A Curriculum Development Project for IBM Linux in Academia Program

Chih-Yang Tsai, Andrew Pletch
School of Business, Department of Computer Science
State University of New York at New Paltz, New York, USA
tsaic, pletcha@newpaltz.edu
Arthur Palmiotti, Gerri Peper, Michael Wuest, Chris Rohrbach, Kevin Curley
IBM, USA
apalmiot, gpeper, mwuest, rohrbach, curley@us.ibm.com

Abstract

This paper introduces the IBM Linux in Academia program and a curriculum development project initiated by the authors for the program. The service of IBM Linux in Academia program is based on the Linux virtual service concept in which a mainframe computer is partitioned into many Linux images supported by IBM's Virtual Machine Operating System. On the IBM S/390 system, each image acts as an independent Linux server. This free service saves the acquisition and management cost of running multiple physically separated servers for participating universities. The curriculum development project intends to create and share curriculum materials for e-Business related courses among participants. The main IBM software used in this project includes DB2 Universal Database and WebSphere. The main objective in the first stage of this project is to develop a data warehouse generator to manipulate a large read-only database obtained from a real world health care application supplied by IBM. Through a web based user interface, an instructor could flexibly create a data warehouse using the Account Data Model developed by some of the authors from the read-only database with the desirable size and attributes to support pedagogical needs. Other aspects of the project are also addressed.

key words:Account Data Model, Data Warehousing, DB2, Linux, WebSphere

1 Introduction

The objective of the IBM Linux in Academia program is to provide a robust and reliable Linux environment for the university community dedicated to educational and research purposes [2]. The service provided by the program is hosted on an IBM S/390 mainframe computer located at Colorado State University, capable of running hundreds of Linux images (servers) on the same hub server at the same time. The hub server partitions the necessary processing, storage and network capacity for each Linux image. As a result, resources can be allocated flexibly to meet the individual need of an image while maintaining the same level of separation between images as any physically separated servers would. A commercial service based on the same concept was launched by IBM recently [6]. The Linux in Academia program allows a participating university free accesses to not only IBM software packages but also virtual servers to host those applications. The major benefit to a participating educational institution from this arrangement is that the institution could offer its instructors and students accesses to the state-of-the-art information technologies without heavy investment in IT infrastructure. The service is especially valuable to colleges without adequate financial and human resources to acquire and manage the hardware and software required to support their IT related curricula. In addition to the common open source packages for Linux, the two major IBM software components included in this service are DB2 Universal Database and WebSphere.

Despite the abundant resources provided by the Linux in Academia program, the service does not include instructional resources such as course materials and sample databases. This might not pose a serious problem for a Computer Science program which may require students to build software and network applications on a Linux image from scratch. As the need of a business program, MBA in particular, focuses more on utilizing existing databases for the purpose of training students on analytical skills (marketing research, customer relationship management) and managerial concepts (e-Business strategies), there is a demand for instructional materials, especially those supported by a user friendly interface for students without programming experience. The Business School at the State University of New York at New Paltz (hereafter referred to as SUNY-NP) signed a collaboration agreement with IBM on October 26, 2001, to establish an e-Business Virtual Lab using the service from the Linux in Academia program. As the first business program participating in the Academia program, the authors propose a project to develop course materials for the program. We hope the instructional resources developed from the project could encourage more colleges to join the Linux in Academia program which in turn helps the program develop and test more contents.

