MANIPAL INSTITUTE
OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF INFORMATION & COMMUNICATION TECHNOLOGY

CERTIFICATE

This is to certify that Mr/Ms.……

Reg. No. …..……..…… Section …… Roll No. ……..…… has satisfactorily completed the lab exercises prescribed for Data Warehousing and Data Mining Lab [ICT 3212] of Third Year B. Tech. Degree at MIT, Manipal, in the academic year 2017-2018.

Date: …….....

Signature of the faculty

Signature
Head of the Department

CONTENTS

Course Objectives

To implement different data mining algorithms.
To gain practical key skills in using data warehousing tools.
To obtain hand-on experience of applying data mining solutions to data sets.
To motivate the students to publish their work.

Course Outcomes

At the end of this course, students will be able to
Able to implement the Data Warehousing and Data Mining concepts using IBM's InfoSphere warehouse software.
Able to identify and apply the suitable data mining technique on preprocessed data.
Improves analytic skills and problem solving skills.
Able to publish technical papers.

Evaluation plan

Total marks: 100 (Regular Lab Evaluation + End semester Lab Exam)

Split up of 60 marks for Regular Lab Evaluation
Total of 6 regular evaluations which will be carried out in alternate weeks. Each evaluation is for 10 marks of which will have the following split up: Experiment completion with output recording: 6 Marks Viva/Quiz: 4 Marks Total Internal Marks: 6 * 10 = 60 Marks
End Semester Lab Evaluation: 40 marks
Write-up and Execution: 20 Marks Project: 20 Marks Total = 40 Marks

INSTRUCTIONS TO THE STUDENTS

Pre-Lab Session Instructions

1. Students should carry the Lab Manual Book and the required stationery to every lab session
2. Be on time and follow the institution dress code
3. Must Sign in the log register provided
4. Make sure to occupy the allotted seat and answer the attendance
5. Adhere to the rules and maintain the decorum

In-Lab Session Instructions

1. Follow the instructions on the allotted exercises
2. Show the design to the instructors on completion of experiments
3. On receiving approval from the instructor, copy the design in the Lab record
4. Prescribed textbooks and class notes can be kept ready for reference if required

General Instructions for the exercises in Lab

1. Implement the given exercise individually and not in a group.
2. The programs should meet the following criteria:
   a. Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
   b. Programs should perform input validation (Data type, range error, etc.) and give appropriate error messages and suggest corrective actions.
   c. Comments should be used to give the statement of the problem and every function should indicate the purpose of the function, inputs and outputs.
   d. Statements within the program should be properly indented.
   e. Use meaningful names for variables and functions.
   f. Make use of constants and type definitions wherever needed.
3. Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
4. The exercises for each week are divided under three sets:
   a. Solved exercise
   b. Lab exercises - to be completed during lab hours
   c. Additional Exercises - to be completed outside the lab or in the lab to enhance the skill

5. In case a student misses a lab class, he/she must ensure that the experiment is completed during the repetition class with the permission of the faculty concerned but credit will be given only to one day's experiment(s).
6. Students missing out lab on genuine reasons like conference, sport or activities assigned by the department or institute will have to take **prior permission** from the HOD to attend **additional lab** (in other batch) and complete it **before** the student goes on leave. The student could be awarded marks for the write up for that day provided he submits it during the **immediate** next lab.
7. Students who fall sick should get permission from the HOD for evaluating the lab records. However, the attendance will not be given for that lab.
8. Students will be evaluated only by the faculty with whom they are registered even though they carry out additional experiment in other batch.
9. Presence of the student during the lab end semester exams is mandatory even if the student assumes he has scored enough to pass the examination.
10. Minimum attendance of 75% is mandatory to write the final exam.
11. If the student loses his book, he/she will have to rewrite all the lab details in the lab record.
12. Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and/or combinations of the questions.
13. A sample note preparation is given as a model for observation.

THE STUDENTS SHOULD NOT

1. Bring mobile phones or any other electronic gadgets to the lab.
2. Leave the lab without permission.

Database management system

The database management system is the foundation for the DB2 InfoSphere Warehouse. DB2 offers industry leading performance, scale, and reliability on your choice of platform from Linux to z/OS. It is optimized to deliver industry-leading performance across multiple workloads, while lowering administration, storage, development, and server costs.

DB2 has been available from IBM for quite a number of years, and has proven to be a highly functioning and reliable database management system. It is a leading database management system offering and is in use world-wide.

Data movement and transformation

One of the key operating principles of the IBM InfoSphere Warehouse is to maintain simplicity. More importantly, the principle is to have integrated tooling to design, develop, and deploy data warehouse applications. In the information environment, we typically find that organizations have several

DBAs that spend a significant amount of time writing SQL scripts to maintain data in the data warehouse and data marts. They must create documentation for those SQL scripts, and keep it up to date and maintained over time. This is not only time consuming for them, but also for new staff to perform data warehouse maintenance. With InfoSphere Warehouse, the data movement and transformation for intra-warehouse data movement is accomplished using the SQL Warehousing tool (SQW).

There are two basic types of flows in SQW:

1. Data flow: A data flow moves and transforms data using SQL based operators to accomplish tasks (such as joins, filtering, splitting data, transforming data with column functions, and maintaining slowly changing dimensions).
2. Control flow: A control flow arranges data flows and other types of data processing into logical sequences of processing. A control flow can invoke OS scripts, SQL scripts, and ftp files, in addition to data flows. Control flows are created using SQW are deployed and managed using the InfoSphere Warehouse Administration Console. The administration console allows for variables to be introduced in transformation jobs that can be modified at deployment or run time.

Modeling and design

All the modeling and design for the InfoSphere Warehouse is done using an Eclipse-based integrated development environment (IDE) interface called Design Studio. This IDE allows for design and development of data models, OLAP models, data mining models, and intra-warehouse data movement tasks.

Design Studio includes the following tools and features:

1. Integrated physical data modeling, based on InfoSphere Data Architect
2. SQL Warehousing Tool for data flow and control flow design including integration points with InfoSphere and DataStage ETL systems
3. Data visualization tools for data mining and data exploration
4. Tools for designing OLAP metadata, MQTs, and cube models

Each tool in InfoSphere Warehouse Design Studio has the same look, feel, and mode of operation.

Physical data modeling

The physical modeling component is the subset of the InfoSphere Data Architect tool used to develop and maintain physical data models. Application developers and data warehouse architects use this component to work with physical data models for source and target databases and staging tables.

A physical data model is essentially the metadata representing the physical characteristics of a database. The data model can be newly developed using the data model graphical editor or can be reverse-engineered from an existing database. From the physical data model, you can perform the following tasks:

1. Create the physical objects in a database
2. Compare a data model to the database (and generate just the delta changes)
3. Analyze the model for errors such as naming violations
4. Perform impact analysis

The physical model is also used to provide metadata information to the other functional components of InfoSphere Warehouse, such as the SQL Warehousing Tool. Figure 1.1 shows a physical data model open in the Data Project Explorer, a graphical editor view of the physical data model, and the properties view of a selected table.

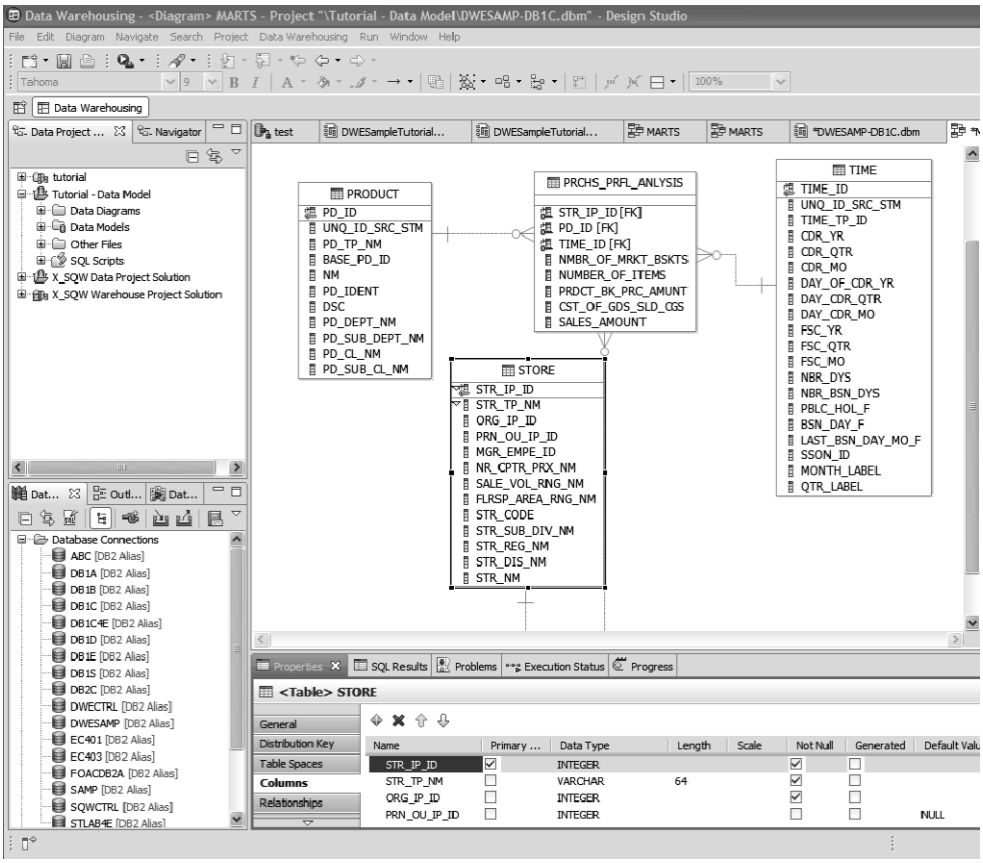


Figure 1.1 InfoSphere Warehouse physical data model

InfoSphere Warehouse Topology

InfoSphere Warehouse components are grouped into four categories for the purpose of installation:

1. Clients

The client category includes the InfoSphere Warehouse administration client, InfoSphere Warehouse Design Studio, the workload manager and the data mining visualization programs. Clients are supported on Windows 32-bit and Linux systems.

2. Database servers

This category includes the DB2 database server with DPF support plus the database functions to support InfoSphere Warehouse Intelligent Miner, InfoSphere Warehouse Cubing Services and the workload manager. The database server is supported on AIX®, various Linux platforms, Windows Server 2003 and System z.

3. Application servers

The application server category includes the WebSphere Application Server, DB2 Alphablox server components and InfoSphere Warehouse Administration Console server components.

4. Documentation

This category includes the pdf and online versions of the manuals and can be installed with any of the other categories.

LAB NO.: 1 **Date:**

DBMS BASICS, DEVELOPING PHYSICAL DATA MODEL USING DATA WAREHOUSE TOOLS

Objectives:

In this lab, student will be able to

- ➢ Revise the DBMS schemas and queries
- ➢ Understand physical data creation using warehousing tools.

Introduction

SQL (Structured Query Language) is a standardized programming language, used for managing relational database, and to perform various operations over the data kept in the database. SQL is generally used by Database administrators and developers for writing data integration scripts, to run analytical queries.

SQL Queries are divided into many types, in which DML (Data Manipulation Language) and DDL (Data Definition Language) are the most common.

DDL (Data Definition Language)

DDL is used to create Schema's which defines and modifies, the structure of tables and Databases which contain the tables.

CREATE

- ➢ Used to create both Database and Tables.

Syntax:

create Table Tablename(attribute_name type(size));

Example:

create database college;

create table instructor(id int, name varchar, dept_name varchar, salary int);

ALTER

Used to alter table structures, they variantly used to carry out functions like,

➢ to add a column to existing table.

➢ to rename any existing column

➢ to change datatype of any column or to modify its size.

➢ *alter* is also used to drop a column.

Syntax:

alter table tablename **modify**(attribute_anme type(size));

Example:

alter table instructor **modify**(phoneNumber varchar(20));

DROP and TRUNCATE

ID	Name	Dept_Name	Salary
10101	Srinivasan	CSE	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	60000
58583	Kim	ECE	80000
76543	Brandit	CSE	92000
83821	Katz	CSE	75000
98345	Crick	History	62000

Table: Instructor

Syntax:

drop table tablename**;**

Example:

drop table instructor;

➢ Deletes table ‘instructor’ from respective database.

Syntax:

truncate table tablename;

Example:
truncate table instructor;

➢ Empties the table, and preserves its structure, enabling further data insertion.

DML (Data Manipulation Language)

The DML vocabulary is used to retrieve and manipulate data, i.e. for selecting, inserting and updating data in a database.

Insert

➢ Command to insert a row into the table.

Syntax:

insert into table (column1 [, column2, column3 ...]) values (value1 [, value2, value3 ...])

Dept_Name	Building	Budget
CSE	Watson	CSE
ECE	Taylor	Finance
Finance	Painter	Music
History	Painter	Physics
Music	Packard	ECE
Physics	Watson	CSE

Table: Department

Example:

insert into instructor (id,NAME,dept_name,salary) values (1,'John','CSE',14000)

Select

➢ Command to select / fetch a row or a set of rows from a table. Select statement has many optional clauses like,

WHERE

- Specifies which rows to retrieve.

GROUP BY

- Group rows sharing a property so that an aggregate function can be applied to each group.

HAVING

- Selects among the groups defined by the GROUP BY clause.

ORDER BY

- Specifies an order in which to return the rows.

AS

- Provides an alias which can be used to temporarily rename tables or columns.

Examples:

Select **NAME** from **instructor**;

Returns the all **NAME** in the table '**instructor**'.

Select **NAME** from **instructor** where id = 1500;

Returns the NAME of instructor, which id, 1500.

Select id,NAME,dept_name,salary from **instructor** where salary > 10000 **ORDER BY** dept_name;

Return information listed, having above salary above 10000, order by dept_name.

Select dept_name from instructor **GROUP BY** dept_name **HAVING SUM(**salary**)** > 100000;

List and orders dept_name, which spends more 100000 as Salary.

Update

➢ Command to update a row or a set of rows in a table.

Syntax:

update *table_name* **set** *column_name* = *value* [, *column_name* = *value* ...] [**where** *condition*]

Example:

update instructor set NAME = "ALBERT" where id = 110;

Updates field 'NAME' where field 'id' has value 110.

Delete

➢ Command to delete a row or a set of rows in a table.

Syntax:

delete from table CONDITION;

Example:

delete from INSTRUCTOR where ID = 110;

Deletes record, having ID filed value 110.

Nested Statements

SQL has an ability to nest queries within one another. A sub-query is a SELECT statement that is nested within another SELECT statement and which return intermediate results. SQL executes innermost sub-query first, then next level.

Nested Statements come handy whenever the data, is spread across tables and we need to refer data from other table to fetch a specific data from current table.

Example:

select name **from** instructor **where** dept_name = (select dept_name **from** department **where** building = 'painter');

In the above example, the names of all instructors working inside 'painter' building is listed. The nested query inside the '(..)' returns the dept_name which works inside 'painter' building, and later the parent query is ran.

Aggregate Functions

Built-in functions to perform operations over data, are called SQL Aggregate **functions**. SQL includes a wide set of Aggregate Function and sum of the most common are shown below:

Function	**Definition**
Avg()	Returns the average value
Count()	Number of rows
First()	First value
Sum()	Returns the sum
Max()	Returns the maximum value or largest.
Min()	Returns the minimum or smallest of a function.

Examples:

- **select AVG(***salary***) from** *instructor*;

  Returns the average of Salaries of all instructors.

- **Select count**(*) **from** instructor

  Returns the number of results, for the query.

- **Select MAX(**salary**) from** instructor;

  Returns the maximum salary in the instructor table.

Introduction to Design Studio

The Workbench is the interface you use to access and use the capabilities of Design Studio.

The workspace

Every time InfoSphere Warehouse Design Studio is launched, you are prompted to provide a path to the workspace, as shown in Figure 2.1. A workspace is a collection of resources and is the central repository for your data files.

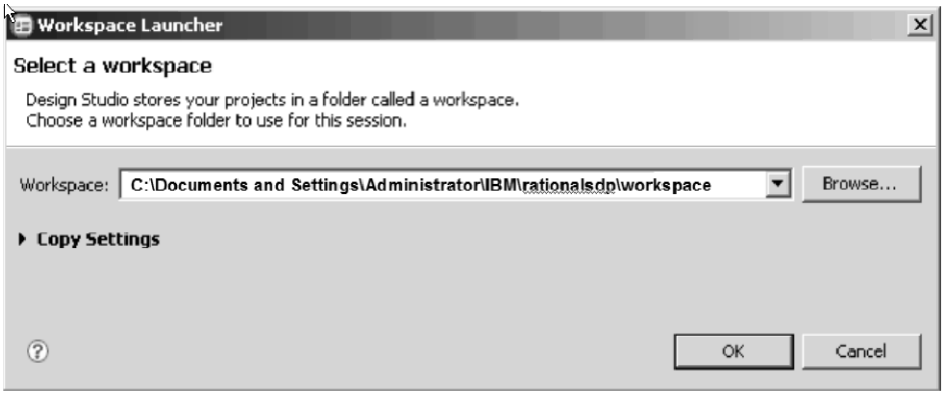


Figure 2.1 Prompt for the workspace location

A project is a container used to organize resources pertaining to a specific subject area. The workspace resources are displayed in a tree structure with projects containing folders and files being at the highest level.

You can specify different workspaces for different projects, but only one workspace is active per running instance of Design Studio. To change workspaces, choose **File->Switch Workspace**. A workspace can hold multiple projects.

Projects and the local file system

When you create a new project, you find a new subdirectory on disk, located under the workspace directory specified at start up. In the project directory is a special file with a .project extension. The .project file holds metadata about the project, including information that can be viewed by selecting the Properties view in Design Studio. Inside the project subdirectory you see all of the files and folders that have been created as part of the project.

The Design Studio Workbench

The term *workbench* refers to the desktop development environment, and delivers the mechanism for navigating the functions provided by the various Design Studio plug-ins. The workbench offers one or more windows, which contain one or more perspectives, views, and editors that allow you to manipulate resources in your project. The default workbench for InfoSphere Warehouse Design Studio contains the following elements:

Perspectives

Perspectives define an initial layout of views and editors in a workbench window. Perspectives provide the functionality required to accomplish a particular task or work with a particular resource. They can be customized or modified and saved for reuse by selecting **Window->Save Perspective As**. If you have rearranged views or closed views, you can reset the perspective by choosing **Window->Reset Perspective**.

Editors

In Design Studio, there are different editors available for different types of files. Text and SQL editors are provided for resources such as SQL scripts, and diagram editors are available for resources such as data models. An editor is a visual component that you use to edit or browse a resource.

Views

A view is a component that you use to navigate a hierarchy of information, open an editor, or display properties for the active editor. Modifications that you make in a view are saved immediately.

Data Project Explorer view:

By default, this view opens in the upper left area of the Design Studio window. The Data Project Explorer in Figure 2.2 shows a logical representation of the currently opened projects. The representation is logical because the name of the folders and resources represented here need not match the real files and directories on the file system.

Navigator view:

The Navigator shows a physical representation of the files and directories of the opened projects on the file system.

Data Source Explorer view:

By default, the Data Source Explorer opens in the lower left area of Design Studio. This view enables you to connect to and explore a database. You can make the changes to the database that you have the proper authority and privileges to complete.

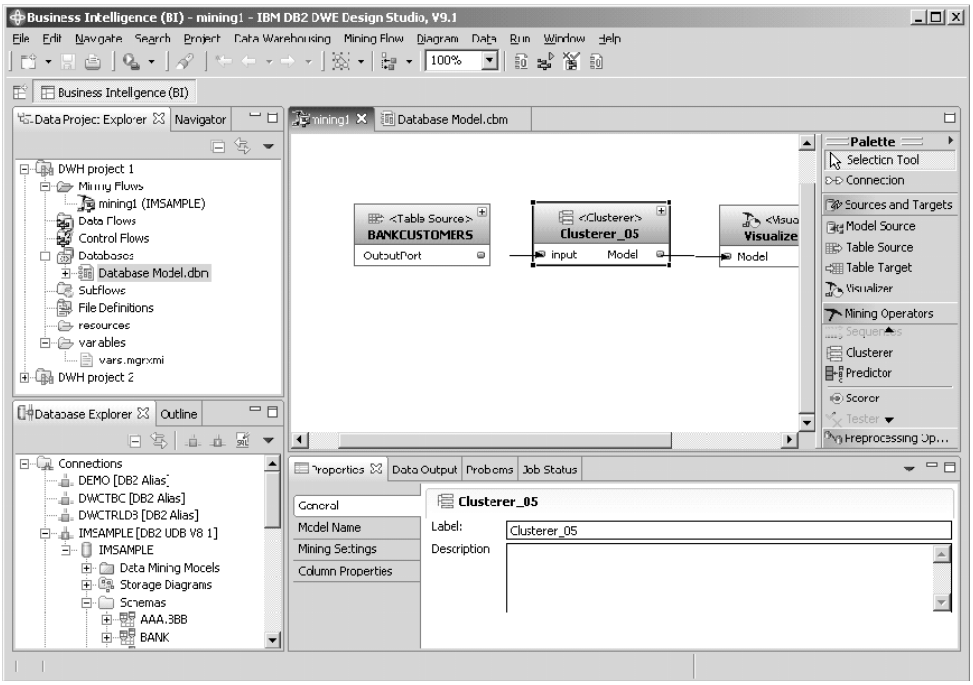


Figure 2.2 Data Project Explorer view

The Data Source Explorer view is used to complete the following tasks:

1. Create JDBC connections to databases.
2. Explore database content including schemata, table relationships and content, and value distributions.
3. Generate storage overview diagrams from databases, and overview diagrams of schemas using either Information Engineering (IE) notation or Unified Modeling Language (UML) notation. Diagrams are a helpful way to visualize data warehousing projects.
4. Reverse engineer databases.
5. Compare database objects.

