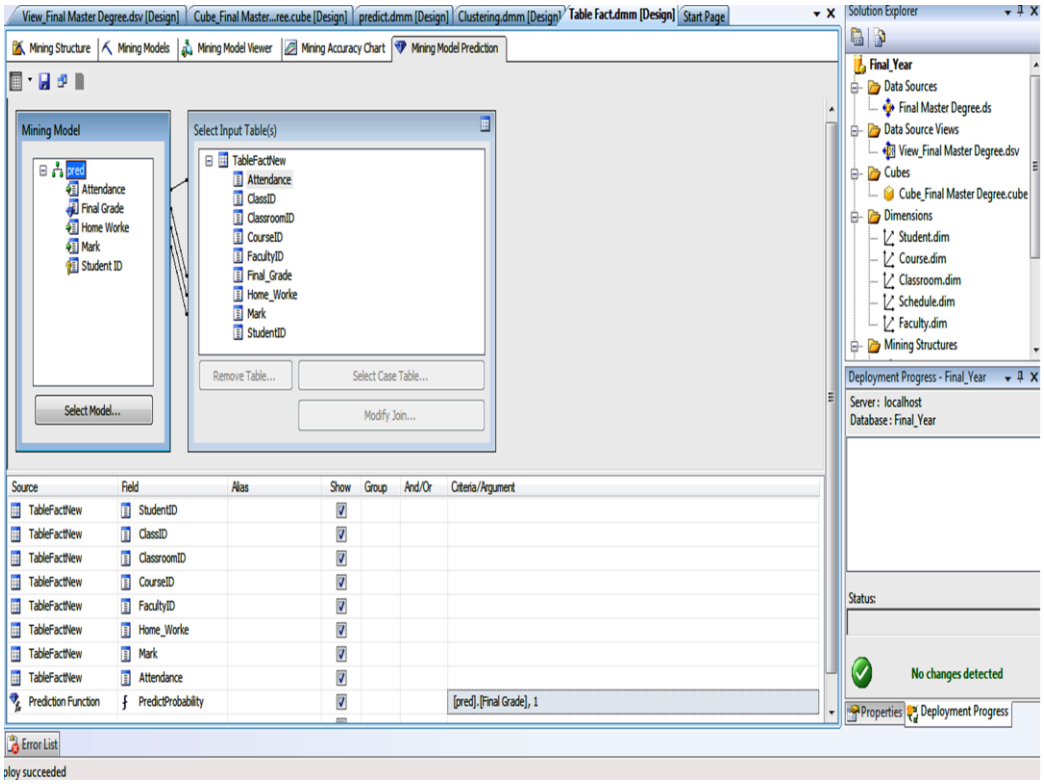

Figure 4.28 The completed Mining Model Prediction view

It is now time to run the prediction and view our results. Click the Switch to Query Result View (leftmost) button in this tab's toolbar. Your prediction query will process, and you will be presented with a grid, shown in **Figure 4.29**, containing Students targets.

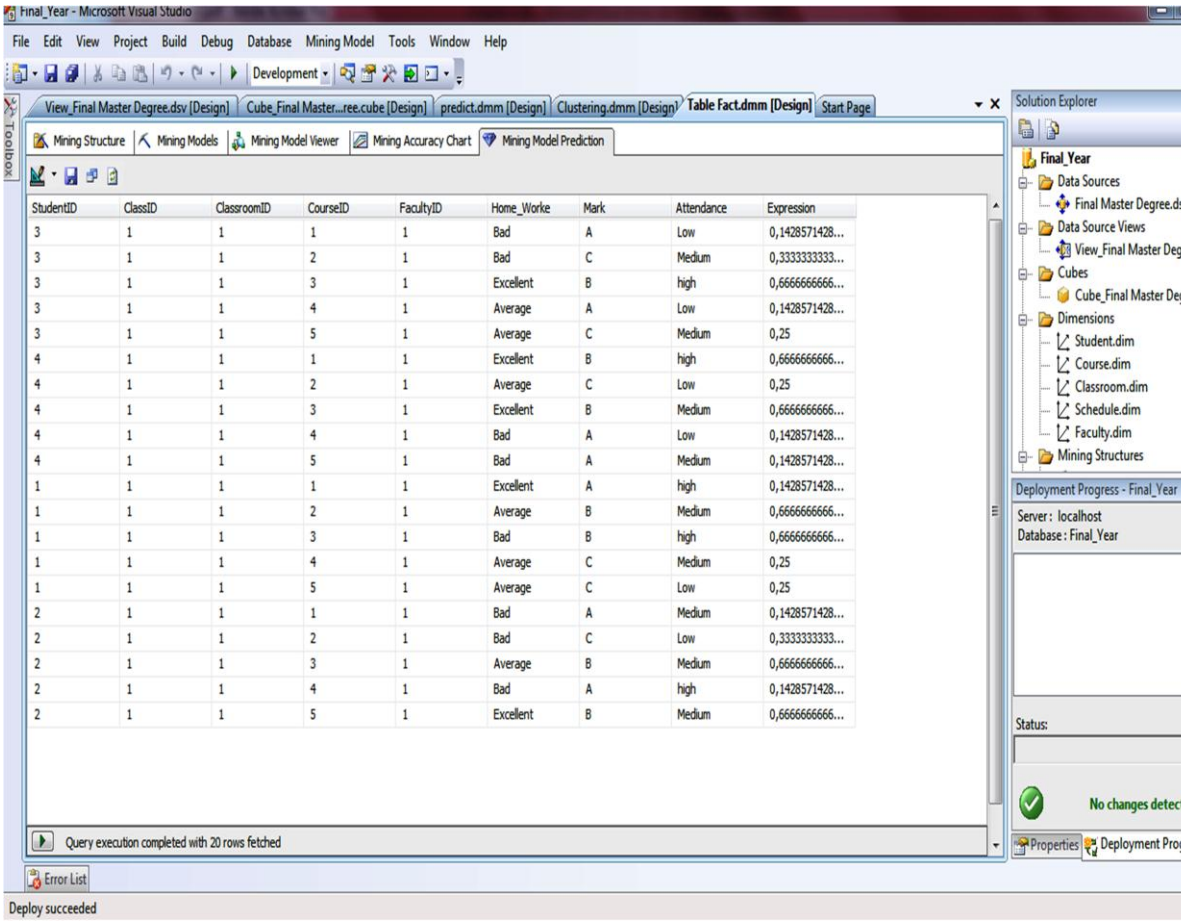

Figure 4.29 future Students Final Grade!

These targets can be saved to your database by clicking the Save Query Result button (second from the left), which is also in this tab's toolbar. In the Save Data Mining Query Result dialog box,