The core of the project is to develop a framework

and all associated software so that an instructor could easily extract and build a customized data warehouse from a large read-only database (57 GBytes) for his or her instructional need on the instructor's own Linux image. This read-only database is structured using a popular data model for data warehouses, namely the star schema. As a result, extracting a small portion from the large database is relatively easy. However, we would like to take this opportunity to test a new data model, Account Data Model (see [7] for details), as the data model for warehousing purposes due to its flexibility. The extracted data warehouse is created from the dimensions and attribute values chosen by the instructor. For example, the instructor may select a shorter period of time or a focused subset of geographical regions to control the size of the resulting data warehouse. The instructor has full privilege to the resulting database and could control the privileges rendered to his/her students. The read-only database is provided by IBM from a real world application in health care (medical insurance) industry with all personal and institutional identities removed or disguised. The size of the extracted data warehouse on the instructor's image is targeted at 2 GBytes, which we believe can still give students a flavor of a realistic data warehouse with considerable size. As transmitting 2 GByte data on the Internet can be very time consuming, the Academia program provides a perfect platform for us to implement this project. Although the read-only database is installed on a Linux image which is logically separated and independent from the instructor's image, both are physically located on the same S/390 system. This configuration significantly reduces the download time. Using a web based GUI interface, the process of creating the instructional data warehouse is reduced to clicking on the desired attribute values presented in a tree structure similar to a file/directory navigating system such as Windows Explorer. Additional web based user interface for students to access the data warehouse on their instructor's image is also intended for reducing the level of computer skills needed to query the database. Those user interfaces will be built on common open source software and be organized and managed by IBM's WebSphere.

The following sections provide more detailed introduction and descriptions of the infrastructure of the Academia program and our plan of supporting this service with course contents developed from this curriculum development project.

2 Infrastructure of IBM Linux in Academia Program

The service of the Linux in Academia program is based on the "utility computing" concept advocated by major UNIX server vendors including IBM, HP, and Sun Microsystems. They argue that the computing resources do not necessarily need to reside in the user's

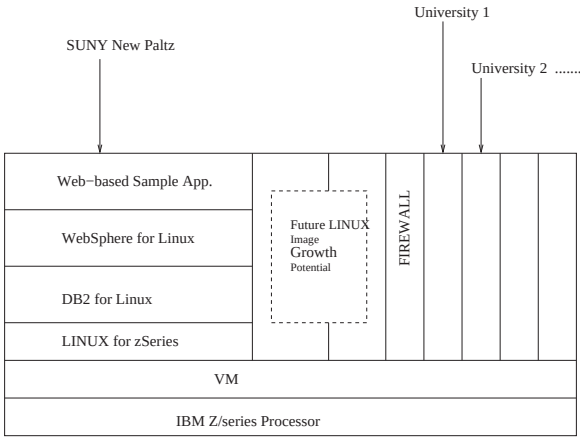


Figure 1: Linux Images on S390

organization. Instead they can be outsourced to a remote server managed by a vendor to save management costs on maintaining system hardware, software, security, and reliability, etc. The organization only needs to maintain lean clients to tap into the utility system. The Academia program has setup its Linux hub at Colorado State University on an IBM S/390 mainframe running a Virtual Machine (VM) operating system which supports the partition of system resources to host many virtual servers (images) each running a Linux (for S/390) operating system. Figure 1 illustrates the architecture of the system. The two layers at the bottom are the zSeries processor and the VM operating system for the S/390. The top four layers on the left demonstrates a sample Linux image created for SUNY-NP. At the foundation of this image is a SuSE version of Linux for S/390 and the middleware includes a DB2 Universal Database Server and a WebSphere Server on which web based applications (top layer) can be built. The other side shows potential future images. These images act like independent servers protected by their own firewalls. For a Computer Networking course, each student may obtain an image of his/her own with small disk space. For an e-Business course offered by a business school to students with limited computer experience, the whole class may share an image having much bigger memory and disk space than those images created for individuals. Thus, an image can be created to serve the need of a course, a user, or an application area with the desired software packages, computing power and storage space.

Accesses to the Linux images are provided via three channels as listed below.

- An end user either on or off campus can connect to the campus LAN which in turn connects to the server through a secure VPN session.
- A user could also access the image directly through an internet Secured Shell (SSH) connection.

- A user can use a browser to access a pass word protected web sites as a gateway to the resources and applications on various Linux images.