6. Analyze a database or a schema to ensure that it meets certain specifications.

   Model analysis helps to ensure model integrity and helps to improve model quality by providing design suggestions and best practices.
7. Analyze the impact of changes to models.

   You can also use the Impact Analysis features to find dependencies. For example, if you want to copy a schema from the Data Source Explorer to the Data Project Explorer, you can find dependencies on the schema to ensure that all references are resolved. You can analyze data objects in the Data Source Explorer, the Data Project Explorer, or in the data diagram.
8. Create new database objects.
9. Drag database objects to the Data Project Explorer.

Outline view:

The Outline view shows an overview of the structural elements of the file that is currently open for editing. The contents of this view depend upon the nature of the edited file.

Properties view:

The Properties view, shown in Figure 2.3, is taken from the Outline View that is stacked with the Database Explorer View. It allows you to view and modify the properties of the current selected object. The edit capabilities of this view are dependent on the type of object that is currently selected.

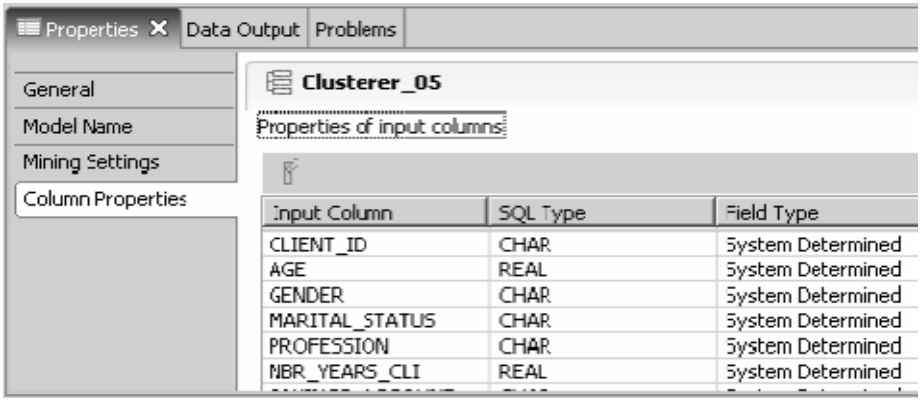


Figure 2.3 The Properties view in InfoSphere Warehouse Design Studio

SQL Results view:

The SQL Results view is used to see messages, parameters, and results of the objects that you are working with. The SQL Results view displays the results of various actions when they are executed on a database. Use this view to inspect the contents of a table

through the Sample Contents feature, execute a data mining flow, or execute an SQL or DDL script or perform any operation on a database. The SQL Results view is stacked with the Properties view.

The SQL Results view is divided into two parts. The left part of the view contains a read-only table with four columns:

- Status (indicates the state of the associated action)
- Operation (indicates what kind of action occurred)
- The data of the action
- The database connection of the associated action

The right side of the Data Output view contains two tabs, as depicted in Figure 2.4:

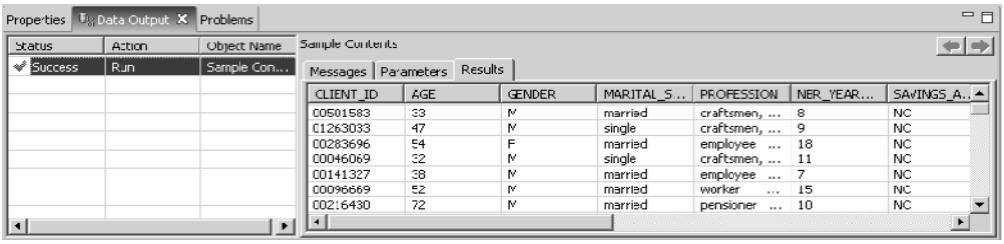


Figure 2.4 The SQL Results view

- Status

  This tab shows messages generated while the statement was being run. If there are errors in the SQL statement, an error message appears on this page.

  For INSERT, UPDATE, and DELETE statements, a message is displayed on this page if the statement runs successfully. If an INSERT, UPDATE, or DELETE statement runs successfully, the database is modified.

- Results

  This tab contains any results that were generated from running the statement (for example, a table of sales data). The Results tab is selected by default.

Problems view:

While working with the resources in your data warehousing projects, certain editors might log problems, errors, or warnings for the Problems view. The Problems view is stacked with the Properties view and the SQL Results view. The Problems view shows three levels of severity:

- Errors
- Warnings
- Information

Creating projects

Data warehouse projects are containers for the development of data warehouse applications in Design Studio. Before you can use Design Studio to perform any of the processes in the BI Solutions Development life cycle, you must create a project. The data warehouse project contains artifacts (such as your data schemas, SQL control flows, and mining flows).

To create a project, from the main menu select **File->New->Project->Data Warehousing**. There are two types of projects provided by Design Studio:

- Data design projects (OLAP)
  Design database physical models, including OLAP models. You can engineer or reverse-engineer database models.
- Data warehousing projects
  Design SQL Warehouse and data mining flows. This project type also provides the mechanisms to generate data warehouse application packages to deploy on the Information Warehouse server. Data warehouse projects can also contain your physical data models.

Adding database connections

Every time Design Studio is launched, the Data Source Explorer displays all local DB2 databases, any remote DB2 databases that have been cataloged on the IW client machine, and any previously defined non-DB2 database connections. Before you can browse or explore a new database, you must define a database connection to the data source.

The database connections are represented in the Data Source Explorer view by a node. The connections are initially disconnected (no [+] sign on their left side). You can connect to a database by right-clicking its connection node and choosing **Connect**. If the user ID and password have not been saved in this connection profile, you are prompted for a user ID and password. After the connection has been made, you can expand the tree to explore its schemata, tables, and other objects.

If you want to add another remote database or non-DB2 database to your Database Explorer view, right-click the **Connections** node and select **New Connection**. A dialog box collects the connection information (such as database vendor and version, database name, port number, JDBC driver class location, and user ID and password). You can also specify filters for the database connection if it is appropriate to do so. After this information is supplied, the database is added to the Connections list.

Editing, loading, and extracting data

To get to the Data context menu, expand a database tree in the Data Source Explorer view, select a table, right-click, and chose **Data**. The following actions are available from the context menu:

- Edit
  With the proper database authority, edit table contents in an editor.
- Load
  Load the table with data from a flat file.
- Extract
  Write table contents to a flat file through the Extract option.
- Extract as XML
  Write table contents to an MXML file.
- Load from XML
  Load the table with data from an XML file.

Working with databases offline

Database connection information can be saved locally to allow database objects to be viewed from Design Studio without having an active connection to the database. Not all features are available without an active connection. Capabilities that can be performed include viewing database objects and their properties and creating and viewing overview diagrams.

Modeling data structures

As part of the process of developing your BI solution, you design and modify physical database models and define schemas and storage specifications. To get started, open the data design project in which you want to work, and launch the New Data Model Wizard by right-clicking the Data Models folder and choosing **New□Physical Data Model**. Choose the destination project folder, provide a name for your data model, and continue to follow the prompts from the wizard. Models can be created from scratch or reverse engineered from existing DDL or databases. Other tasks that are performed as part of this phase in the development life cycle are as follows:

- Designing table spaces and containers for DB2 databases.
- Comparing data objects and models with each other or with other objects.
- Analyzing designs to confirm that standards are being met.
- Deploying the model.

Lab Exercises

1. Develop a physical model using reverse engineering and get familiarize the IBM Infosphere warehouse software.
2. Consider the following database of student enrollment in courses & books adopted for each course. Develop a physical model and also create the overall view for the below schema.

   Department (deptid, dname, location)
   Student (snum, sname, deptid, slevel, age)
   Faculty (fid, fname, deptid)
   Class (cname, time, room, fid)
   Enrolled (snum, cname, grade)

   Write the SQL query to:
   a) Get student numbers and names of those students who have an 'A' in all the courses they are enrolled in)
   b) Get the names of faculty teaching a class in room 'AB5-309'. If the same faculty teaches for more than one class in the same room, only one entry should be listed.

3. Consider the employee database given below where the primary keys are underlined, develop the physical model for employee database. Write SQL queries for the following questions.

   Employee (EmpID,name, street, city)
   Works (EmpID, CompID, company-name, salary)
   Company (CompID, city)
   Manages (EmpID, ManagerID)
   a) Find the names of all employees who work for First Bank Corporation.
   b) Find the names and cities of residence of all employees who work for First Bank Corporation.
   c) Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000 per annum.
   d) Find all employees who live in the same city as the company for which they work is located.

[OBSERVATION SPACE – LAB1]

LAB NO.: 2 **Date:**

DEVELOPING DATA FLOW FOR SIMPLE QUERIES USING DATA WAREHOUSE TOOLS

Objectives:

In this lab students should be able to:

➢ Understand execution of simple queries using warehouse tools.

Development environment

A physical data model of the source and target relational databases are required to develop data movement routines. The physical data model provides metadata about the structure of the source and target tables.

Figure 2.1 shows the structure of a data warehouse project. The project is organized into folders that contain the various types of development artifacts. The following folders are used most, as these contain the primary development artifacts:

1) Data Flows
2) Control Flows
3) Subprocesses
4) Subflows

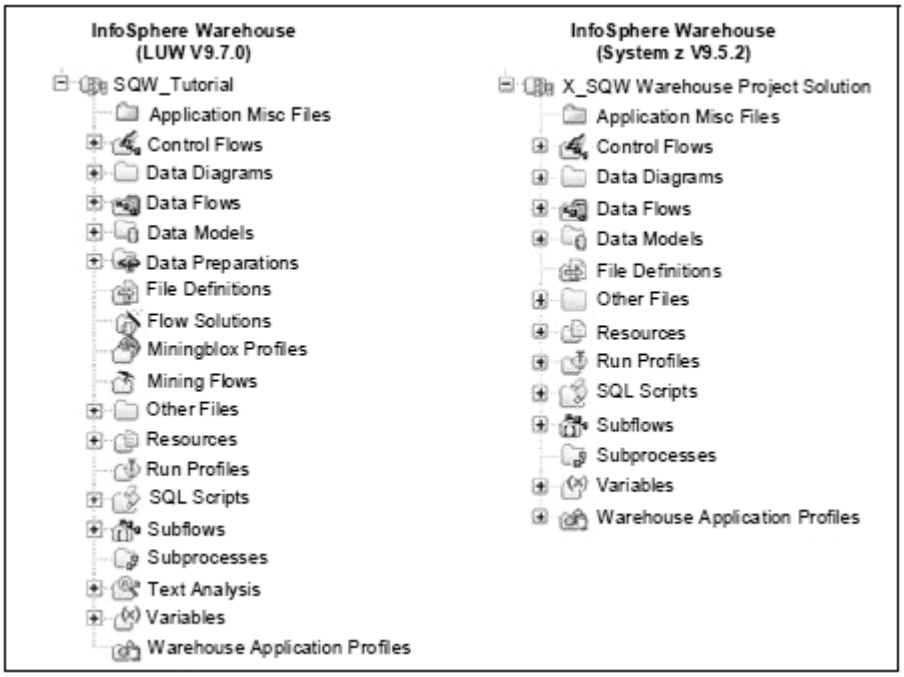


Figure 2.1 Data warehouse project folders

Data flows and subflows

Data flows model the SQL-based data movement and transformation activities that are executed by the DB2 database engine. A data flow consists of activities represented by graphical operators that extract data from flat files or relational tables, transform the data, and load it into a relational table in a data warehouse, data mart, or staging area. Data can also be exported to flat files. A sample data flow is shown in Figure 2.2.

Design Studio provides a graphical data flow editor with an intuitive way to visualize and design data flows. Graphical operators model the various steps of a data flow activity. By arranging these source, transform, and target operators in a canvas work area, connecting them, and defining their properties, models can be created that meet the business requirements. After creating data flows, SQL code is generated which perform the specific DB2 SQL operations needed to complete the operations defined by the data flow model.

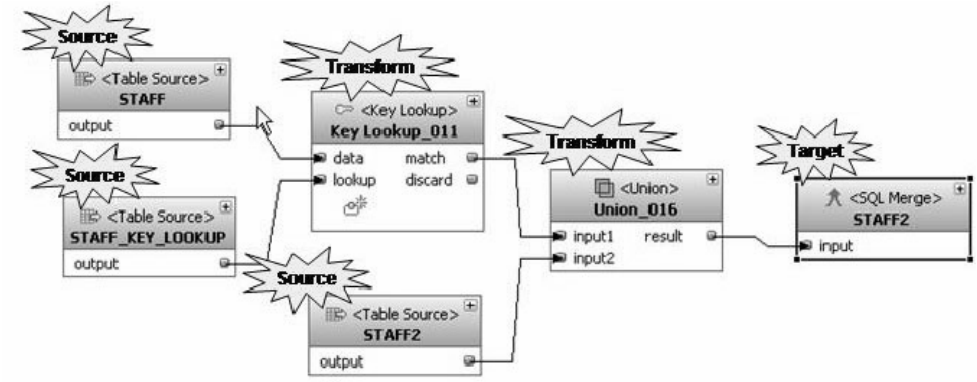


Figure 2.2 A simple data flow

Subflows are like data flows but are used like macros or subroutines.
Subflows are included in a data flow by using a subflow operator. Subflows have special input and output connections that pass the data into and out of the subflow. Subflows can be used to define common data flow components that might be used in multiple data flows, or just to break a complex data flow into multiple logical sets of operations (like using structured techniques in a programming language).

Developing data flows

A data flow is essentially the set of operations that performs the following tasks:

- ➢ Bring data into the data warehouse
- ➢ Move and transform the data in the data warehouse
- ➢ Update target tables

Data flows and subflows are developed using the Design Studio graphical data flow editor to create a flow model representing the data movement and transformations of the data. From this model, Design Studio generates optimized SQL to perform the actions specified in the flow model. The SQL is executed at the transformation DB2 database selected as the execution database.

Data flow basics

As a data model is a representation of a database, a data flow is a representation of what is to happen to the data as it flows into and in the execution database from source to target. Data flow operators define source data, transformations, and target tables. Operators are placed on the data flow editor canvas and connected to other operators using connectors. Operator properties define the details of the operator behavior.

Operators, connectors and ports

In general, there are three categories of operators:

- ➢ Sourcing data operators
- ➢ Transforming data operators
- ➢ Data targets operators

Figure 2.4 shows the operators in a simple data flow. There are a total of three operators that represent sources of data from relational tables. These are Table Source operators for the STAFF table, STAFF_KEY_LOOKUP table and the STAFF2 table. These table sources are connected to transform operators that perform actions on the data, a Key Lookup and a Union. The Union transform operator is connected to a target operator that does an SQL Merge of the incoming data into the STAFF2 table.
Operators are linked together using connectors, as shown in Figure 2.5.

Connectors represent how the data flows from one operator to another. In this example, the output from the STAFF table source operator flows into the key lookup operator, as does the output from the STAFF_KEY_LOOKUP table source operator. The output from the match port of the key lookup operator flows into the union operator, as does the output from the STAFF2 table source operator. The result of the union operator is merged into the STAFF2 table using the SQL Merge operator.

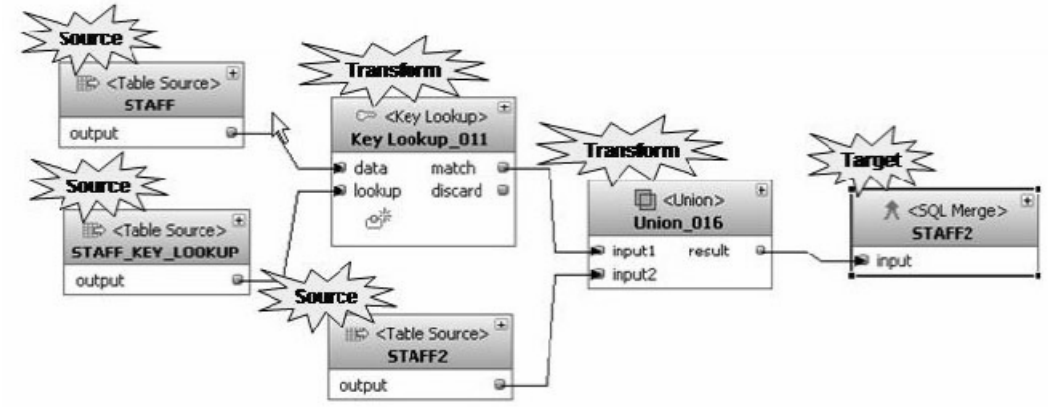


Figure 2.4 Operators in a simple data flow

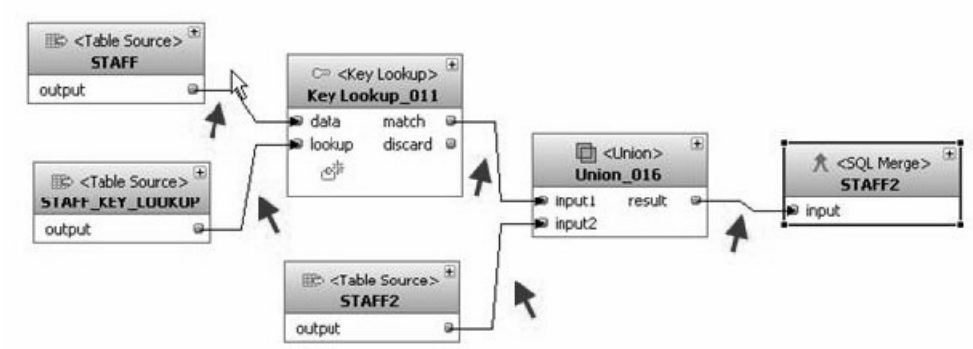


Figure 2.5 Connectors in a simple data flow

Connectors connect to the input and output ports of operators. Ports represent the virtual table definition of the data coming through that port, defining the virtual columns and the data type and size of each virtual column. The virtual table definition of output ports of source operators typically is initialized based on the column metadata of the underlying source or target operator.

The virtual table definition of output ports of other operators depends on the properties of the particular operator. The virtual port definition of input ports is typically initialized when first connected to an output port by propagating the virtual table definition from the connected output port. When initialized, the input port retains the virtual table definition even if the connector is deleted. During re-connection, you can specify if the output virtual table should be propagated to the input port and have the virtual table columns mapped by name or mapped by position. In addition, if a modification to an operator's properties changes the definition of the output port's virtual table, only the changes can be propagated.

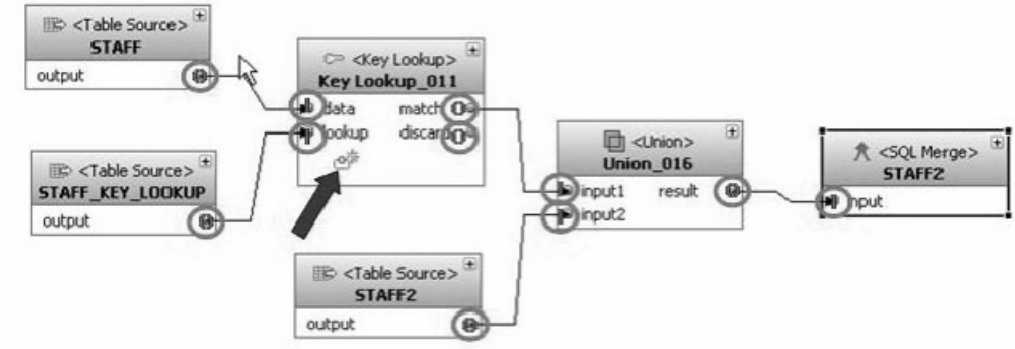


Figure 2.6 Ports in a simple data flow

Source, target, and execution databases

The execution database is the database that does the transformation work when the SQL code in a data warehouse application runs. This database must be a DB2 database appropriate to the platform of your InfoSphere Warehouse product. The execution database alias is a primary property of a data flow and all SQL code for that data flow is executed at the execution database. Different data flows can have different execution databases. The data flow SQL code is submitted to the execution database defined in the data flow properties. (For example, if you need to move data from a warehouse database to a data mart in another DB2 database or subsystem.) Part of the processing can happen at the data warehouse database and part of the processing can happen at the data mart database.

Data flow editor

In the InfoSphere Warehouse Design Studio, data flows are developed using several common Design Studio functions (such as the Data Project Explorer) views (such as Properties, Data Output, and Problems) and, optionally, the Database Explorer. However, the actual creation and editing of data flows occurs in a specific graphical editor called the data flow editor, depicted in Figure 2.7.

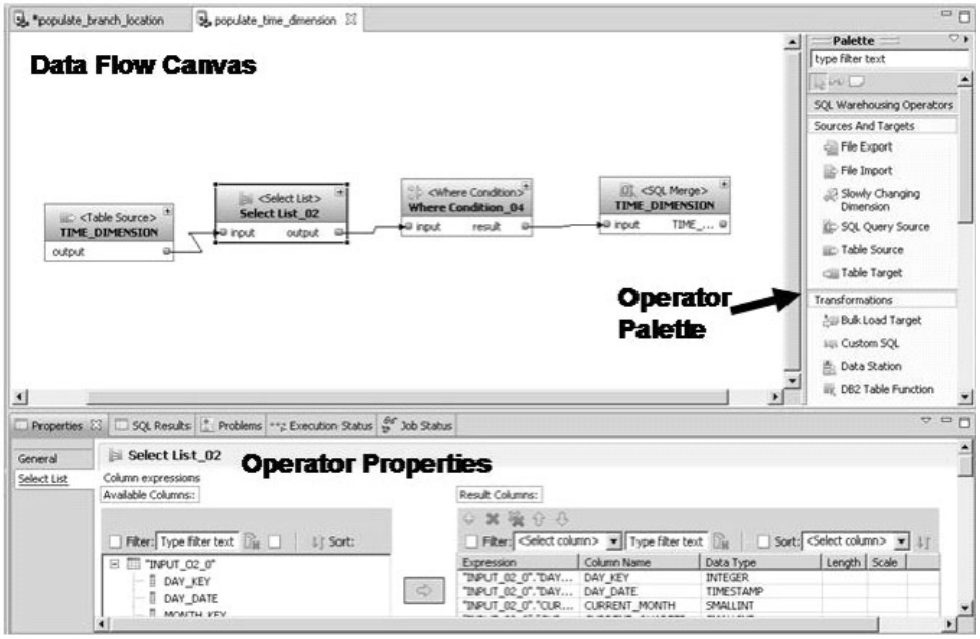


Figure 2.7 Data flow editor and properties

As with most graphical editors, the data flow editor follows the drag, drop, connect, and set properties paradigm of development. When you create a new data flow or open an existing data flow, the data flow editor opens in a new tab in the editor area of Design Studio. The data flow editor consists of a canvas (onto which graphical elements are drawn) and an operator palette (that contains all of the graphical elements that are relevant to building a data flow). In addition, you use the Properties View tab to define the characteristics of the objects on the canvas. As you validate and test run the data flows, you also use the Problems view tab and the Data Output tab.

Developing a simple data flow

Let us consider a simple example, a data flow that updates the data warehouse table, MARTS.ORDER_FACT from the source database order tables, GOSALES.ORDER_HEADER and GOSALES.ORDER_DETAILS. The data warehouse database is GSDB.

Target Column	Source	Transformation Notes
ORDER_DETAIL_CODE	GOSALES.ORDER_DETAILS.ORDER_DETAIL_CODE	
QUANTITY	GOSALES.ORDER_DETAILS.QUANTITY	
UNIT_PRICE	GOSALES.ORDER_DETAILS.UNIT_PRICE	
BRANCH_CODE	GOSALES.ORDER_HEADER.SALES_BRANCH_CODE	
DAY_KEY	GOSALES.ORDER_HEADER.ORDER_DATE	Use ORDER_DATE to lookup TIME dimension key from GOSALES.TIME_DIMENSION

Figure 2.8 Source to target mapping requirements

Creating a new data flow

To create a new data flow, open your data warehouse project (or create a new project), right-click the Data Flows folder and click New ☐Data Flow. This opens a new data flow editor similar to Figure 2.13 with an empty canvas. One task that should be done at this time is to define the database at which this data flow executes. This is called the execution database and is typically the target database.

The execution database is a property of the data flow itself. To see the properties of the data flow, click any white space on the data flow canvas, as indicated by the hand icon shown in Figure 2.9. To actually access the data flow properties use the Properties View tab, which is highlighted at the bottom of the white space with another hand icon.

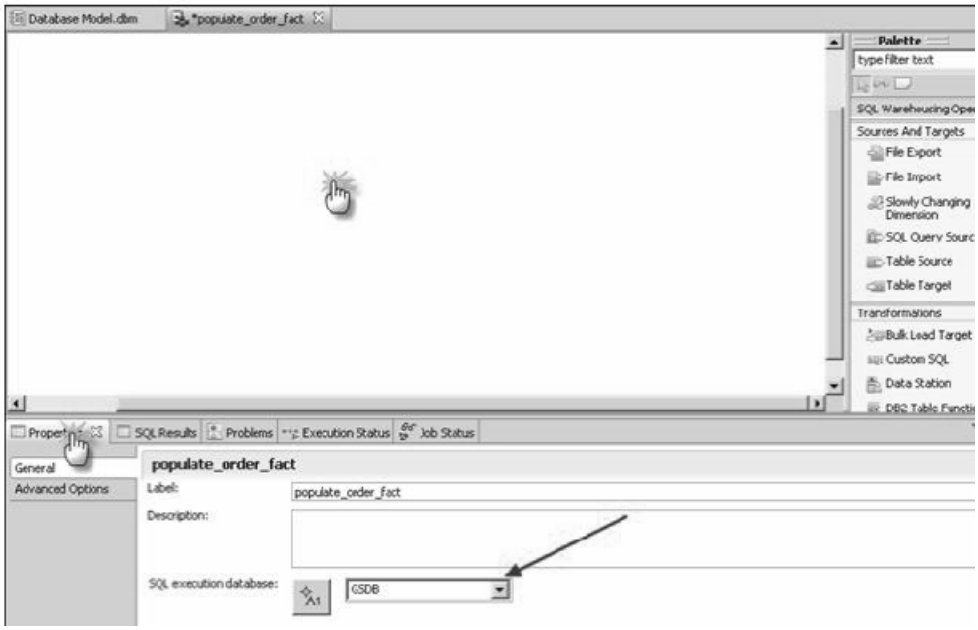


Figure 2.9 Setting the execution database

Accessing the source data

Source operators access various types of source databases and files, extract the data and bring it into the data warehouse. The primary source type is the relational table. Any relational database from which a data model can be created can be used as a source table. For our example data flow, there are two relational tables for source data, GOSALES.ORDER_HEADER, and GOSALES.ORDER_DETAILS. The table source operator is used to extract data from relational tables. The following are the steps for extracting the source data:

- From the data flow editor palette, drop a Table Source operator onto the canvas.
- Click the ellipse button that is to the right of the field to open the table picker and select the desired table from the appropriate data model. This populates the Table name, Schema name, and Source database name fields. It also uses the metadata from the data model to create the virtual table of the output port.
- Also on the General property tab is the property to specify whether the table is local or remote in relation to the execution database for the data flow. Selecting remote presents another property to specify the connection for the remote database.

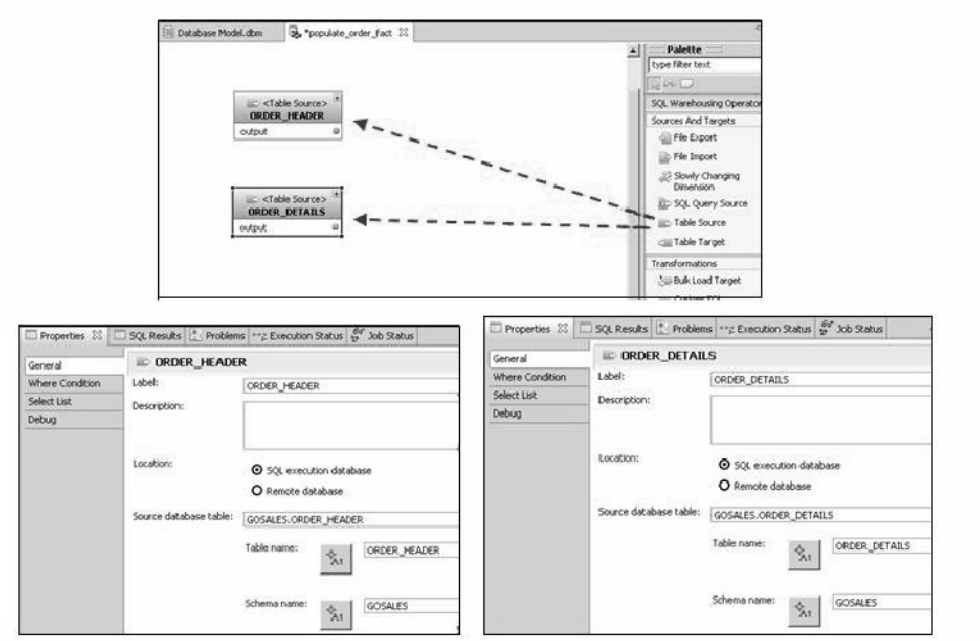


Figure 2.10 Using the table source operator

The Where Condition property is used to send a Where Condition to the remote database to filter the returned rows. This needs to be written in the SQL dialect appropriate to the source database system.

The Select List property is used to select only the needed columns and to create derived columns again to push processing down to the source database system. This is to reduce traffic over the network. As with the Where Condition property, the SQL needs to be written in the SQL dialect for the source database system, and no validation is performed.

Joining the data

The requirements map the two source tables to the one target table. This requires that we join the tables together and select only the needed columns. To do that, we use the Table Join Operator. The Table Join Operator can perform inner, left outer, right outer, and full outer joins. This is determined by using the appropriate output port of the Table Join Operator. An example of using the Table Join Operator is now presented for the scenario depicted in Figure 2.11.

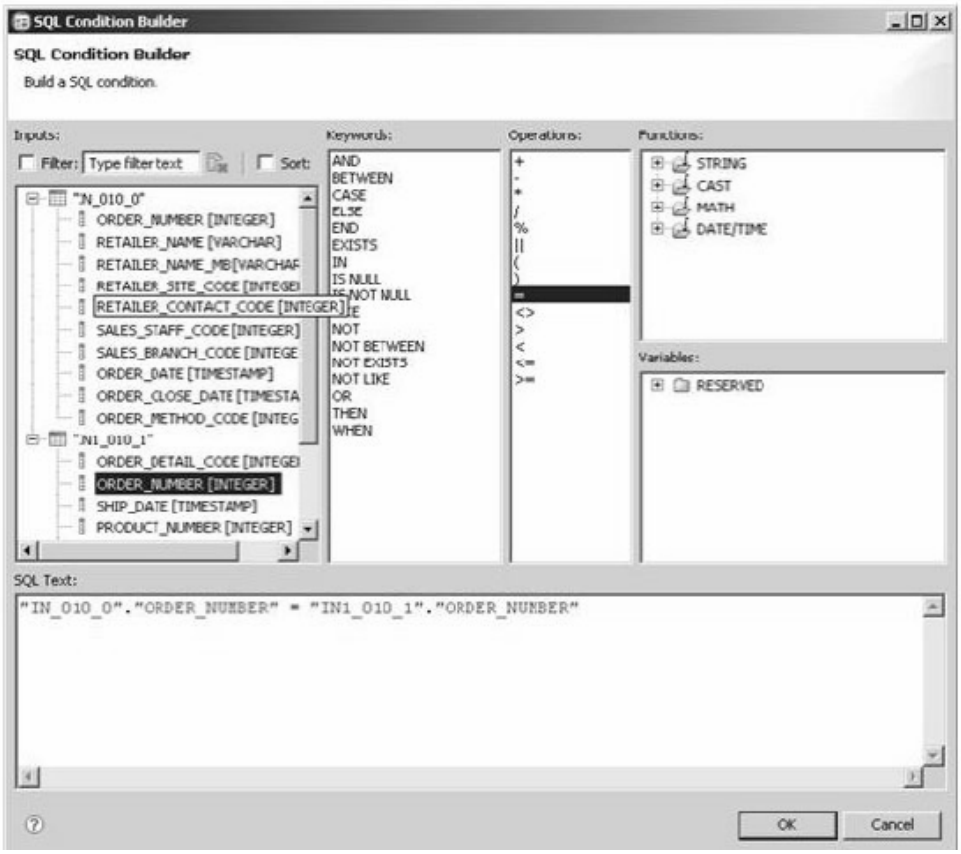


Figure 2.11 Using the table join operator

The following steps are required for the join using the Table Join Operator:

1. Select the Table Join operator from the palette and drop it on the canvas to the right of the table source operators.
2. Connect the output port of the ORDER_HEADER table source operator to the first input port of the Table Join operator with the label in.

   This propagates the virtual table definition from the output port to the input port. This can be seen by expanding the two operators. Do this by clicking the expansion button (+) in the upper right corner of each operator to see the details of the mapping. Clicking the expansion button again (-) returns the operator to its normal view. You can also view the properties for the virtual table by selecting the port label and opening the Properties View.
3. Connect the output port of the ORDER_DETAILS table source operator to the input port of the Table Join operator labeled in1. The join condition is defined using the Condition property page, depicted in Figure 2.12.

4. Clicking the ellipses button opens the Expression Builder, which is used to build the condition expression specifying which columns and how they are going to be compared.

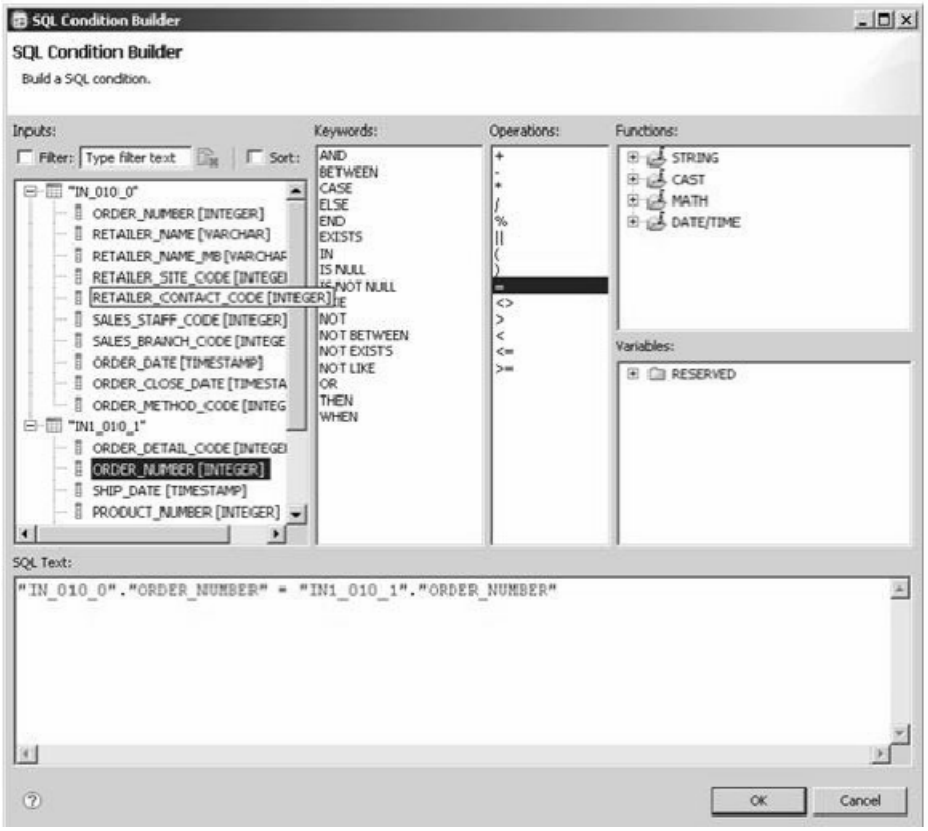


Figure 2.12 Defining the join condition using the Condition Builder

5. Use the Select List property page to select the output columns from the available input columns and to add derived columns. By default, all input columns are selected. Select only the columns needed to pass through to the next operator. These columns define the structure of the output virtual table.

Updating the target table

The final step of the data flow is to insert the resulting rows into the target table. There are multiple operators that use different techniques to update target tables from a simple insert to using DB2 bulk load utilities. In this example, we use a simple insert operation using the Table Target operator.

1. Add a Table Target operator and pick GOSALES.ORDER_FACT as the target database table, as shown in Figure 2.13.

2. Set the SQL operation property to **Insert**. A Table Target operator can insert incoming data, delete data based on the incoming data or update rows based on incoming data. These are mutually exclusive operations and do not constitute an UPSERT type of operation.

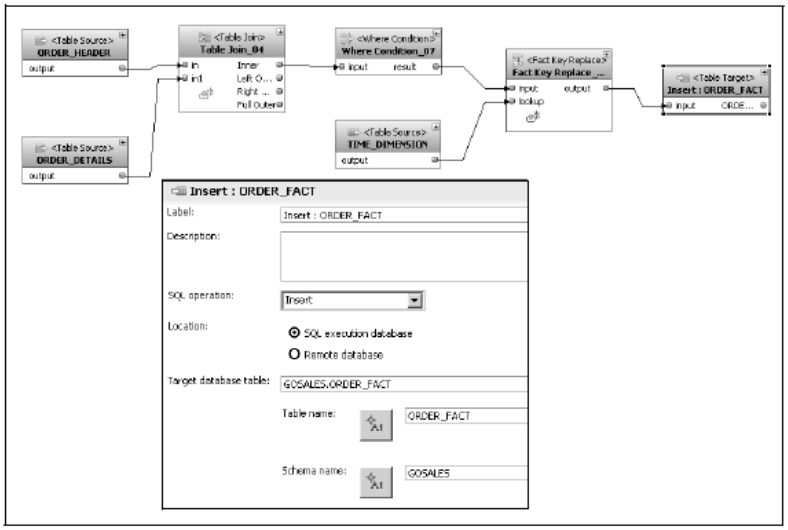


Figure 2.13 Using the Table Target operator

Validating and finding problems in a data flow

The process of validation examines the data flow to identify specific kinds of problems. Validation can be invoked by selecting **Data Flow** ☐**Validate** from the menu bar. By default, validation is also done when a data flow is saved. Validation is also invoked as part of executing a data flow in Design Studio. Problems might be errors or might be warnings. Errors prevent subsequent execution of the data flow. Warnings do not prevent execution. You have to make a judgement call on whether the warnings are a problem. Even if a data flow validates successfully, it does not guarantee that it will produce the desired results. There might be errors that validation cannot detect and there might be errors in the logic. For this reason, testing the data flow is required.

Testing and debugging

The next step is to see if the data flow does what it is designed to do. The data flow can be tested without leaving Design Studio. A data flow can be manually executed from start to finish, or breakpoints can be added to the flow and the flow executed in debug mode. How you go about testing and debugging is a matter of personal preference.

Executing a data flow

We now look at how to execute a data flow manually without debugging. Execution is started by selecting **Execute** from the data flow menu. This opens a Flow Execution dialog box where execution parameters are defined. Once the run parameters are set, they can be saved in a run profile that could be used in subsequent executions. Click **Execute** to begin execution.

Debugging a data flow

Due to the nature of optimized generated code it can be difficult isolating which operator caused a particular issue. The debugger can be used to help isolate where in a data flow an error is occurring. Breakpoints can be added to the connectors between operators. In a debug session execution stops at the defined breakpoint and shows the number of rows flowing through the breakpoint. The rows in the breakpoint can be sampled. You can even execute SQL against the breakpoint rows. This is because tables are created for each breakpoint, and data flowing through the connector is stored in that table.

Code generation

The data flow is a logical flow model that represents the movement and transformation activities needed to accomplish the intended tasks. From this flow model comes DB2-specific SQL code. That code is generated as part of the test execution process and as part of the deployment preparation process. The generated SQL is embedded into an internal script-like control language that the runtime server uses to control the execution of the SQL. This is called the execution plan graph (EPG). You can explicitly generate and view the EPG.

Lab Exercise

1. Consider the following database of student enrollment in courses & books adopted for each course. Draw the data flow to perform the tasks mentioned below.
   Department (deptid, dname, location)
   Student (snum, sname, deptid, slevel, age)
   Faculty (fid, fname, deptid)
   Class (cname, time, room, fid)
   Enrolled (snum, cname, grade)
   a) List the students in sorted according to their names in alphabetical order.
   b) List all the students studying in fifth semester.
   c) List all enrolled student's numbers whose grade is 'C'.

2. Consider the employee database where the primary keys are underlined. Draw the data flow diagrams to perform the tasks mentioned below. Also write the query for each question.

   Employee (EmpID,name, street, city)
   Works (EmpID, CompID, company-name, salary)
   Company (CompID, city)
   Manages (EmpID, ManagerID)

   a) List all employees sorted as per names who belong to "Udupi" city.
   b) List all employees ID whose salary is between 40000-80000.
   c) List all employees ID who belong to "TCS" and salary is greater than 70,000.
   d) List all employees ID who does not belong to "TCS".

Additional Questions

1. Consider the relational database given below and draw the data flow diagrams to perform the tasks mentioned below. Also write the query for each question.

   Person (driver-id#, name, address)
   Car (license, model, year)
   Accident (report-number, date, location)
   Owns (driver-id#, license)
   Participated (driver-id, car, report-number, damage-amount)

   a) Find the license numbers of "Maruti Alto" models which is registered after 2017.
   b) List driver ID's whose damage amount has crossed 1,000,00.
   c) Sort the Accident report numbers as per the accident date in descending order.

[OBSERVATION SPACE – LAB 2]

LAB NO.: 3 **Date:**

DEVELOPING DATAFLOW FOR NESTED QUERIES USING DATA WAREHOUSE TOOLS

Objectives:

In this lab students should be able to:

- ➢ Understand development of data flow for nested queries.
- ➢ Understand execution of multiple data flows using control flow.