Software packages such as Putty and WinSCP2 are ideal tool to access Linux images through a secured shell (SSH). Both programs are small and do not require installation. This is particularly convenient for students to carry them in a floppy disk and work on different locations. However, the easiest way to access an application for non-computer savvies on a Linux image is to do it through a web browser which is available almost everywhere. As a result, this is the approach we recommend for our business students.

3 Potential Curriculum Supports

There are many areas and courses that can utilize the service and its contents. We discuss a few potential areas in this section.

Data Analysis: In courses like Marketing Research and Decision Support Systems, students can access a read-only database created by their instructors and retrieve necessary data through a browser on selected dimensions and ranges of attribute values. The interface translates the user's request into a SQL SELECT statement and connect to the data warehouse to get the data which is saved as a file in the user's account. The student can then download the resulting data set to local machines for further analysis. This application is also suitable for covering course subjects such as data mining and customer relationship management. The instructor can observe and discuss the different viewpoints and conclusions generated by students from the same database and demonstrate the benefit obtained from such analysis.

Virtual Company: Students create their own small databases with full access privilege to store product, customer, and order information for their virtual companies. This exercise involves students from the design phase of setting up an information system for a company. Students are asked to consider the issues of how their designs can help the organization achieve its strategic goals and manage daily operations. The instructor can address the importance of design issues related to a MIS project and how a well designed information system can facilitate business decision processes, which in turn enhance an organization's competitive advantages.

Virtual Community: The first two applications focus more on the database issue which is the building block for other e-Business applications. The virtual community application extends the database built earlier to create an online community by connecting the virtual companies from various students together. For example, a manufacturer wants to purchase materials online from its supplier and a bank handles the payment between the manufacturer and the supplier. Students can team up with Computer Science majors on these web application projects to develop their first e-Business practice, which handles online transactions and update their databases. This service can be provided either by the Apache web server with CGI supports or from a fully integrated software for e-Business such as IBM WebSphere.

One concern we have at this point is the internet traffic during peak hours. During certain time of the day, the connection to the hub may be much slower. In that case, the instructor may consider installing a local copy of the application to support instructional purposes. For example, the instructor can further extract a 2 MByte data warehouse from the 2 GByte data warehouse on his/her image to a local Linux server (DB2 is available through IBM Scholar program for local installation). Students may practice and prototype their work on the local server during internet peak hours and submit their pretested queries to the 2 GByte warehouse for their project later.

4 The Medical Insurance Database

In this section, we introduce the Medical Insurance database used in this project. This database is a database used by IBM for software testing purposes. It is obtained from a real practice at a health insurance company. All sensitive information is either removed or disguised in this database. The database is structured under a popular data model for data warehouses, namely, the *star schema*. As a result, understanding the meaning of attributes and writing queries against it are relatively easy. The data warehouse only contains one year medical insurance claim data which already consumes 57 GByte hard disk space on a Linux image.

At the center of this data warehouse sits a fact table, INS700TB, which records detailed insurance claim history. The metrics in this fact table are dollar amount and frequencies of insurance claims, including measures such as payment amount, deductible amount, coinsurance amount, third party payment amount, and claim utilization counts, etc. This table has 228,221,751 rows. There are 11 dimension tables as follows.

1. INS002TB: Each row in this table represents an individual who enrolled in this insurance policy (1,428,826 rows).
2. INS800TB: This table stores 15,186 different diagnosis categories (15,186 rows).
3. INS801TB: Diagnosis are also classified into 512 diagnosis related groups (512 rows).
4. INS804TB: There are 19 different reasons for discharging a patient from a hospital stay (19 rows).
5. INS805TB: This is a list of states where the patient resides. It covers all fifty states of the United States (50 rows).
6. INS806TB: All claims are classified into 5 different types (5 rows).

7. INS808TB: This table lists the year when services are provided. Since we only have one-year data, this dimension table has no effect on any query results.
8. INS809TB: This table records all month and year combination during the one-year period (12 rows).
9. INS810TB: This table records the month/date/year and month/year combinations for the recorded period (365 rows).
10. INS811TB: There are 4275 different medical procedures (4275 rows).
11. INS812TB: This table records the provider's locations which include the 50 US states, the identification and short name of the state, and some foreign countries or regions (132 rows).

In addition to the main fact table, there are 5 summary tables with higher level of aggregation. Therefore, their sizes are much smaller.

1. INS750TB: This table summarizes the utilization of the insurance policy by patient state, year, month, and claim types (2998 rows).
2. INS751TB: This is a summary of utilization by providers, provider state, year, month, and claim types (2998 rows).
3. INS752TB: This is a summary of service grouped by diagnosis, diagnosis related group, patient state, and primary procedure performed, year, month, and claim types (7,811,520 rows).
4. INS752TB: This is a summary by claim type and year (5 rows).
5. INS756TB: This is a summary by provider state and year. It has 51 rows, one for each of the 50 states and the remaining one for the sum of all non-US countries and areas.

The number of dimensions for those summary tables is also smaller, ranging from two to seven. Due to the fact that we are only given one year data, the year dimension (INS808TB) is irrelevant in this case. Nevertheless, we still retain this dimension to preserve the overall structure of the data warehouse. The main fact table, INS700TB is the most detailed one which has all dimension tables except for INS808TB as its star dimensions. Summary table INS750 has four dimension tables including

1. INS805TB (50)
2. INS806TB (5)
3. INS808TB (1)
4. INS809TB (12)

Numbers in the parentheses indicate the number of rows in each table. Figure 2 shows the dot model for summary table INS751TB. The dot model is centered at table INS751TB, its fact table, and four dimension tables INS806TB, INS808TB, INS809TB, and INS812TB, which maintain one-to-many relationship with the fact table. Summary table INS752TB (7,811,520) has seven dimension tables as shown below.

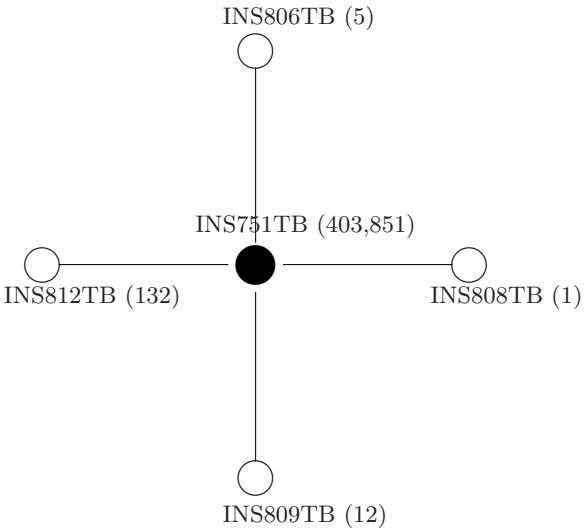


Figure 2: Dot Model for INS751TB

1. INS800TB (15,186)
2. INS801TB (512)
3. INS805TB (50)
4. INS806TB (6)
5. INS808 (1)
6. INS809TB (12)
7. INS811TB (4,275)

Summary table INS853TB (5) is joined by two dimension tables INS808TB (6) and INS808TB (5). Tables INS808TB (1) and INS812TB (132) are the dimension tables for INS756TB (51) which summarizes the claim amount and frequencies by providers' locations with all non-US locations aggregated into one row.

Those tables and their indices take about 57 GByte of disk space. Table INS700TB alone occupies more than half of that space and due to its size, has to be separated into 30 containers. We also observe that the design of this data warehouse follows a practice suggested by some practitioners, i.e., having a main fact table accompanied by a half dozen summary tables [8] to shorten response time on frequently requested queries about highly aggregated information.