Introduction

Nested queries can be executed using operators like group by, join etc.

Example: Consider the employee schema with employee and department tables. Figure 3.1 depicts the employee table content.

EMPID [INTEGER]	Salary [INTEGER]	Name [CHAR(5)]	ID [INTEGER]
333	5435345	Steve	1
666	234234	John	2
345	53455	Roger	1
567	545435	Sagar	1
7877	554534	Josna	2

Figure 3.1 Employee table

Suppose we need to find the highest paid employee details for each of the department. Initially join operation can performed on department and employee. The join output will be connected to the group by operator. Here the department ID and maximum of the salary is selected as shown in Figure 3.2. Select list should contain only columns on which either the aggregation functions are applied or the column is part of the group by clause. The final dataflow is shown in Figure 3.3. The result of the data flow is shown in Figure 3.4.

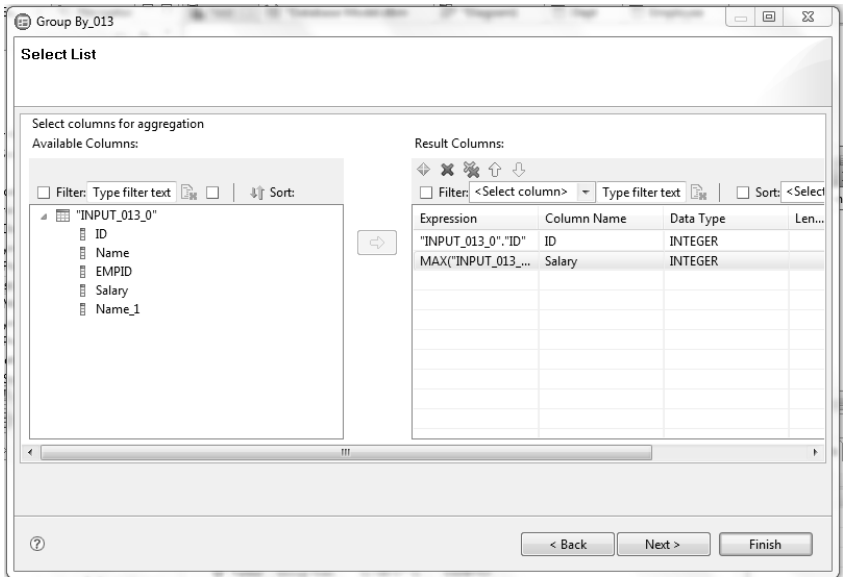


Figure 3.2 Select list of Groupby operator

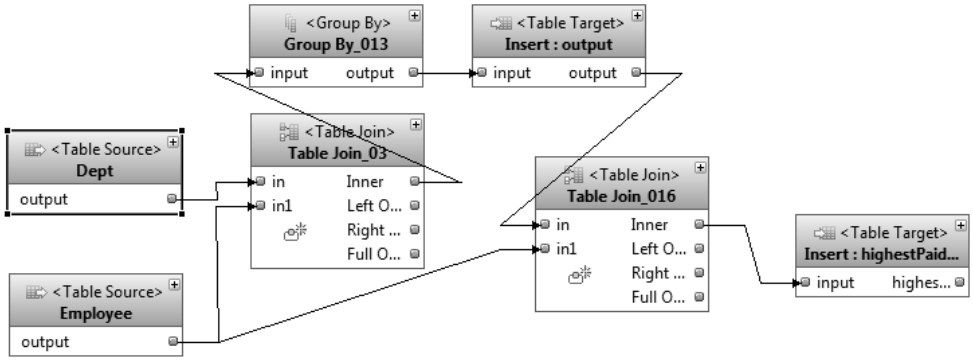


Figure 3.3 Nested Dataflow

Status | Result1

	EmpID	DeptID	Salary	EmpName
1	333	1	5435345	Steve
2	7877	2	554534	Josna

Figure 3.4 Output of the nested dataflow

Multiple data flows can be executed either serially or in parallel using control flow warehouse tool.

Control flow and subprocesses

A control flow sequences and manages the flow of activities. Activities are independent units-of-work, and could be, as examples, a data flow, a database utility, an operating system script, or a stored procedure. A control flow is the unit of execution in the runtime environment.

Design Studio provides a graphical editor with an intuitive capability to visualize and design control flows. Graphical operators model various SQW and data processing activities. By arranging these operators in a canvas work area, connecting them, and defining their properties you can create work flow models that define the sequence of execution of the activities as shown in Figure 3.3.

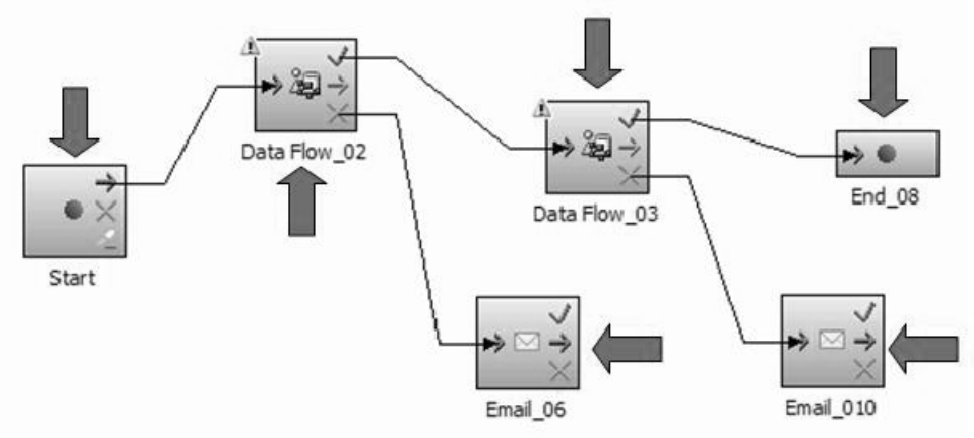


Figure 3.3 A simple control flow

A subprocess is similar to a control flow but is used like a macro or subroutine. Subprocesses can be used to define common control flow components that might be used in multiple control flows or just to break a complex control flow into multiple logical sets of operations similar to using structured techniques in a programming language.

Testing and debugging

Data flow and control flows can be tested and debugged without leaving the Design Studio development environment and without deploying the flows to the runtime environment. Live connections to the source and target databases are required. When testing from Design Studio, local client machine acts similarly to the SQW runtime

service in that it becomes the control point that manages the execution of the flows. There is no scheduling capability in the development environment. Data flows can be tested and debugged individually as they are developed. Each control flow can also be tested and debugged individually as well. There is a specific debugger for data flows and for control flows.

Deployment & Runtime environment

Deployment is the process of promoting a set of control flows from development to a runtime environment. For each runtime environment, you need perform the following steps:

1. Execute control flows automatically.
2. Monitor the execution and completion of the control flows.

Developing control flows

A control is a set of operations that accomplish the following tasks:

➢ Invoke activities of various technologies

➢ Sequence activities into a defined flow

➢ Defines how control is passed between activities based on defined conditions.

Control flow basics

Control flow is used to develop a flow model. Discrete work activities can be defined, where those activities execute, in what order, and under what conditions can be specified.

Operators, connectors, and ports

In general, control flow operators fall into two categories:

➢ Task activities
  Task activities are operators that cause a work task to happen.

➢ Flow control activities
  Flow control activities are operators that manage the flow of control. Connectors between operator ports define the path to take after the activity completes following the appropriate connector to the next activity. Figure 3.14 shows a simple control flow. This flow contains four task operators, which cause work to happen and two flow control operators. The start operator is always present in a control flow and is the point where execution starts. The end operator, End_08, is an explicit terminal

point ending a flow branch. These are flow control operators. There are two operators, data flow_02 and data flow_03, that each invoke a previously developed data flow. There are also two email operators, Email_06 and Email_010, that send a status message through the email network.

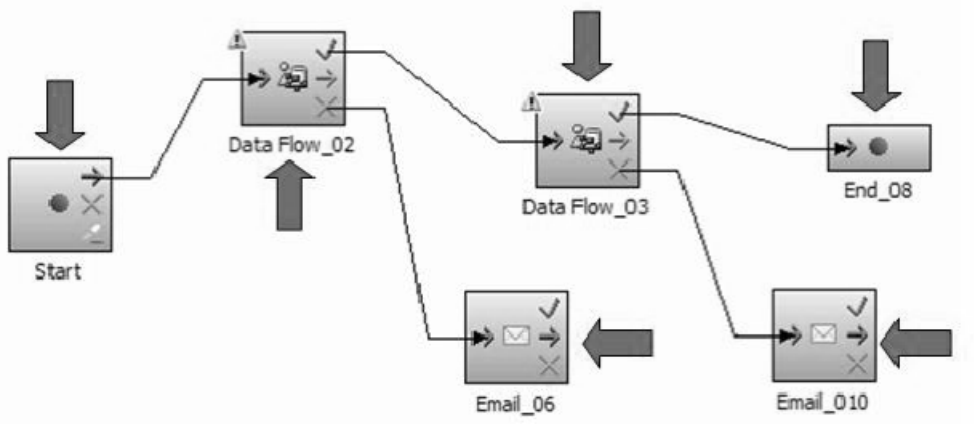


Figure 3.14 A simple control flow: Operators

Operators are sequenced together by drawing connectors between output and input ports of operators. These connectors define what operator to execute after the completion of an activity, as shown in Figure 3.15.

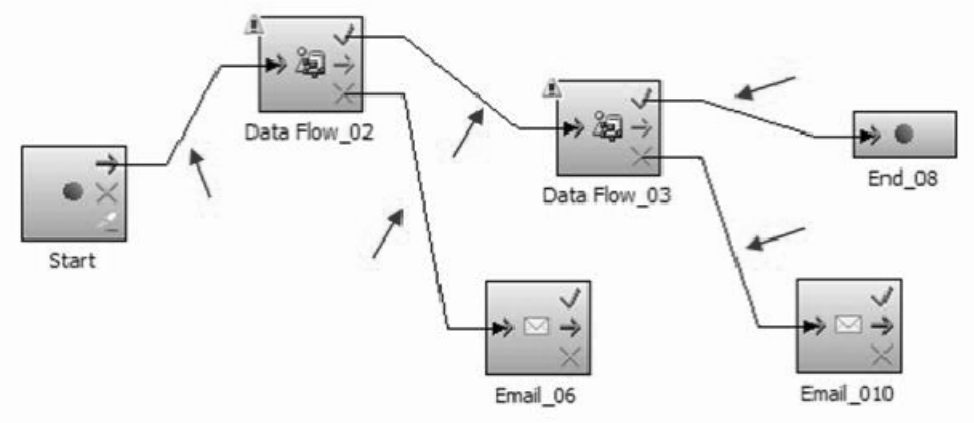


Figure 3.15 A simple control flow: Connectors

Control flow editor

In Design Studio, control flows are developed using a number of common Design Studio functions (such as the Data Project Explorer), views (such as Properties, Data Output, and Problems), and, optionally, the Database Explorer. However, the actual

creation and editing of data flows occurs in a specific graphical editor called the control flow editor, shown in Figure 3.16.

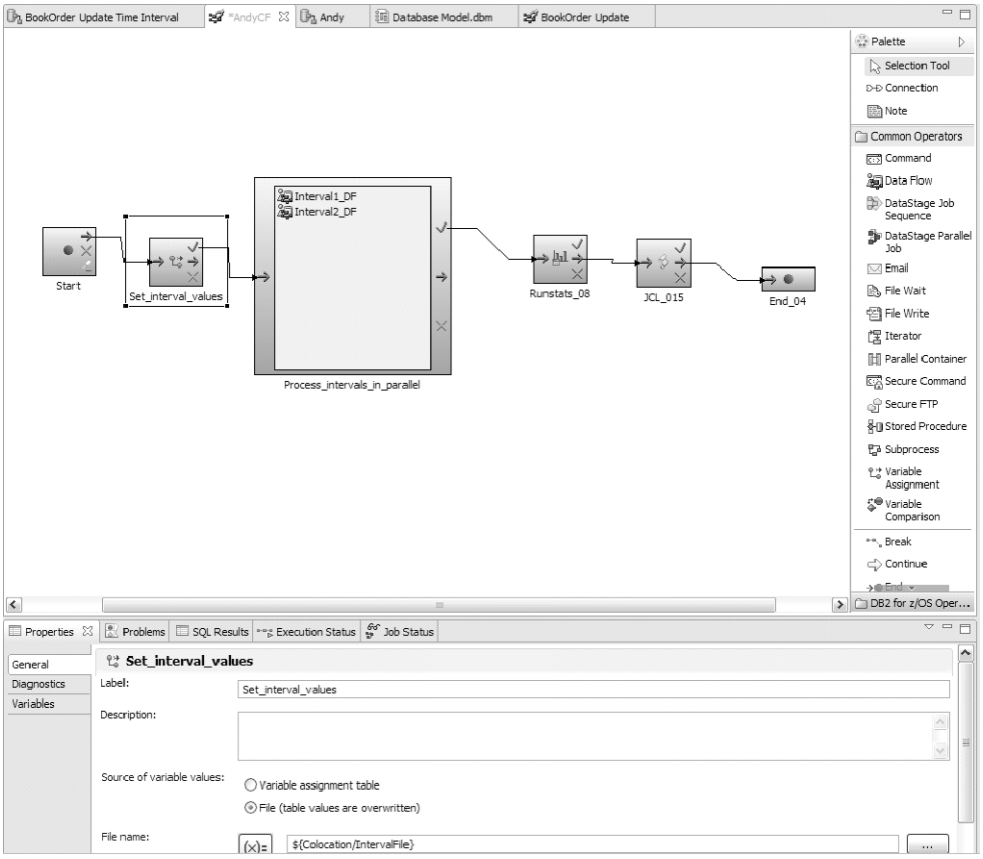


Figure 3.16 Control flow editor

Creating a new control flow

To create a new control flow, perform the following steps:

1. Open the data warehouse project (or create a new project), right-click the data flows folder and click **New ->Control Flow**. This opens a new data flow editor with an empty canvas.
2. Define the database at which this data flow executes. This is called the execution database and is typically the target database. The execution database is a property of the data flow itself. To see the properties of the data flow, click any white space on the data flow canvas. The data flow properties can be accessed in the Properties View tab.
3. Name the control flow Populate Order Star.

4. Set the SQL execution database as shown in Figure 3.17. Setting the tracing level property field to **Both** gives you the maximum amount of information in the run log, which is helpful if the flow fails and needs to be debugged.

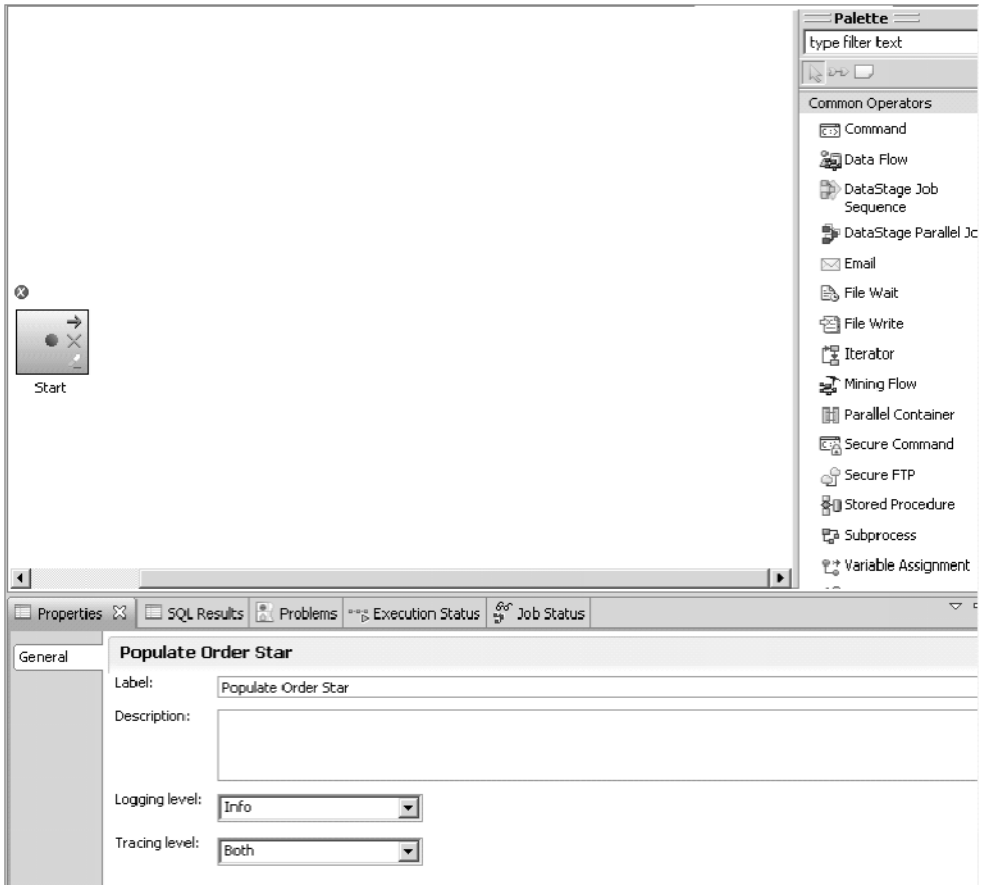


Figure 3.17 New control flow and control flow properties

Populating the fact table

The next task is to populate the fact table by executing the Populate_Orders_Fact data flow but only after the successful completion of all of the dependent activities that are executed in parallel.

1. Add a data flow operator to the canvas and connect the On-Success port of the parallel container to the input port of the data flow operator. The On-Success port is only taken if all of the activities in the parallel operator complete successfully. If any of the operators fail, the On-Failure port is taken.
2. Set the properties of the data flow operator so it executes the Populate_Orders_Fact data flow.

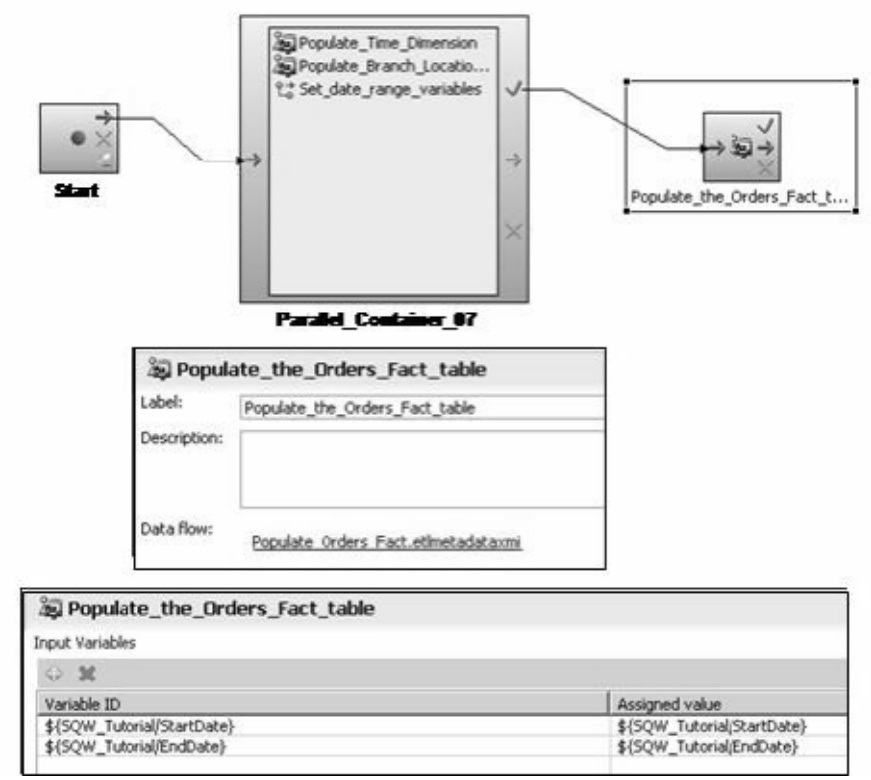


Figure 3.18 Data flow operator with mapped variables

Updating DB2 statistics

After a massive update to a table, update the database statistics so that the DB2 optimizer has the latest information for creating access paths.