5 The Account Data Model

In this section, we briefly introduce the Account Data Model (ADM) used in our data warehousing component of the project. Detailed discussion of the model can be found in Pletch, *et al.* [7]. In most of database implementations, the transaction support

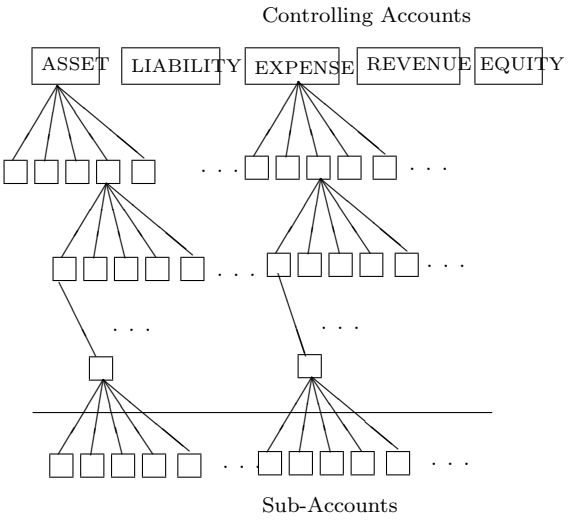


Figure 3: An Account Hierarchy

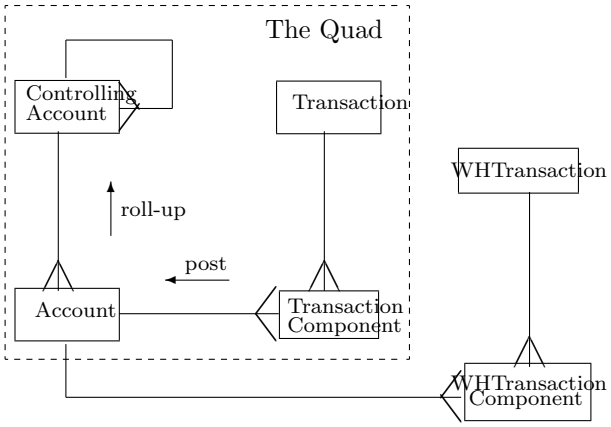


Figure 4: The Quad with Data Warehouse Extension

database and decision support database (data warehouse) are built based on separate data models. The most popular data model for the former is the Entity Relationship (ER) Model [1] and the latter usually adopts a dimensional model with a star schema or a snowflake schema [5]. In our database design, the transactional and warehousing databases are integrated using the same data model, the ADM. From a data warehousing perspective, this data model can be viewed as a special case of a snowflake schema in which there is only one table joined to the fact table that specifies the type of account each row of the fact table belongs to. It uses a hierarchy of accounts to model the information found in an organization's transactional as well as decision support processing. Most of this hierarchy follows standard double-entry accounting practices, i.e. a set of accounts organized in sub-trees rooted in five major accounting categories, namely ASSET, LIABILITY, EQUITY, REVENUE, and EXPENSES. The leaf nodes in Figure 3 represented by boxes below the solid line are called sub-accounts. They are the only accounts which actually participate in transactions and are the ones that link to the fact table from a warehousing perspective. Boxes above the solid line (referred to as controlling accounts) form a type hierarchy and are used as summary accounts, summarizing the activity of the accounts below them in the hierarchy.

Following accounting practices, a *transaction* is a collection of components that document either debit or credit operations against accounts. Transactions should be *balanced* in that every crediting component is offset by a corresponding debiting component. Ensuring that transactions are balanced before they are accepted pushes some of the data cleansing activity typical of the migration of data from an ER-based database to a data warehouse into the data entry phase. However, unlike the ER model, the design of the hierarchical account structure is less intuitive and requires better understanding of the business side of the organization. Nevertheless, a well designed account hierarchy serves as a good managerial tool for companies wanting to practice the principles of the Balanced Scorecard [4] which requires measuring performances from four interrelated categories. This is due to the fact that the design process forces a designer to create accounts which could also represent performance measures. In addition, the double entry system guarantees that an activity is viewed at least from two perspectives. The two entries can also be used to track the links or causal relationship between two performance measures. The pain of designing the hierarchy can somewhat be eased by adopting the standard charts of accounts developed by specific industries.

Part of the justification of our claim that an ADM implementation can be used as a data warehouse is our confidence in the quality of the data stored due to the balanced-transaction nature of the model. Transactions and transaction components along with accounts and controlling accounts eventually come to reside in a standard four-table structure as shown in the dashed box of Figure 4. We refer to this table structure as the *Quad*. Each transaction involves a row in the transaction table and several rows in the transaction component table. A typical transaction can be better described using the following example.

Example 5.1 *A customer, XYZ, orders four units of widget at the unit price of $15 and unit cost of $10 on July 15, 2002.*

The event in Example 5.1 generates the following accounting entries.

$$\begin{cases} \text{(a) Account Receivable–XYZ Co.} & \$60 & \\ \quad \text{(b) Sales–XYZ Co.} & & \$60 \\ \text{(c) Cost of good Sold–widget} & \$40 & \\ \quad \text{(d) Inventory–widget} & & \$40 \end{cases} \quad (1)$$

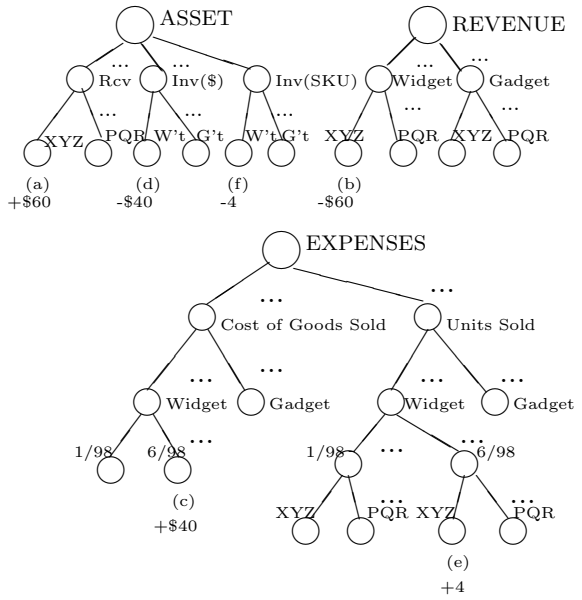


Figure 5: The Components in Accounting Trees

In addition, two inventory entries are also recorded for this event, one debited four units of widgets into customer XYZ's account and the other credited four units from the inventory account of widgets as shown in the following entries.

$$\begin{cases} \text{(e) Units sold –XYZ Co./widget} & 4 \\ \text{(f) Inventory–widget} & 4 \end{cases} \quad (2)$$

One row is added to the Transaction table in Figure 4 to represent this transaction. The six accounting entries correspond to six transaction components, half of them debiting accounts and the other half crediting accounts. Hence, six rows are added to the Transaction Component table of Figure 4.

At some point after a transaction has been entered, it undergoes a process called *posting* which updates the sub-accounts in the account table affected by the new transaction. These modifications are further propagated to the controlling accounts (rows in the Controlling Account table) higher up the account hierarchy from the affected sub-accounts by a process called *roll-up*. Both these activities are indicated diagramatically in Figure 4. When these components are posted, we see the changes in Figure 5 to the balances of the six affected sub-accounts. Note that the negative values under inventory and inventory SKU (Stock Keeping Units) in the figure come from debiting a credit account (or crediting a debit account). Both posting and roll-up become part of the model through automated means - stored procedures, triggers and the like.

The ADM does not support modifying or deleting an existing Quad transaction directly. When a business activity, a sales order for example, is later modified the ADM implements this modification by adding a totally new transaction. The net effect of combining the components in the new transaction and the original transaction produces new account balances. Thus the original transaction and a corresponding modifying transaction provide a history of the ordering process. From a process point of view, every possible change to an account must be kept for the purpose of process control.