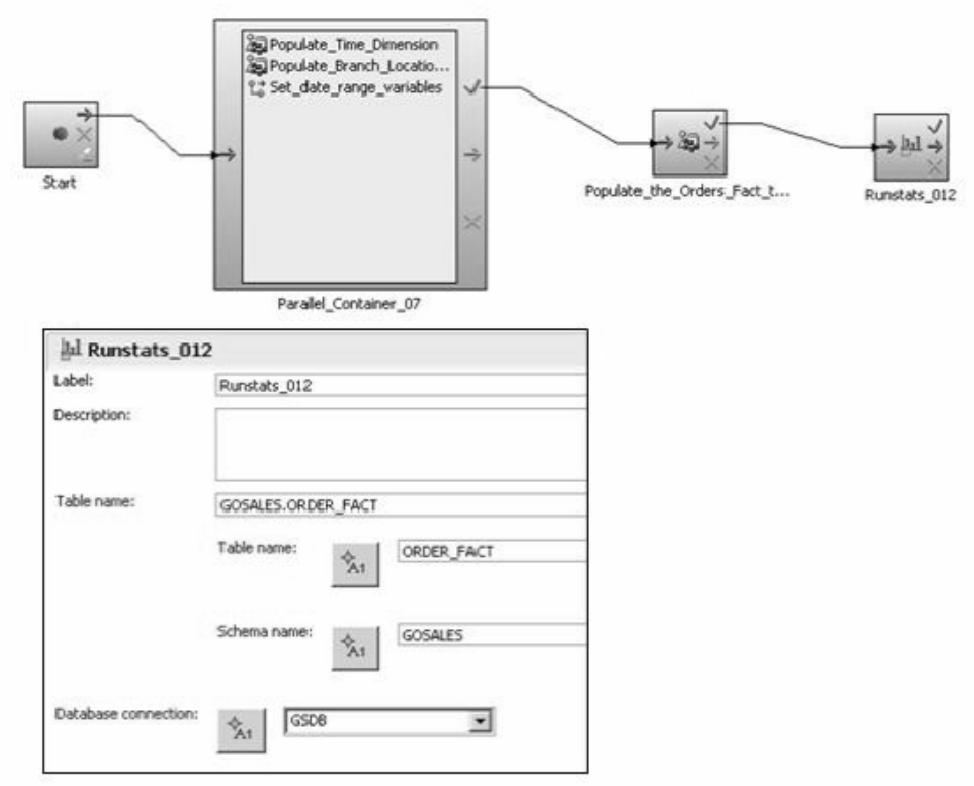


Figure 3.19 Data flow operator with database connection selection

Terminating the branch

At this point all of the tasks for the main nominal processing branch have been added to the data flow. After the runstats operator, there are no connections. At execution time this results in an implicit completion of the data flow and the data flow completes without a problem. However, it is good practice to end the flows with either the end or fail terminal operator.

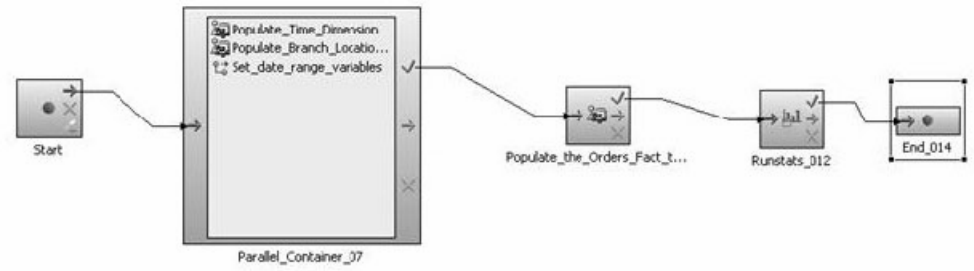


Figure 3.20 Terminating the branch with the end operator

Testing and debugging

The next task is to see if the control flow does what it is designed to do. The control flow can be tested without leaving Design Studio. That is, it can be manually executed from start to finish or breakpoints can be added to the flow and the flow executed in debug mode.

Debugging the flow

A control flow behaves more like a traditional program, with independent units of execution in a sequential fashion. Debugging is also more traditional as compared to the data flow debugger. You can set breakpoints on individual operators which, during the debug session, pause just before executing that operator. You have the option to execute to the next operator to the next breakpoint, to execute to the end of the control flow, or to cancel.

Executing the flow

We could continue to execute in debug after correcting the above error. However, we choose to execute the flow providing the correct value for the ApplicationPath variable. Perform the following steps:

1. Select **Execute** from the control flow menu. This opens the flow execution dialog box.
2. Set the value of the ApplicationPath variable to c:/OrdersApp. This is the default value.
3. Execute the flow. It successfully completes and populates the target ORDER_FACT table.

Code generation

When code is generated for a control flow, code for all of the underlying data flows and subprocesses is also generated. The code generated for a control flow is an EPG similar to the EPG we saw with a data flow.

Lab Exercise:

1. Consider the following database of student enrollment in courses & books adopted for each course. Draw the data flow to perform the tasks mentioned below.
   Department (deptid, dname, location)
   Student (snum, sname, deptid, slevel, age)
   Faculty (fid, fname, deptid)
   Class (cname, time, room, fid)

Enrolled (snum, cname, grade)

a) Get student numbers and names of those students who have an 'A' in all the courses they are enrolled in.
b) Get the names of faculty teaching a class in room 'AB5-309'. If the same faculty teaches for more than one class in the same room, only one entry should be listed.

2. Consider the employee database where the primary keys are underlined. Draw the data flow diagrams to perform the tasks mentioned below. Also write the query for each question.

Employee (EmpID,name, street, city)
Works (EmpID, CompID, company-name, salary)
Company (CompID, city)
Manages (EmpID, ManagerID)

a) Find the names and cities of residence of all employees who work for First Bank Corporation.
b) Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.
c) Find all employees in the database who live in the same cities as the companies for which they work.
d) Find all employees in the database who live in the same cities and on the same streets as do their managers.
e) Find all employees in the database who do not work for First Bank Corporation.
f) Find all employees in the database who earn more than each employee of Small Bank Corporation.
g) Find all employees who earn more than the average salary of all employees of their company.

Additional Question :

1. Consider the relational database given below and draw the data flow diagrams to perform the tasks mentioned below. Also write the query for each question.

Person (driver-id#, name, address)
Car (license, model, year)
Accident (report-number, date, location)
Owns (driver-id#, license)
Participated (driver-id, car, report-number, damage-amount)

a) Find the total number of people who owned cars that were involved in accidents in 1989.
b) Find the number of accidents in which the cars belonging to "John Smith" were involved.
c) Add a new accident to the database; assume any values for required attributes.
d) Delete the Mazda belonging to "John Smith".
e) Update the damage amount for the car with license number "AABB2000" in the accident with report number "AR2197" to $3000.

[OBSERVATION SPACE – LAB 3]

LAB NO.: 4 **Date:**

CUBE CREATION USING INFOSPHERE WAREHOUSE

Objectives:

In this lab students should be able to:

- Understand development and use of CUBE

Introduction

Online Analytical Processing (OLAP) is a popular and powerful data analytical method, and is a core component of data processing. It gives users the ability to interrogate data by intuitively navigating data from a summary level to a detail level. InfoSphere Warehouse Cubing Services enables OLAP applications to access terabytes of data through industry-standard OLAP connectivity. This warehouse-based OLAP capability is a core pillar of the InfoSphere Warehouse analytics platform. An overview of modeling and deploying OLAP functionality using Design Studio is discussed and is referred to as *Cubing Services*.

Implementing OLAP

OLAP is a key component of many data warehousing and business intelligence projects and often is a core requirement from users. OLAP enables users (business analysts, managers, and executives) to gain insight into data through a fast, consistent, and interactive interface, enabling increased productivity and faster decision making.

Dimensional model

The OLAP style of reporting is typically performed using a specific type of data organization called multidimensional modeling. Multidimensional modeling organizes the data into facts, also called measures and dimensions. Facts are the data values that you want to analyze. Dimensions contain the values of how you want to view the data. In this example, a “sales amount “is a fact that we want to measure, while “store”, “region”, and “time” are the dimensions (or the way in which we want to see the data presented). This is called viewing sales by store, by geography, and by time and is depicted in Figure 4.1.

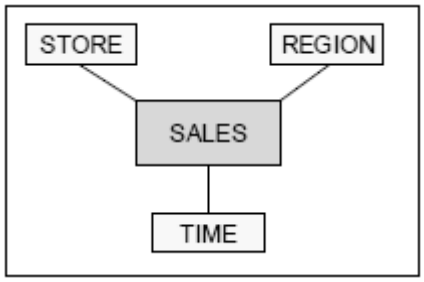


Figure 4.1 A simple star-schema

In Figure 4.1, all of the values from where we want to derive analytical information are contained in a relational table called SALES, which is the fact table. All of the other tables, such as STORE, REGION, and TIME, contain descriptive information, and define how you can view the data. These tables are the dimensions of the star. The data in the fact table refers to the data in a dimension table using the primary key of each dimension table. Fact tables are many times larger than dimension tables. In a star-schema design, the data is denormalized, which results in fewer tables that need to be joined to satisfy a user query.

In addition to the relationship between fact and dimensions, there are data relationships present in a dimension. Attributes in a dimension represent the way in which data is summarized or aggregated. These are organized in a hierarchy.

Providing OLAP data

Although relational databases are good at ad hoc types of queries, the ability to deliver highly aggregated data effectively has traditionally been an issue. It is an issue that has led to the development of speciality database engines that delivered highly-aggregated data with good response times. These specialty databases used special storage mechanisms and pre-aggregated the data along all dimensions using highly-indexed systems to access specific cells of aggregated data across the various dimensional values directly. These types of databases were called *Multidimensional OLAP databases* (MOLAP).

Although MOLAP systems delivered highly-aggregated data across dimensions with impressive response times, there was definitely an associated cost. These databases required that data be extracted from the data warehouse and loaded into the MOLAP database, duplicating the data. MOLAP databases also required that the data access paths be pre-calculated across all combinations of dimension members, which resulted in an exponential growth rate in the required data storage. This calculation step was also

expensive in terms of the CPU resource cost and elapsed time. However, the MOLAP engines did provide excellent response times. On the other end of the spectrum, there were products that used the relational database as the OLAP engine. These products as known as *Relational OLAP engines* or *ROLAP engines*. ROLAP products developed a way to provide aggregated data in an effective manner, usually using summary tables. The ROLAP engines determined, either automatically or through user choice, whether to use summary tables or the detail tables to satisfy a query.

InfoSphere Warehouse Cubing Services takes a hybrid approach to delivering OLAP. From a user or application viewpoint, it appears to be a MOLAP engine using multidimensional expressions (MDX) as the query language. However, the data is not copied or persisted in another storage mechanism, but remains in the DB2 relational engine. Cubing Services then, is what we call a *multidimensional OLAP hot cache*. That is, InfoSphere Warehouse Cubing Services loads data into a multidimensional structure in memory.

Modeling OLAP with InfoSphere Warehouse Design Studio

In Design Studio, creating an OLAP model primarily takes place using the Data Source Explorer, Data Project Explorer, and the Properties View, as seen in Figure 4.2. This is a busy figure, and you are not expected to read the contents. It is to give you a perspective of the segments in Design Studio.

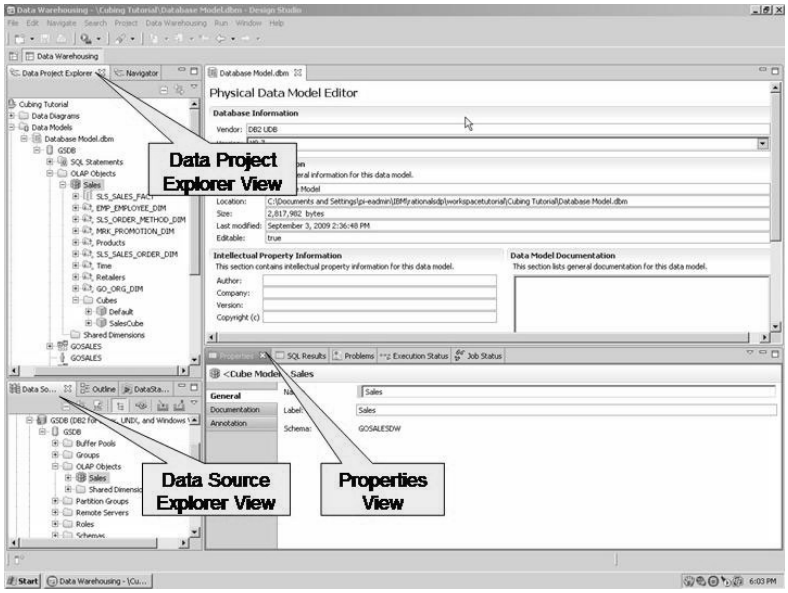


Figure 4.2 Design Studio OLAP views

Data Source Explorer

The Data Source Explorer provides a tree view that allows the user to browse the catalog contents of a relational database. Initially a connection to a database is created, then the connection node can be expanded to reveal a database icon, which can be further expanded to display nodes representing the catalog contents of the database.

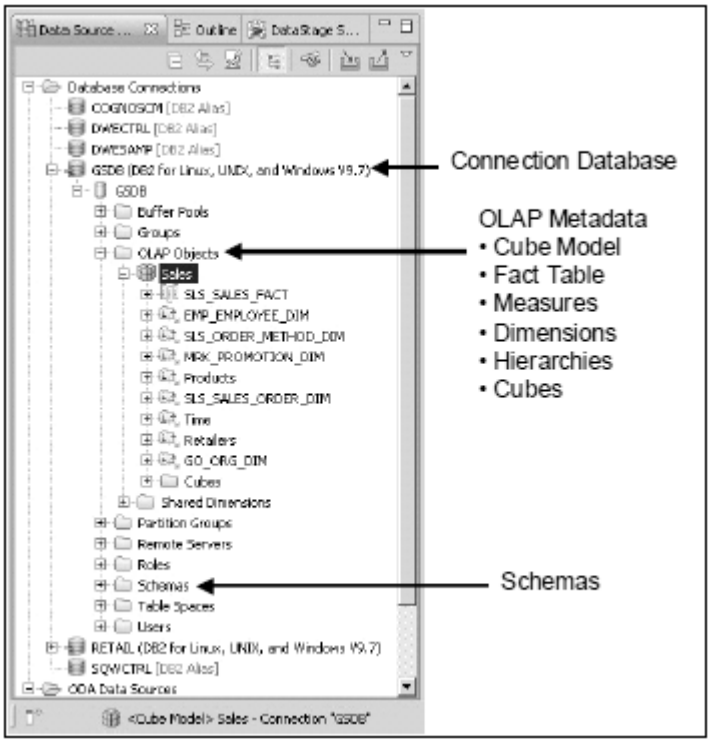


Figure 4.3 Data Source Explorer

Properties View

The Properties View, as shown in Figure 4.4, allows the users to view the properties of the selected object. The Properties View is applicable to objects in either the Data Source Explorer or the Data Project Explorer. Properties for objects selected in the Data Source Explorer are read only. Properties for objects selected in the Project Explorer are editable.

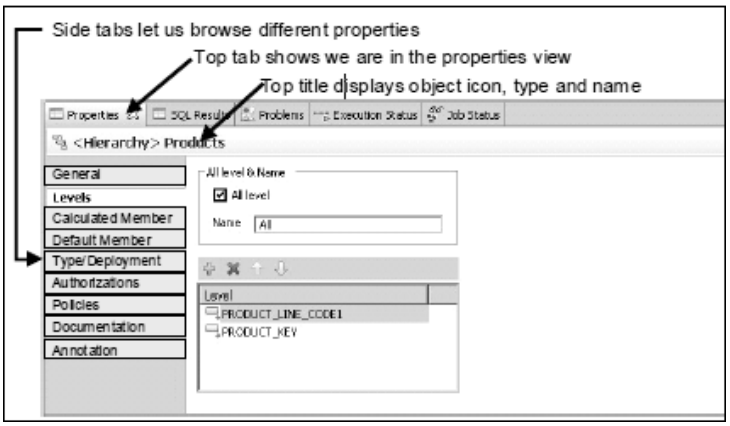


Figure 4.4 Properties view

Data Project Explorer View

The Data Project Explorer View in Design Studio is used to view projects and data models. A project can contain one or more physical data models, and each physical data model is a model of either a complete relational database or a part of a relational database such as a schema and its contents. Physical data models can contain OLAP objects as well as the traditional relational objects such as tables and views, as shown in Figure 4.5.

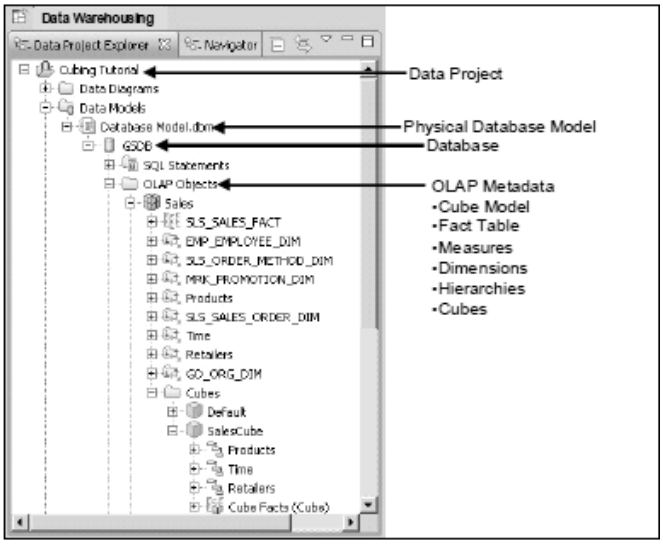


Figure 4.5 The Data Project Explorer View

A physical data model containing OLAP objects can be created with the following approaches:

- Reverse engineering an existing DB2 database containing OLAP objects

- Creating an empty physical model and creating OLAP objects by:
- Importing objects from an OLAP XML file
- Using the quick start wizard
- Using the context menus (right-click)
- Manually copying objects from a database in the Data Source Explorer using either the clipboard or drag and drop

Creating a new data project

A data project needs to be defined before an OLAP model can be created. There are several types of data projects, but for Cubing Services use the data design project (OLAP).

1. From Design Studio, select **File →New→Data Design Project (OLAP)**, as shown in Figure 4.6.

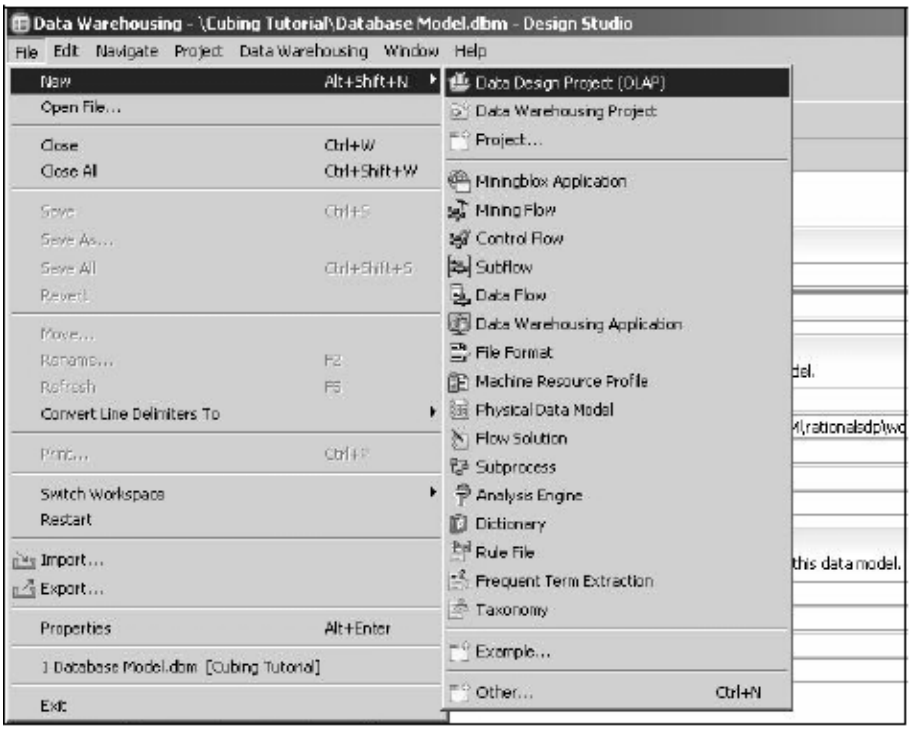


Figure 4.6 New data project

Creating a new physical data model

Before the OLAP model is created, the physical data model of the underlying base tables (facts and dimensions) should be defined.

1. Expand the Cubing Tutorial project.

2. Right-click the **Data Models** folder and use the context menu to select **New →Physical Data Model**, as shown in Figure 4.7.

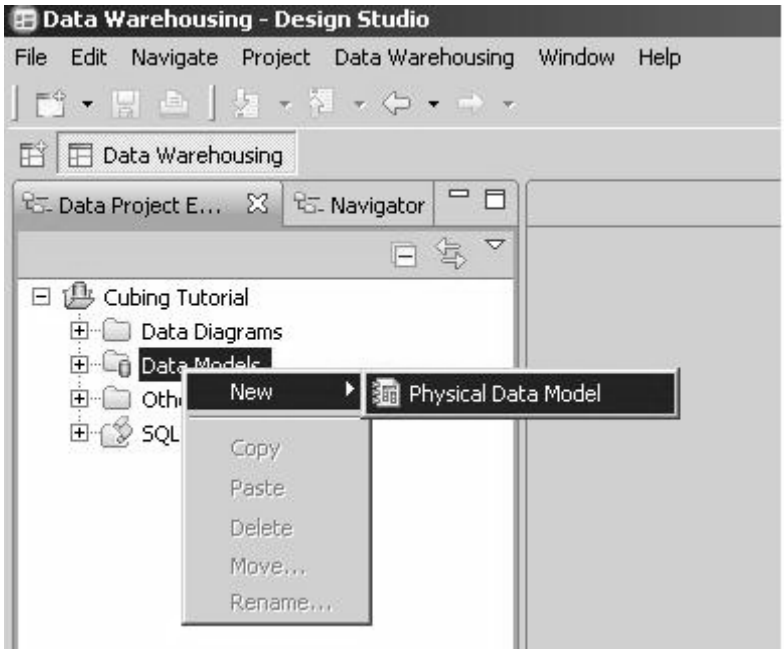


Figure 4.7 New physical data model

3. In the File Name box, give the data model a name.
4. Select the appropriate database and version (DB2 and v9.7 in this example), as shown in Figure 4.8.