The two tables not in the dashed box of Figure 4 are used specifically for warehousing (decision support) purposes, the WHTransaction and WHTransaction Component tables share exactly the same format as their transactional support counter parts, Transaction table and Transaction Component table. As we use the same data model for the two, they share the same Account and Controlling Account tables. In fact, some of the performance measures from the Financial category of a balanced scorecard may have already been built into the Quad in the transactional support part of the Quad while the performance measures from the remaining three categories (customers, internal business process, learning and growth) are generated for decision support purposes. The warehouse version of the Transaction Component table may choose to keep only the net result of a transaction after it has been fulfilled.

The ADM shows its true strength in its ability to add new information about any transactional and warehousing applications without the need for table redesign. This is achieved by adding new accounts and sub-accounts for the new attributes and entities and then, for each type of affected transaction, additional transaction components are generated to support debiting or crediting the new accounts. In such cases we add more rows to existing tables; never more columns nor more tables. Indeed in the current implementation, transactions saved under an old design schema, which is then redesigned to affect different sub-accounts, will automatically reflect the new design simply by being brought into the user-interface and saved again without change. This is particularly useful if changes of design are needed to reflect new business processes or performance measures. What makes the ADM a viable data warehousing model is the fact that all transactional and decision support components are stored in the same table structures. Adding new types of transactions or even modifying the design of existing transactions becomes a data-entry activity rather than a design endeavor. Thus the line between the transactional database and data warehouse becomes blurry.

6 The Data Warehousing Component of the Project

Universities throughout the world often need access to large databases for various pedagogical reasons. In educational programs such as Computer Science where students are simply learning to use SQL or where query

performance or execution are the issues, the kind of data in the database is of little concern. Benchmark databases such as TPC [9] are useful in those situations. Downloading the benchmark database generator written in C/C++ takes almost no time. The generator could install randomly generated databases of various sizes by manipulating the values of some control parameters. On the other hand, in educational programs where the students are more interested in the query results and what they mean, students in a business program for example, it is very difficult to make any real-world sense of the query results from randomly generated databases because these databases contain information that is only marginally realistic. It is nevertheless not reasonable to provide a real database for download, even one of moderate size (a couple of gigabytes) because the download time over the Internet would be unacceptable. The IBM Linux in Academia computing hub provides us with an opportunity to make available to universities a realistic database of considerable size and avoid the typical download time problems. The medical insurance database labelled *insurdb* in Figure 6 plays this role. In order to keep maintenance to a minimum, this install should be read-only.

Although it is possible to allow students to query *insurdb* directly, the result set is usually too large for them to download to their local computers for further analysis using other software packages. In addition, an ill structured query may take hours before a result can be produced. Therefore, we would like to give an instructor control of the size of a customized data warehouse. This project component gives instructors necessary user interface to build their own data warehouses from *insurdb* with the size they deem reasonable for their classes. Since a certain amount of customization would probably be needed we envisage each university/program having its own *Linux* kernel and own DB2 installed on the same S/390 where the *insurdb* database is installed and that extracts of some size (up to 1 or 2 GBytes) could realistically be made from the *insurdb* database into these user-controlled databases, labelled *WHQuad 1* through *WHQuad n* in Figure 6.

We show how SUNY-NP could use the Linux virtual service environment to support business intelligence course taken by students in our business program and how our own work could be packaged so as to allow other universities to replicate our activities with their own students. We envisage a package that would make it possible for many different users/universities to produce local extracts from the *insurdb* to copies of DB2 running on their own images located on the same S/390 system and into databases we have labelled *WHQuad 1* through *WHQuad n* in Figure 6 using the Account Data Model.

An instructor at university *i* could specify the desired time range, dimensions, and range of attribute values to extract a small portion of *insurdb* and build a customized data warehouse *WHQuad i*. The extract resides in databases controlled by the instructor. In most cases, the instructor may want to make the *WHQuad i* data warehouse read-only to students. The *WHQuad i* databases, although smaller than the *insurdb* database, would still be too large for downloading onto machines that are local to the final users. However they could be used to produce much smaller exports - tables on the order of 2000 to 20000 rows - which would be small enough to download onto local machines (either as files or into a local copy of *db2*) - and it is our intention that these latter tables, looking much like the *fact* tables in a *star-schema* data warehouse but much smaller, could easily be analyzed using local OLAP, Statistics and Data Mining software which, at this time, we do not envisage providing to the user.