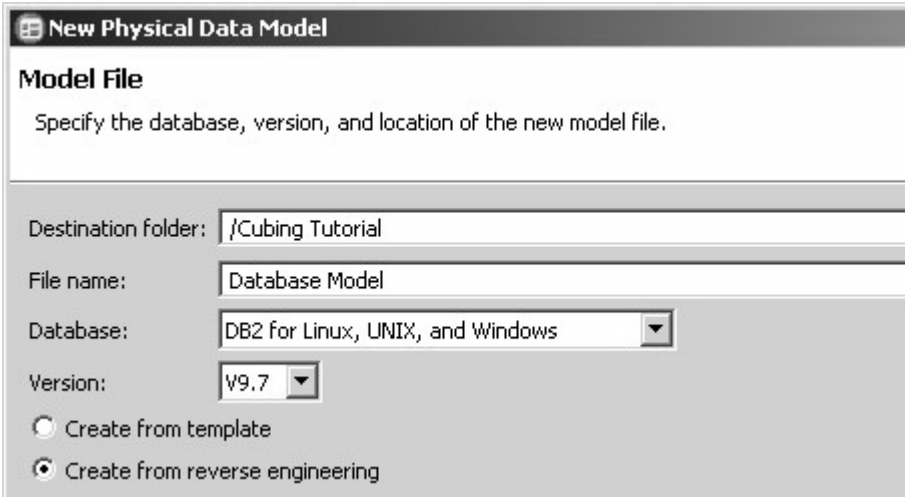


Figure 4.8 New physical data model options

5. Select the **Create from reverse engineering** radio button and click **Next**.
6. From the "New Physical Data Model" window (Figure 4.9), select the **Database** radio button. This specifies that an existing database will be used.

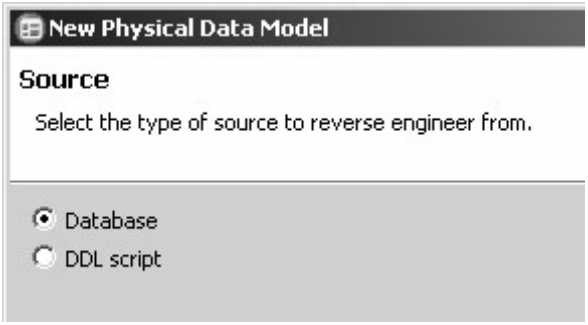


Figure 4.9 Reverse engineering options

7. On the "Select Connection" window (Figure 4.10), select **Use an existing connection**, select the **GSDB** connector, and click **Next**.

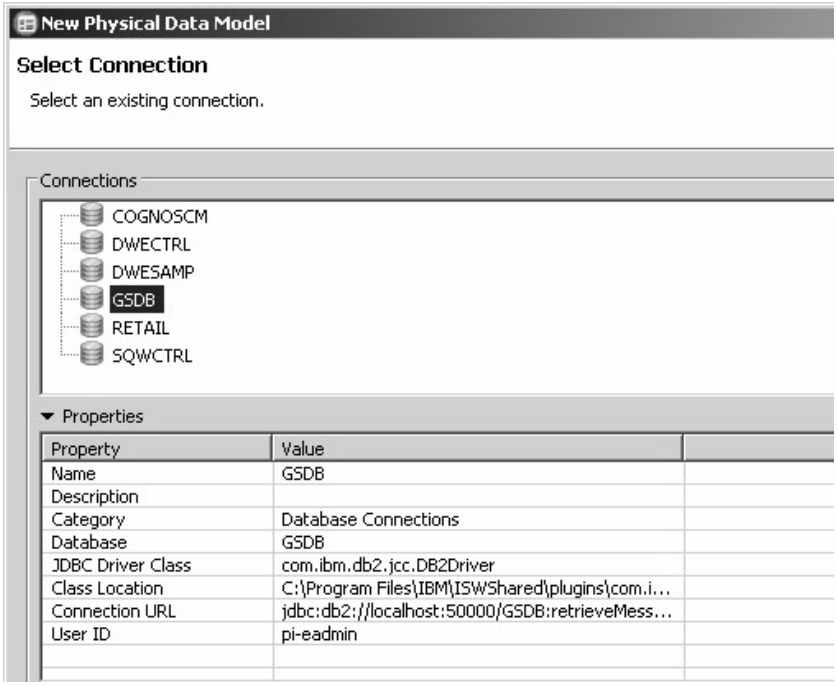


Figure 4.10 New physical data model connections

8. On the "Select Schema" window (Figure 4.11), click **Select All** to import all schemas, and click **Next**.

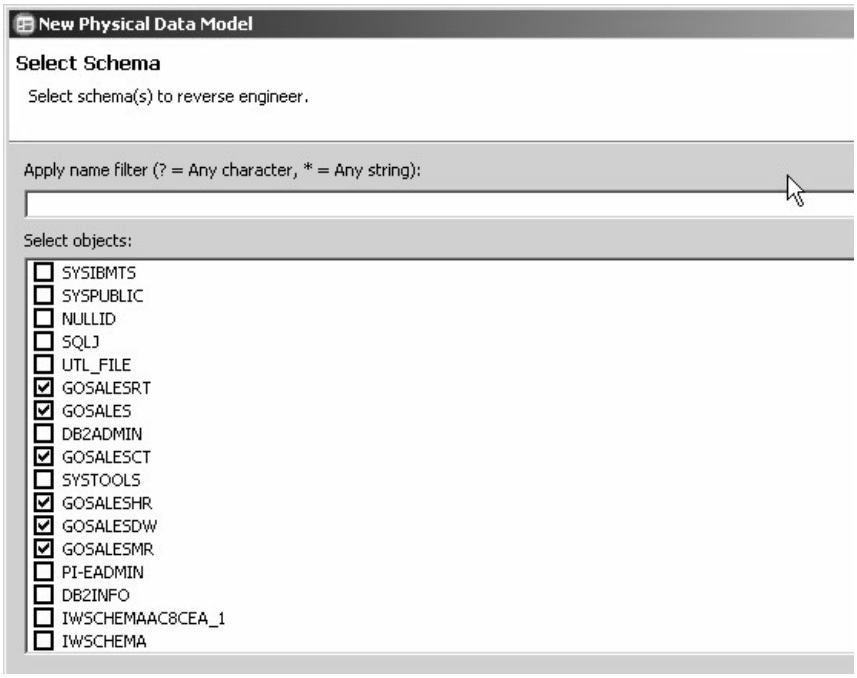


Figure 4.11 Select schemas

9. On the "Database Elements" window (Figure 4.12), accept the default database elements, and click **Next**.

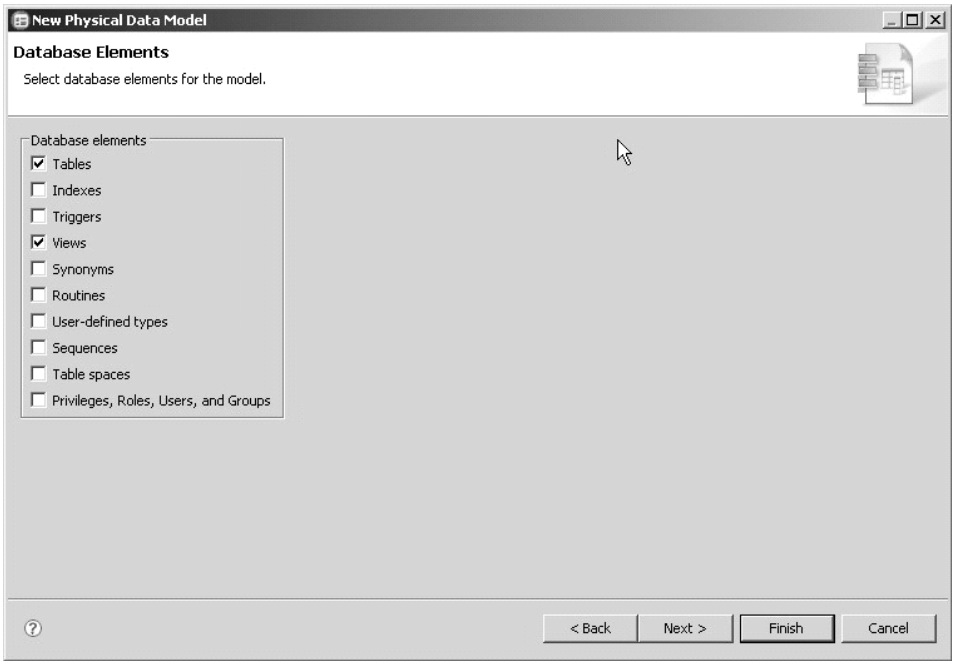


Figure 4.12 Database elements options

10. On the "Options" window (Figure 4.13), select **Overview** in the Generate Diagrams field, and click **Finish**.

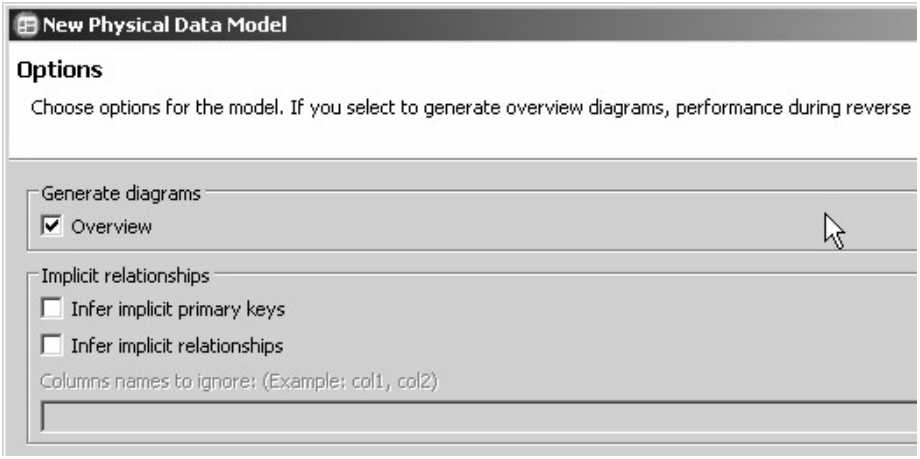


Figure 4.13 Generate diagrams wizard

Cube Model wizard

Cubing Services provides a wizard to define OLAP objects in a physical data model. This can be opened by using the context menu, as shown in Figure 4.14.

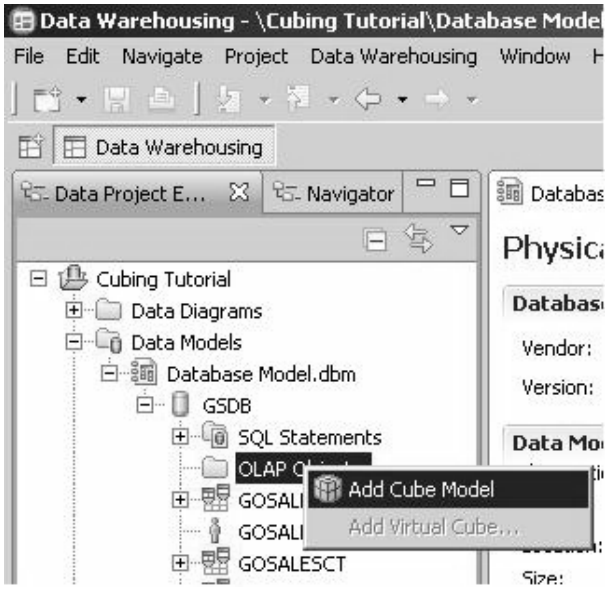


Figure 4.14 Cube model wizard

The Cube Model wizard creates the OLAP objects that it can logically infer from the relational star schema or snowflake schema. Based on the fact table, the wizard detects the corresponding dimensions, joins, attributes, and measures. Before using the Cube

Model wizard, make sure that the star schema database is well formed with primary keys and foreign key pairs and referential integrity in place, either enforced or informational. The wizard completes the cube model by creating the dimensions, attributes, and joins using the RI constraints.

Defining the fact table

When you have started the Cube Model wizard, the first step is to identify the fact table.

1. To start the Cube Model Wizard, expand the Data Model folder until you reach the OLAP Objects folder. Right-click the **OLAP Objects** folder and select **Add Cube Model.**
2. The "Select a Facts Table or View" window (Figure 4.15) opens. Select the **Show only best candidates** check box, expand the GOSALESDW schema and select the **SLS_SALES_FACT** table. Click **Next**.

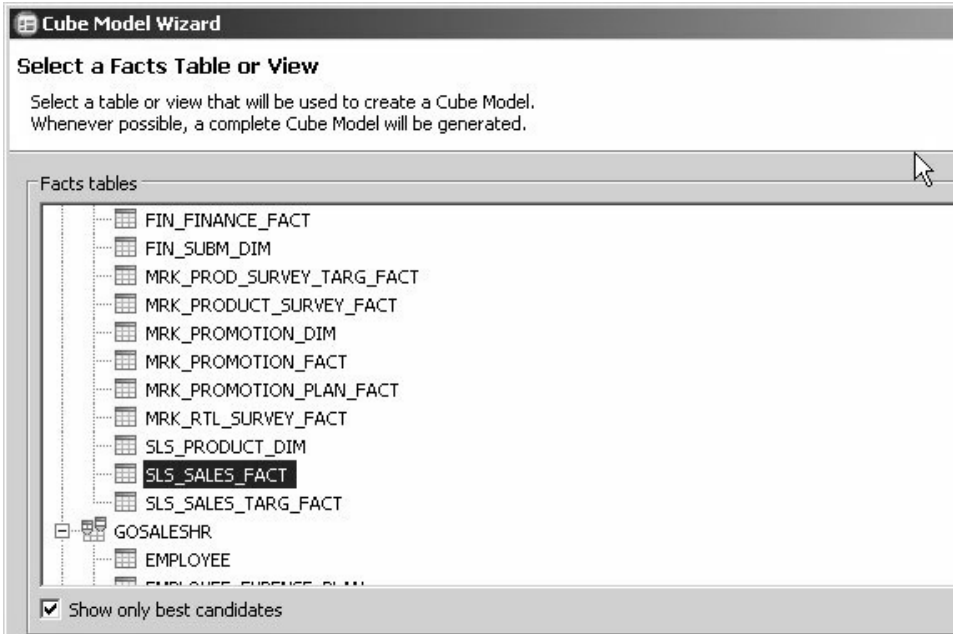


Figure 4.15 Cube model wizard: Fact table selection

3. The "Summary" window lists the dimensions and hierarchies that have been created for the cube model. You can rename an object by double-clicking it to give them more descriptive names.
4. Click **Finish**.

Creating a new cube

The default cube created by the wizard contains all of the metadata from all of the schemas that make up the cube model. You might want to create a cube with only a subset of this metadata to optimize the response time for your queries.

To create a new cube, perform the following steps:

1. Right-click the **Cubes** folder of the new cube model, and select **Add Cube,**
2. Highlight the cube and change its name in the Properties View to SalesCube.

Adding hierarchies and measures to a cube

SalesCube currently contains no measures or hierarchies, and the cube is empty. Hierarchies and measures must be added to the cube so that it can be made available for users to query.

To add hierarchies to a cube, perform the following steps:

1. Right-click the **SalesCube** and select **Add Data Object→ Hierarchy** as shown in Figure 4.16.

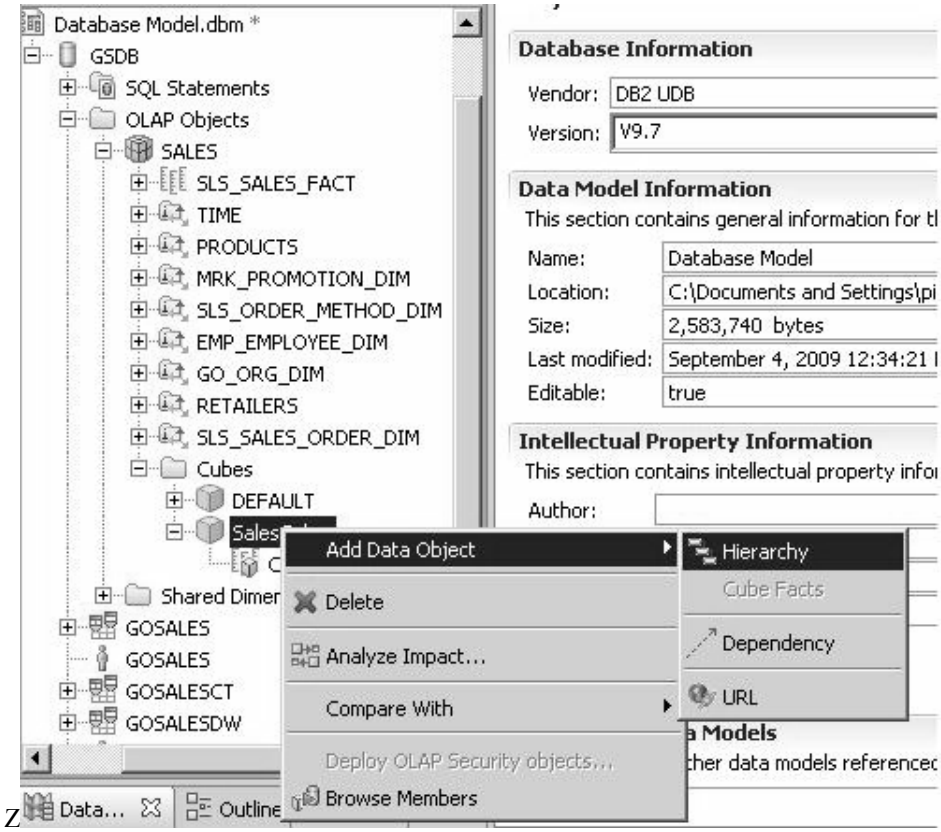


Figure 4.16 Adding hierarchy

2. Select **Time**, **Products**, and **Retailer hierarchies** from the list of available hierarchies, as shown in Figure 4.17.

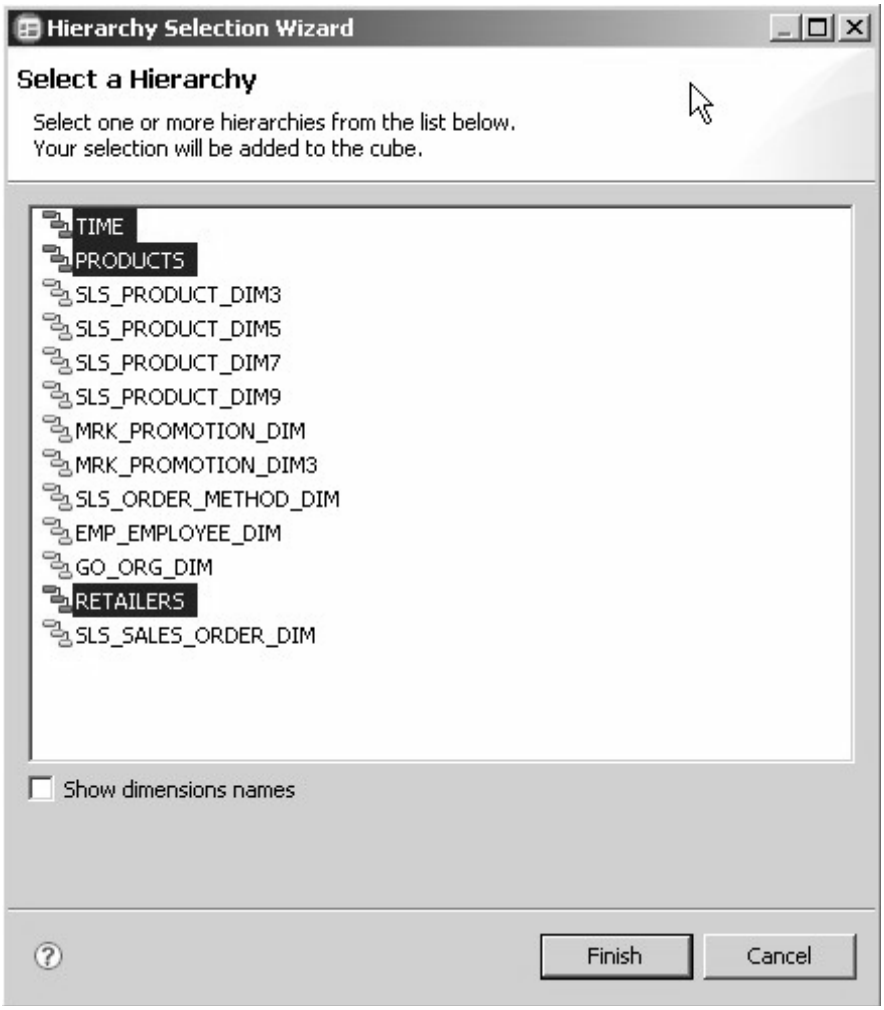


Figure 4.17 Selection of hierarchy

To add measures to a cube, perform the following steps:

1. Expand the SalesCube to locate the Cube Facts object. Right-click the **Cube Facts (Cube)** object as shown in Figure 4-18 and select **Add Data Object →Cube Measures**.

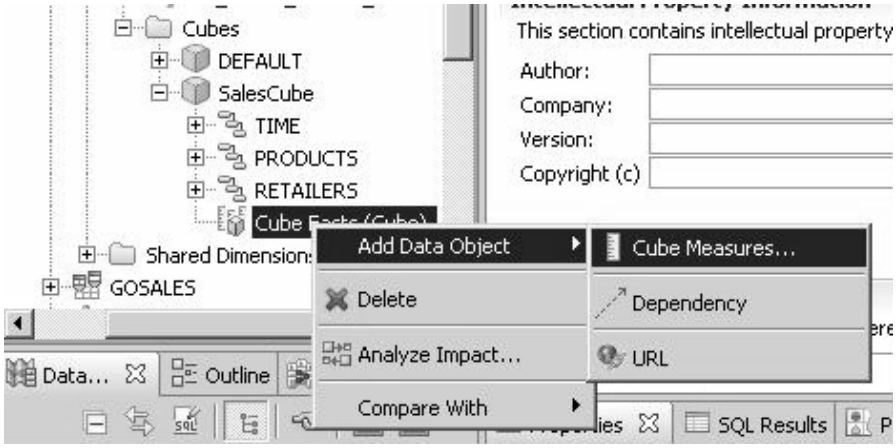


Figure 4.18 Adding cube measures

2. Select the **SALE_TOTAL** and **UNIT_SALE_PRICE** check boxes, as shown in Figure 4-19. Click **OK**.

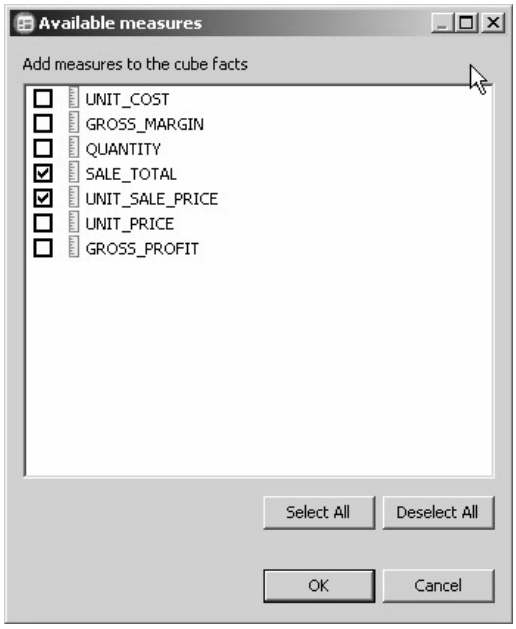


Figure 4.19 Selecting measures

Modifying a hierarchy

To create new levels for a hierarchy, you need to add attributes that can be used to define the level. Attributes map to a column in the physical tables. To add attributes, perform the following steps:

1. Locate the Attributes folder in the Retailers dimension.
   Right-click the **Attributes** folder and select **Add attributes from columns**.

To add a level to a hierarchy, perform the following steps:
1. Right-click the **Levels** folder under the Retailers hierarchy and select **AddLevel**

Adding tables to a cube model
To add an additional table to a cube model, perform the following steps:
1. Locate the Tables folder in the Retailers dimension. Right-click the **Tables** folder and select **Add Existing Table** (Figure 4.20).

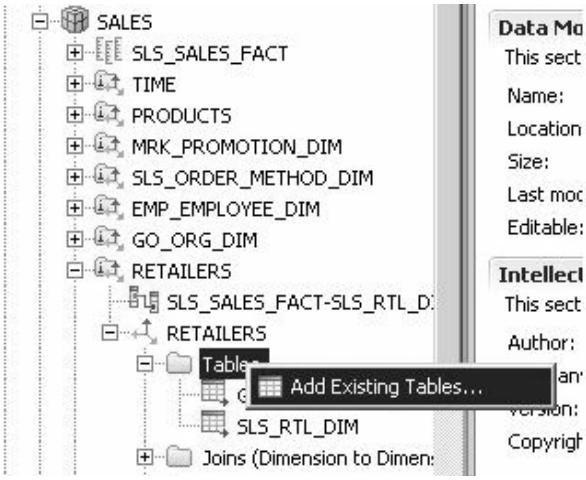


Figure 4.20 Adding tables

Analyzing OLAP objects for validity
After creating a physical data and OLAP model, the models can be validated using the Analyze Model option available from the context menus displayed from the Data Project Explorer. A cube model validates successfully only after you add the following mandatory components:
- At least one facts object
- At least one dimension
- A hierarchy defined for at least one dimension
- Joins between the existing facts objects and dimensions
- Attributes that reference existing table columns

To validate a cube model in the Data Project Explorer:
1. Right-click the **SalesCube model** and select **Analyze Model**
2. Select the **Physical Data Model** check box. All of the rules beneath the Physical Data Model object are also selected

Exporting a cube model from Design Studio
To export a cube model to an XML file, perform the following steps:
1. Click **File →Export**.

2. Expand **Data Warehousing**, and select **OLAP metadata**. Click **Next**.
3. Specify the file name. Browse to C:\temp and save the file.
4. Expand the GSDB container and select the **SalesCube** model. Click **Finish**.

Running the Cubing Services Optimization Advisor

1. Right-click the **Sales** cube model in the Data Source Explorer and select **Optimization Advisor**.
2. Ensure that the **InfoSphere Cubing Services Cube Server** radio button is selected as the target query, as shown in Figure 6-53. Click **Next**.
3. Ensure that the **Deferred** radio button in the summary table is selected.
4. Specify a maximum time limit of 1 minute.
5. Click **Start Advisor**.
6. After the Advisor finishes, click **Next**.

Specify the file path and name of the SQL script that creates the recommended MQTs shown in Figure 4.21. Click **Finish** to save the scripts.

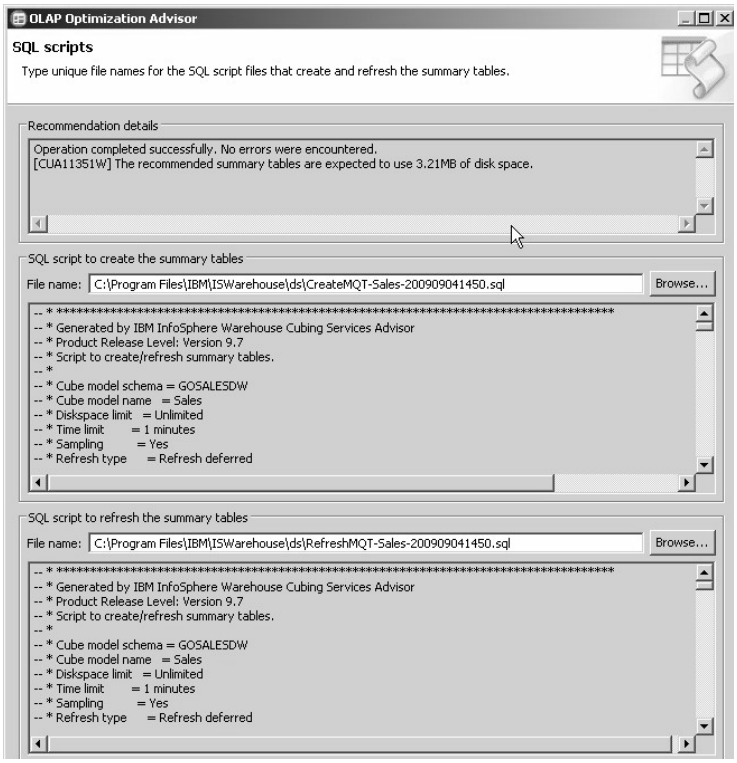


Figure 4.21 SQL script

Lab Exercise

1. Demonstrate and write the steps for the creation of Cube using Infosphere Warehouse

[OBSERVATION SPACE – LAB 4**]**

LAB NO.: 5 **Date:**

DATA PREPROCESSING

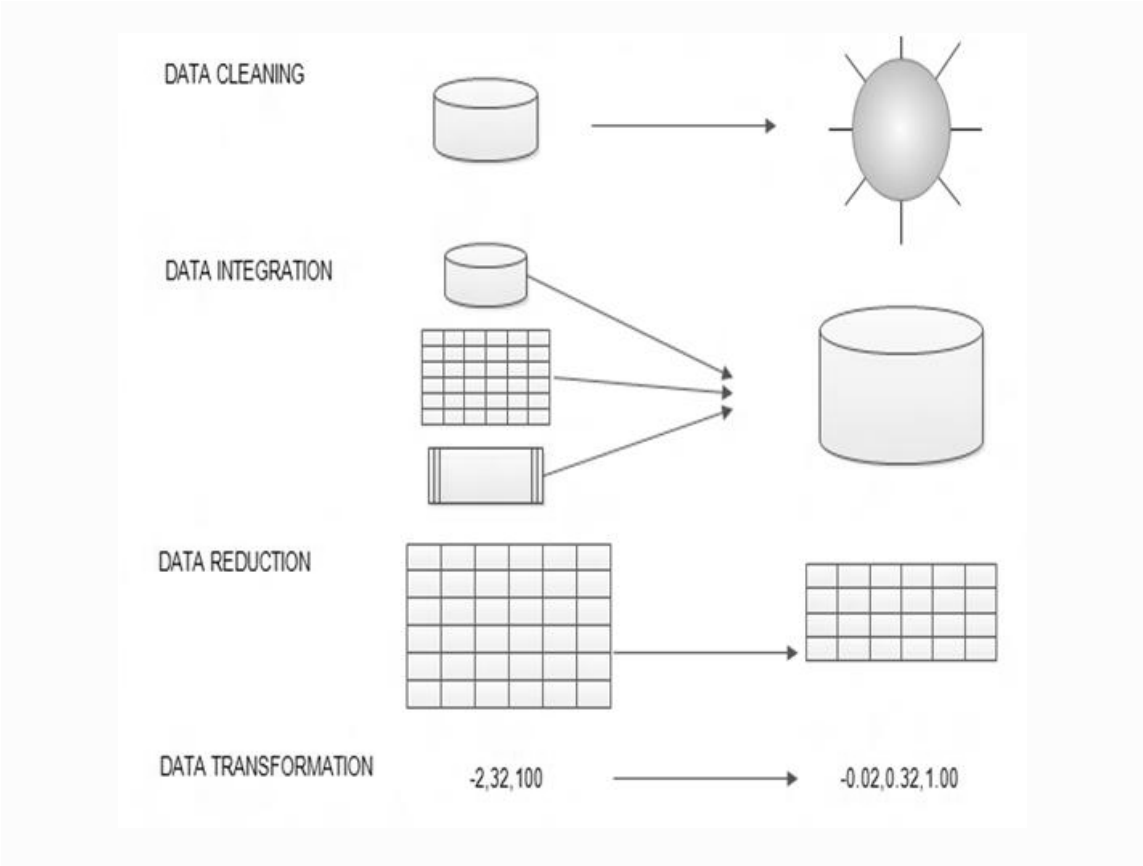
Objectives:

In this lab students should be able to:

➢ Understand preprocessing of dataset before applying any mining techniques.

Introduction

Data pre-processing consists of a series of steps to transform raw data derived from data extraction into a "clean" and "tidy" dataset prior to statistical analysis. Pre-processing aims at assessing and improving the quality of data to allow for reliable statistical analysis.



Several distinct steps are involved in pre-processing data. Here are the general steps taken to pre-process data:

Data cleaning

This step deals with missing data, noise, outliers, and duplicate or incorrect records while minimizing introduction of bias into the database.

Missing Data- There are three possible ways to deal with missing data:

- Ignore the record. This method is not very effective, unless the record (observation/row) contains several variables with missing values. This approach is especially problematic when the percentage of missing values per variable varies considerably or when there is a pattern of missing data related to an unrecognized underlying cause such as patient condition on admission.
- Determine and fill in the missing value manually. In general, this approach is the most accurate but it is also time-consuming and often is not feasible in a large dataset with many missing values.
- Use an expected value. The missing values can be filled in with predicted values (e.g. using the mean of the available data or some prediction method). It must be underlined that this approach may introduce bias in the data, as the inserted values may be wrong. This method is also useful for comparing and checking the validity of results obtained by ignoring missing records.

Noisy Data - Noise is a random error or variance in an observed variable.

- Binning methods. Binning methods smooth a sorted data value by considering their neighborhood, or values around it. These kinds of approaches to reduce noise, which only consider the neighborhood values, are said to be performing local smoothing.
- Clustering. Outliers may be detected by clustering, that is by grouping a set of values in such a way that the ones in the same group (i.e., in the same cluster) are more similar to each other than to those in other groups.
- Machine learning. Data can be smoothed by means of various machine learning approaches. One of the classical methods is the regression analysis, where data are fitted to a specified (often linear) function.

Inconsistent Data - There may be inconsistencies or duplications in the data. Some of them may be corrected manually using external references. This is the case, for instance, of errors made at data entry. Knowledge engineering tools may also be used to detect the violation of known data constraints. For example, known functional dependencies among attributes can be used to find values contradicting the functional constraints.

Data integration

Data integration is the process of combining data derived from various data sources (such as databases, flat files, etc.) into a consistent dataset. There are a number of issues to consider during data integration related mostly to possible different standards among data sources. For example, certain variables can be referred by means of different IDs in two or more sources. Once data cleaning and data integration are completed, we obtain one dataset where entries are reliable.

Data transformation

This step transforms the data values into a format, scale or unit that is more suitable for analysis. Here are few common possible options:

Normalization - This generally means data for a numerical variable are scaled in order to range between a specified set of values, such as 0–1.

Aggregation - Two or more values of the same attribute are aggregated into one value. A common example is the transformation of categorical variables where multiple categories can be aggregated into one.

Data reduction

Complex analysis on large datasets may take a very long time or even be infeasible. The final step of data pre-processing is data reduction, i.e., the process of reducing the input data by means of a more effective representation of the dataset without compromising the integrity of the original data. The objective of this step is to provide a version of the dataset on which the subsequent statistical analysis will be more effective. Data reduction may or may not be lossless. After the dataset has been integrated and transformed, this step removes redundant records and variables, as well as reorganizes the data in an efficient and tidy manner for analysis.

Pre-processing is sometimes iterative and may involve repeating this series of steps until the data are satisfactorily organized for the purpose of statistical analysis. During pre-processing, one needs to take care not to accidentally introduce bias by modifying the dataset in ways that will impact the outcome of statistical analyses. Similarly, we must avoid reaching statistically significant results through trial and error analyses on differently pre-processed versions of a dataset.

Lab Exercise

1. Apply the above mentioned pre-processing techniques using any three open source data mining tools.

[OBSERVATION SPACE – LAB 5]

LAB NO.: 6 **Date:**

IMPLEMENTATION OF APRIORI ALGORITHM TO DISCOVER FREQUENT ITEM SETS

Objectives:

In this lab students should be able to:

- Understand the Apriori algorithm to find the frequent item sets.
- Implement the algorithm using any of the mentioned languages.

Apriori Algorithm

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

Key Concepts:

- Frequent Itemsets: The set of items which has minimum support (denoted by L_i for i^{th} Itemset).
  Find the frequent itemsets: the set of items that have minimum support. A subset of a frequent itemset must also be a frequent itemset i.e., if {AB} is a frequent itemset, both {A} and {B} should be a frequent itemset. Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)
- Apriori Property: Any subset of frequent itemset must be frequent.
- Join Operation: To find L_k, a set of candidates k-itemsets is generated by joining L_{k-1} with itself.
  Join Step: C_k is generated by joining Lk-1with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ ϕ; k++) {
(3)      Ck = apriori_gen(Lk−1);
(4)      for each transaction t ∈ D { // scan D for counts
(5)         Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)         for each candidate c ∈ Ct
(7)            c.count++;
(8)      }
(9)      Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;
```

```
procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)      for each itemset l2 ∈ Lk−1
(3)         if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)            c = l1 ✖ l2; // join step: generate candidates
(5)            if has_infrequent_subset(c, Lk−1) then
(6)               delete c; // prune step: remove unfruitful candidate
(7)            else add c to Ck;
(8)         }
(9)   return Ck;
```

```
procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)      if s ∉ Lk−1 then
(3)         return TRUE;
(4)   return FALSE;
```

Example:

➢ Consider a database, D, consisting of 9 transactions.

➢ Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)

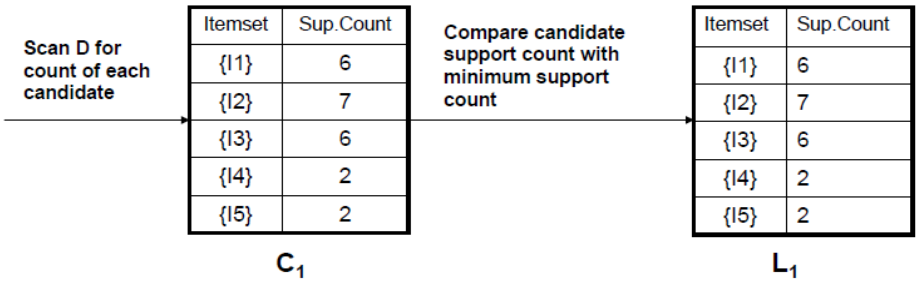
➢ Let minimum confidence required is 70%.

➢ We should first find out the frequent itemset using Apriori algorithm.

➢ Then, Association rules will be generated using min. support & min. confidence.

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

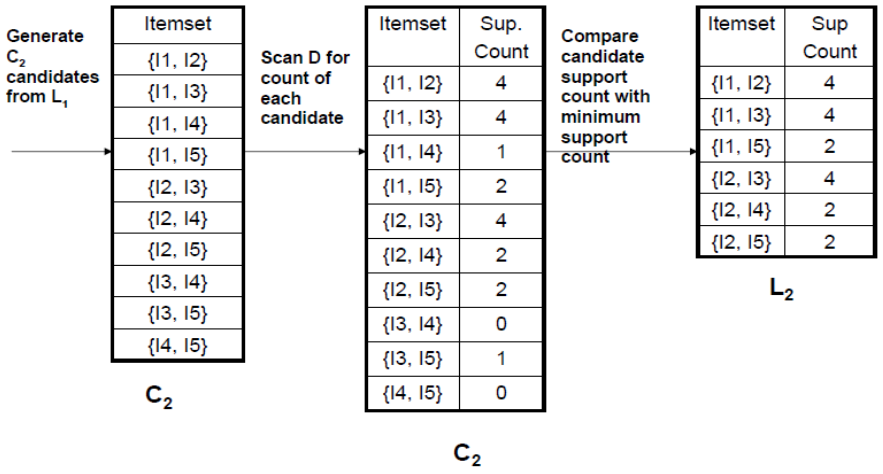
Step 1: Generating 1-itemset Frequent Pattern



➢ The set of frequent 1-itemsets, L1, consists of the candidate 1-itemsets satisfying minimum support.

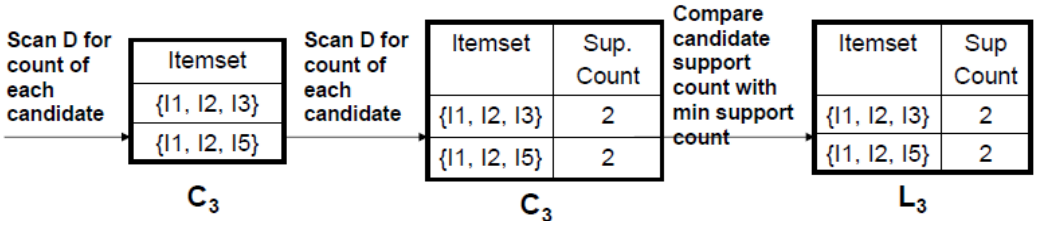
➢ In the first iteration of the algorithm, each item is a member of the set of candidates.

Step 2: Generating 2-itemset Frequent Pattern



- To discover the set of frequent 2-itemsets, L2, the algorithm uses L1 *Join* L1to generate a candidate set of 2-itemsets, C2.
- Next, the transactions in D are scanned and the support count for each candidate itemset in C2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.
  Note: We haven't used Apriori Property yet.

Step 3: Generating 3-itemset Frequent Pattern



- The generation of the set of candidates 3-itemsets, C3, involves use of the Apriori Property.
- To find C3, we compute L2*Join*L2.
- C3= L2 *Join*L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.
- Now, join step is complete and Prune step will be used to reduce the size of C3. Prune step helps to avoid heavy computation due to large Ck.
- Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent.
- For example, let's take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L2, we will keep {I1, I2, I3} in C3.
- Let's take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3, I5}.
- But, {I3, I5} is not a member of L2and hence it is not frequent violating Apriori Property. Thus, we will have to remove {I2, I3, I5} from C3.

- ➢ Therefore, C3= {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- ➢ Now, the transactions in D are scanned to determine L3, consisting of those candidates 3-itemsets in C3having minimum support.

Step 4: Generating 4-itemset Frequent Pattern
The algorithm uses L3 *Join*L3to generate a candidate set of 4-itemsets, C4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.

Thus, C4= φ, and algorithm terminates, having found all of the frequent items.
These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets
For each frequent itemset ***"l",*** generate all nonempty subsets of ***l.***
For every nonempty subset ***s***of ***l***, output the rule **"s ->(l-s)"**if
support_count(l) / support_count(s) >= min_conf
where min_confis minimum confidence threshold.

We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1, I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.
Let's take ***l*** = {I1, I2, I5}. It's all nonempty subsets are {I1, I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

Step 5: Generating Association Rules from Frequent Itemsets
Let minimum confidence threshold is, say 70%.
The resulting association rules are shown below, each listed with its confidence.
R1: I1 ^ I2 ->I5
Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
R1 is Rejected.
R2: I1 ^ I5 ->I2
Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
R2 is Selected.
R3: I2 ^ I5 ->I1
Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
R3 is Selected.

Step 5: Generating Association Rules from Frequent Itemsets
R4: I1 ->I2 ^ I5
Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
R4 is Rejected.
R5: I2 ->I1 ^ I5
Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
R5 is Rejected.
R6: I5 ->I1 ^ I2
Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
R6 is Selected.
In this way, we have found three strong association rules.

Lab Exercise:

1. Implement the Apriori algorithm using any programming language. The program should work for given dataset. Download the mushroom dataset from UCI repository https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/ and find the frequency of white mushrooms being edible.

[OBSERVATION SPACE – LAB 6]

LAB NO.: 7 **Date:**

IMPLEMENTATION OF K-MEANS CLUSTERING ALGORITHM

Objectives:
In this lab, student will be able to:

- Understand and implement K-Means clustering algorithm

Partitioning Methods: The Principle
Given

- A data set of n objects
- K the number of clusters to form
- Organize the objects into k partitions (k<=n) where each partition represents a cluster
- The clusters are formed to optimize an objective partitioning criterion
- Objects within a cluster are similar
- Objects of different clusters are dissimilar

The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. Cluster similarity is measured regarding the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity.

The *k*-means algorithm proceeds as follows:
First, it randomly selects *k* of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges. Typically, the square-error criterion is used, defined as

$$E = \sum_{i=1}^{k} \sum_{\boldsymbol{p} \in C_i} |\boldsymbol{p} - \boldsymbol{m_i}|^2,$$

where *E* is the sum of the square error for all objects in the data set; ***p*** is the point in space representing a given object; and ***mi*** is the mean of cluster *Ci* (both ***p*** and ***mi*** are multidimensional). In other words, for each object in each cluster, the distance from the

object to its cluster center is squared, and the distances are summed. This criterion tries to make the resulting *k* clusters as compact and as separate as possible.

Algorithm:

Algorithm: *k*-**means.** The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

Example:

Consider the following data set consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set is to be grouped into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

But we cannot be sure that each input item has been assigned to the right cluster. So, we compare each item's distance to its own cluster mean and to that of the opposite cluster. And we find:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each item's distance to its own cluster mean should be smaller than the distance to the other cluster's mean (which is not the case with individual 3). Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

The iterative relocation would now continue from this new partition until no more relocations occur. However, in this example each item is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.

Also, it is possible that the k-means algorithm won't find a final solution. In this case, it would be a good idea to consider stopping the algorithm after a pre-chosen maximum of iterations.

Lab Exercises

1. Implement k-means clustering algorithm in any programming language. The code should cluster input data in csv/JSON data. Download diabetes UCI dataset and formulate positive and negative clusters. Given a random input feature and find which closet possible cluster it will fall into.

[OBSERVATION SPACE – LAB 7]

LAB NO.: 8 **Date:**

IMPLEMENT CLASSIFICATION ALGORITHM

Objectives:

In this lab, student will be able to:

- ➢ Understand the decision tree algorithms.
- ➢ Implement the algorithm(s) using any of the mentioned languages.

Introduction

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. A classification task begins with a data set in which the class assignments are known. Classifications are discrete and do not imply order. In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target.

Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown. Classification models are tested by comparing the predicted values to known target values in a set of test data. The historical data for a classification project is typically divided into two data sets: one for building the model; the other for testing the model.

Several major kinds of classification algorithms includes Decision trees(C4.5, ID3 etc), k-nearest neighbor classifier, Naive Bayes, SVM, and ANN. Generally a classification technique follows three approaches Statistical, Machine Learning and Neural Network for classification.

There are many specific decision-tree algorithms. Notable ones include: ID3 (Iterative Dichotomiser 3)

C4.5 (successor of ID3)

CART (Classification And Regression Tree) and so on..

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an

associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Decision trees have three main parts: a root node, leaf nodes and branches. The root node is the starting point of the tree, and both root and leaf nodes contain questions or criteria to be answered. Branches are arrows connecting nodes, showing the flow from question to answer. Each node typically has two or more nodes extending from it. For example, if the question in the first node requires a "yes" or "no" answer, there will be one leaf node for a "yes" response, and another node for "no."

Decision Tree Induction
Generating a decision tree form training tuples of data partition D
Algorithm : Generate_decision_tree

Input:
Data partition, D, which is a set of training tuples
and their associated class labels.
attribute_list, the set of candidate attributes.
Attribute selection method, a procedure to determine the
splitting criterion that best partitions that the data
tuples into individual classes. This criterion includes a
splitting_attribute and either a splitting point or splitting subset.

Output:
A Decision Tree

Method
create a node N;

if tuples in D are all of the same class, C then
  return N as leaf node labeled with class C;

if attribute_list is empty then
  return N as leaf node with labeled
  with majority class in D;|| majority voting

```
apply attribute_selection_method(D, attribute_list)
to find the best splitting_criterion;
label node N with splitting_criterion;

if splitting_attribute is discrete-valued and
  multiway splits allowed then  // no restricted to binary trees

attribute_list = splitting attribute; // remove splitting attribute
for each outcome j of splitting criterion

  // partition the tuples and grow subtrees for each partition
  let Dj be the set of data tuples in D satisfying outcome j; // a partition

  if Dj is empty then
    attach a leaf labeled with the majority
    class in D to node N;
  else
    attach the node returned by Generate
    decision tree(Dj, attribute list) to node N;
  end for
return N;
```

According to statistics, decision tree algorithm is one of the most widely data mining algorithms at present. But in practical application process, the existed decision tree algorithm also has many deficiencies, such as lower computational efficiency, variety bias, and so on. Therefore, is has significance in theory and reality to further improve decision tree, enhance the performance of decision tree, and to make it suit the application requirements of data mining technology.

Lab Exercises

1. Consider the spam dataset given in the link below, apply classification algorithm and compare the accuracy. Assume 80% of the dataset as training dataset and remaining as testing dataset.

   Dataset:
   http://csmining.org/index.php/spam-email-datasets-.html

2. **MINIPROJECT SYNOPSIS**

   The students should

   ➢ Submit the synopsis of the project that is supposed to be implemented using any data mining algorithm(s).

   ➢ Understand the basic concepts of algorithm(s) chosen and submit reference papers related to the existing system.

Format for the Project Synopsis

PROJECT TITLE *<Title of the project>*

<Synopsis of the project work and should be written in 3 paragraphs. The first paragraph should introduce the area of the topic and give importance of the work/topic in the present-day scenario, hence leading to the objective of the project work. The second paragraph should discuss briefly the important results that were obtained and its significance. The third paragraph should discuss the important conclusion(s) of the project work. If you have used some software tools/packages or hardware/systems, indicate them in the last line. (The abstract should fit in one page only)>

Submitted by

<name>		*<registration no>*
Instructor	:	*<place your instructor's name here>*
Course	:	*<place your course name here>*
Lab Section	:	*<place your lab section here>*
Date	:	*<place the date of submission here>*

[OBSERVATION SPACE – LAB 8]

LAB NO.: 9 **Date:**

DEVELOP AN APPLICATION USING GUI DATA MINING TOOLS

Objectives:

In this lab, student will be able to:

➢ Understand the GUI tools to perform data mining.

Introduction

Many open source GUI tools are available which involves drag/drop of various mining operators to perform data mining. Few among them are RapidMiner, Weka, Orange, Rattle etc.

RapidMiner is the Java based, open source, GUI**.** More information available at https://rapidminer.com/training/videos/

The following lists the first terms used in RapidMiner Studio.

Attribute

The information elements describing a scenario. Attributes are the table columns of a data set.

Classification

The process of predicting which category (or class) an example belongs to, based on existing data for which category membership is known. A category is defined as the possible values for a label. (Similarly, regression is the process for predicting numerical results.) That is, with classification you construct a model that, when trained, uses the learned rules to predict the category of new data.

Data set

The training set is the data used to discover predictive relationships and train models. The test set is the data used to test the accuracy and meaningfulness of a model's representation of the predictive relationship (typically discovered using the training set). The new data set is the data with missing labels; the rules derived from the training set are applied to predict outcome for the new data set. Example

Characterized by its attributes, an example has concrete values that can be compared with other examples. Examples are the table rows of a data set.

Example Set

The table created from the attributes (columns) and examples (rows). Also known as data or data set.

Label

The identifying attribute in relation to the current question. The goal is to know or learn this attribute's (the label's) value, or learn rules for deriving it from the regular attributes, for each row in the example set. Sometimes referred to as the target attribute or variable, it is the thing to predict for new examples that are not yet characterized. There can be only one label per data set.

Model

The data mining method or prediction instruction. A model explains the discovered rules and/or predicts unknown situations for current and future examples.

Operator

The building blocks, grouped by function, used to create RapidMiner processes. An operator has input and output ports; the action performed on the input ultimately leads to what is supplied to the output. Operator parameters control those actions. There are more than 1500 operators available in RapidMiner. Operators, in the Operators panel of the Design view, are both browsable and searchable.

Panel

Each view has its own set of panels, or tools, related to the view. They can be moved, sized, and hidden to suit. You can access additional panels from the View > Show Panel pull-down menu

Parameter

The setting(s) whose value(s) determine the characteristics or behavior of an operator. RapidMiner presents parameters in the Parameters panel of the Design view. There are regular parameters and expert parameters. The expert parameters are indicated by italic names and are displayed or hidden by clicking the Show/Hide advanced parameters link at the bottom of the panel.

Ports

The point through which data moves, represented by a semicircle labeled icon on the sides or operators and the Design view. See the list of port abbreviations below.

Prediction

The most probable value for a target attribute; predictions are derived by data mining. If you have rules and data, you can predict an outcome. Figure 9.1 depicts one of the prediction model.

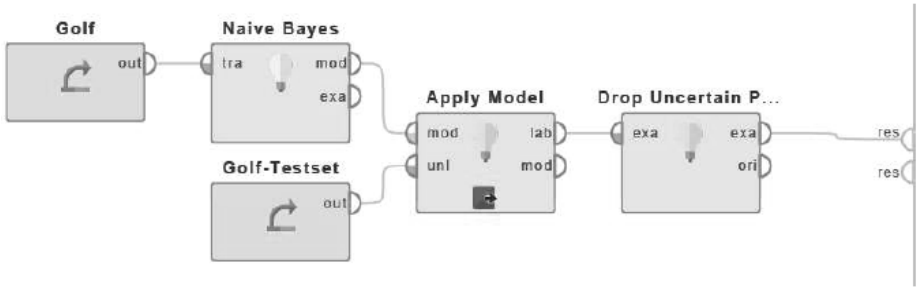


Figure 9.1 Naïve based prediction model on Golf dataset

Process

A set of interconnected operators represented by a flow design, where each operator manipulates your data. A process might, for example, load a data set, transform the data, compute a model, and apply the model to another data set.

Process view

The working area for building processes. This is the canvas in the Design view where you drag operators or where, when you double-click a process, the operators of that process appear.

Repository

The storage mechanism for data and RapidMiner processes. Best practice recommends you use the repository for data storage instead of reading directly from a file or database.

Data visualization:

RapidMiner already provides several plotters in order to visualize the properties of a data set. Each time an ExampleSet is part of the (intermediate) result, the user can select between data view, statistics view and charts view. The charts view provides several 2D and 3D charts, histogram charts, and other charts for high-dimensional data sets like survey or parallel charts.

Weka is the also a Java based, open source GUI data mining tool. Weka is a collection machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

It is also well-suited for developing new machine learning schemes. Weka stands for the Waikato Environment for Knowledge Analysis, which was developed at the University of Waikato in New Zealand. WEKA is extensible and has become a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java and runs on almost every platform. WEKA is easy to use and to be applied at several different levels. You can access the WEKA class library from your own Java Program, and implemented new machine learning algorithms. There are three major implemented schemes, WEKA.

1. Implemented schemes for classification,
2. Implemented schemes for numeric prediction,
3. Implemented "mete-schemes".

Besides actual learning schemes, WEKA also contains a large variety tools that can be used for pre-processing datasets, so that you can focus on your algorithm without considering too much details as reading the data from files, implementing filtering algorithm and providing code to evaluate the results.

The GUI Chooser consists of four buttons-one for each of four major Weka applications and four for menus.

The buttons can be used to start the following applications:

- **Explorer** An environment for exploring data with WEKA.
- **Experimenter** An environment for supporting experiments and conducting statistical tests between learning schemes.
- **Knowledge Flow** this environment supports essentially the same functions as the Explorer but with a drag and drop interface. One advantage is that is supports incremental learning.
- **Simple CLI** Provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface.

To perform decision tree classification follow below steps:

1. Open Weka GUI Chooser.
2. Select EXPLORER present in Applications.
3. Select Preprocess Tab.
4. Go to OPEN file and browse the file that is already stored in the system "bank.csv".
5. Go to Classify tab.
6. Here the c4.5 algorithm has been chosen which is entitled as j48 in Java and can be selected by clicking the button choose
7. and select tree j48
8. Select Test options "Use training set"
9. if need select attribute.
10. Click Start.
11. Now we can see the output details in the Classifier output.
12. Right click on the result list and select "visualize tree" option.

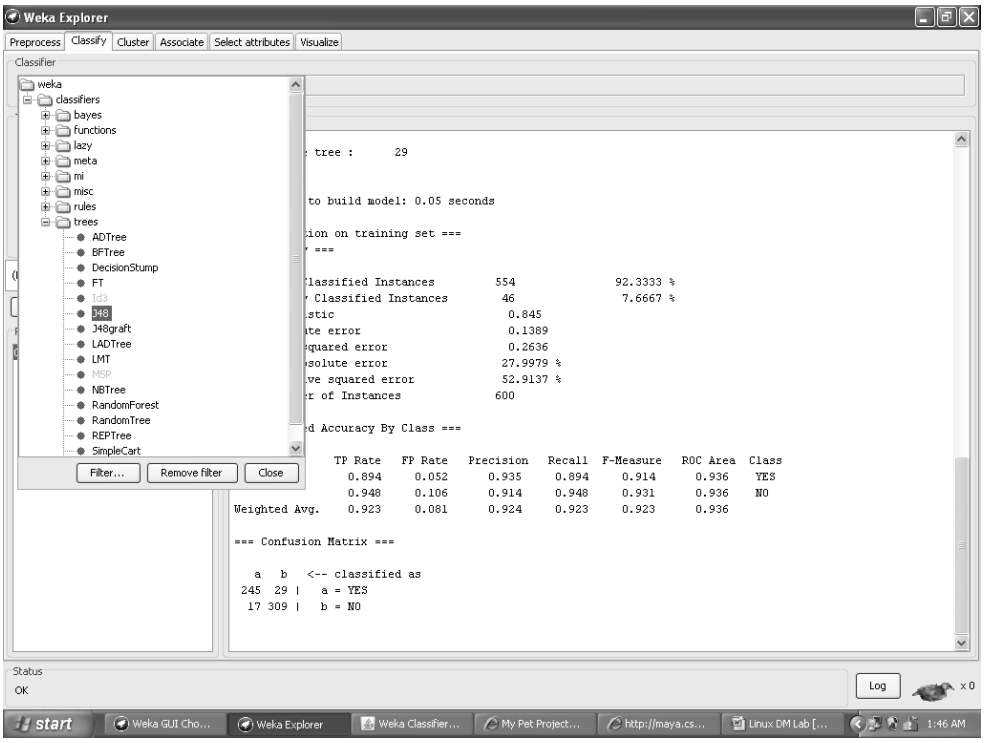


Figure 9.1 Sample output

Lab Exercises

1. Implement k-means clustering algorithm and decision tree algorithm in Rapid miner and Weka. You may use the same dataset as mentioned in lab 7 & 8. Record the model (along with the operators) used to perform the mining. Also write the sample output.

References

http://www.cs.utexas.edu/users/ml/tutorials/Weka-tut/
https://www.ibm.com/developerworks/library/os-weka1/index.html.

[OBSERVATION SPACE – LAB 9]

LAB NO.: 10 **Date:**

DATA COLLECTION & ANALYSIS FOR THE MINIPROJECT

Objectives:

In this lab, student will be able to

- Collect the data relevant to the mini-project and perform cleansing or preprocessing on the data.

Lab Exercise

1. Collect or create relevant dataset. Apply preprocessing on the dataset as mentioned in Lab 04. Decide on the suitable mining algorithm. Submit the report justifying the preprocessing techniques that are used along with the selected design model.

[OBSERVATION SPACE – LAB 10]

LAB NO.: 11 **Date:**

IMPLEMENTATION OF THE MINIPROJECT

Objective:

In this lab students should be able to:

➢ Understand working in team and integration of modules.

Introduction

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of several analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Data mining is primarily used today by companies with a strong consumer focus - retail, financial, communication, and marketing organizations. It enables these companies to determine relationships among "internal" factors such as price, product positioning, or staff skills, and "external" factors such as economic indicators, competition, and customer demographics. And, it enables them to determine the impact on sales, customer satisfaction, and corporate profits. Finally, it enables them to "drill down" into summary information to view detail transactional data.

With data mining, a retailer could use point-of-sale records of customer purchases to send targeted promotions based on an individual's purchase history. By mining demographic data from comment or warranty cards, the retailer could develop products and promotions to appeal to specific customer segments.

For example, Blockbuster Entertainment mines its video rental history database to recommend rentals to individual customers. American Express can suggest products to its cardholders based on analysis of their monthly expenditures.

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical,

machine learning, and neural networks. Generally, any of four types of relationships are sought:

- **Class:** Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.
- **Clusters:** Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.
- **Associations:** Data can be mined to identify associations. The beer-diaper example is an example of associative mining.
- **Sequential patterns:** Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Data mining consists of five major elements:

- Extract, transform, and load transaction data onto the data warehouse system.
- Store and manage the data in a multidimensional database system.
- Provide data access to business analysts and information technology professionals.
- Analyze the data by application software.
- Present the data in a useful format, such as a graph or table.

Lab Exercise

1. Depending on the input dataset choose appropriate mining techniques. Implemented the project and explain its working.

LAB NO.: 12 **Date:**

MINIPROJECT RESULT ANALYSIS

Objective:

In this lab students should be able to:

➢ Understand plotting of results, cross validation techniques.

Introduction

Validation is the process of assessing how well your mining models perform against real data. It is important that you validate your mining models by understanding their quality and characteristics before you deploy them into a production environment.

Tools for Testing and Validation of Mining Models

Analysis Services supports multiple approaches to validation of data mining solutions, supporting all phases of the data mining test methodology.

- Partitioning data into testing and training sets.
- Filtering models to train and test different combinations of the same source data.
- Measuring lift and gain. A lift chart is a method of visualizing the improvement that you get from using a data mining model, when you compare it to random guessing.
- Performing cross-validation of data sets
- Generating classification matrices. These charts sort good and bad guesses into a table so that you can quickly and easily gauge how accurately the model predicts the target value.
- Creating scatter plots to assess the fit of a regression formula.
- Creating profit charts that associate financial gain or costs with the use of a mining model, so that you can assess the value of the recommendations. These metrics do not aim to answer the question of whether the data mining model answers your business question; rather, these metrics provide objective measurements that you can use to assess the reliability of your data for predictive analytics, and to guide your decision of whether to use a particular iterate on the development process.

  Among above methods, cross-validation is a standard tool in analytics and is an important feature for helping you develop and fine-tune data mining models. You

use cross-validation after you have created a mining structure and related mining models to ascertain the validity of the model. Cross-validation has the following applications:

- Validating the robustness of a particular mining model.
- Evaluating multiple models from a single statement.
- Building multiple models and then identifying the best model based on statistics.

For each type of mining model, clustered or non-clustered, the stored procedures perform cross-validation in two separate phases.

Partition data and generate metrics for partitions

For the first phase, you call a system stored procedure that creates as many partitions as you specify within the data set, and returns accuracy results for each partition. For each metric, Analysis Services then calculates the mean and standard deviation for the partitions. OR confusion matrix can be generated for each partition. A combined confusion matrix can be used to compute F1 score and the accuracy.

Generate metrics for entire data set

In the second phase, you call a different set of stored procedures. These stored procedures do not partition the data set, but generate accuracy results for the specified data set as a whole. If you have already partitioned and processed a mining structure, you can call this second set of stored procedures to get just the results. Example: 25% of input dataset can be used for testing and 75% for training.

Lab Exercise

1. Plot the results and save as image files. Formulate the table depicting the comparison with existing method in terms of accuracy, time and space complexity. Apply cross validation and compute the accuracy with various data mining approaches.

REFERENCES

1. Jiawei Han, Micheline Kamber and Jian Pei, “Data Mining: Concepts and Techniques, 3rd Edition”, Morgan Kaufmann, June 2011
2. Ian H. Witten and Frank Eibe, “Data mining : practical machine learning tools and techniques with Java implementations”, Morgan Kaufmann, 1999
3. Usama Fayyad et al. (eds), “Advances in Knowledge Discovery and Data Mining”, AAAI/MIT Press, 1996