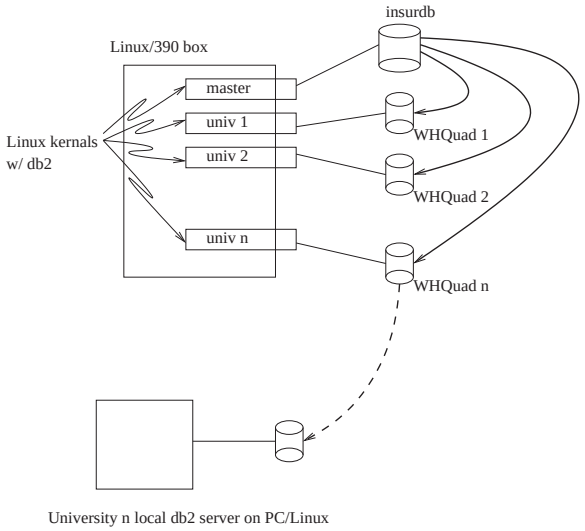


Figure 6: DB2 Structure

Key elements of the package are listed below and explained in Sections 6.1 and 6.2.

- WHQuad Create : This script will create the initially empty WHQuad database tables and indices and install various stored procedures and triggers.
- WHQuad Import : This program will generate insert statements for the tables in the *WHQuad i* database. The program will proceed from a suitably prepared configuration file and extract data from the *insurdb* database.
- WHQuad Cube Build : This program provide the user with a web-based GUI interface from which the user can select *dimensions* and *performance indicators* of interest and build a data cube (typically 2000 to 20000 rows) suitable for analysis using off-the-shelf OLAP tools.
- WHQuad Export : This program will export mini-cubes created using the Cube Build software to the user's local copy of *db2*. Although this step is not essential and involves installation and maintenance of a second (local) copy of *db2* by the user,

it will serve as a backup in case the access to the hub server is interrupted.

6.1 The WHQuad Databases

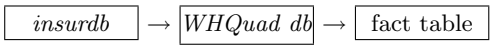
We use the Account Data Model (ADM) [7] to build the data warehouse *WHQuad i*, $i = 1, \ldots, n$, due to the fact that ADM is generic and flexible. Once a Linux image and DB2 are available for a university, the *WHQuad Create* program described earlier can be used to install a fresh copy of the *WHQuad* database. This database can be populated with a subset of the information in the *insurdb* database using the *WHQuad Import* program.

A web-based GUI interface, program *WHQuad Cube Build* described in [7], can now be used to extract information from the *WHQuad* tables into mini-cubes or fact tables. A number of OLAP tools such as *Brio* have been developed precisely to present the information in such fact tables.

At this point, the user is able to extract several fact tables containing different dimension/performance indicator configurations. These could be prepared in advance and given to students as source information for data analysis exercises. It is also possible that these *fact tables* could be exported to local copies of DB2 running at the user's university, which we describe next. We should also point out that since *insurdb* adopts a star schema, creating a smaller data warehouse on instructors' images with exactly the same star schema should be straight forward. Thus, we do not include it as a component in this project.

6.2 Local Installation of *db2*

We expect the size reduction from



to be in orders of magnitude in general. The *insurdb* database is 57 gigabytes, the *WHQuad* databases as large as a couple of gigabytes but usually smaller than 2 gigabyes and the *fact tables* generated by the *WHQuad Cube Build* software could be as big as 2 megabytes.

Only the latter tables are small enough to be realistically downloaded to machines local to the participating universities. They are just on the edge of being too large to be used on an *ad hoc* basis so we feel the appropriate thing to do is to download them once and use the local copy for conducting classroom activity when the connection to the Linux image is either interrupted or unbearably slow. The program *WHQuad Export* would accomplish this task. It is also possible to use IBM data migration software to do this.

7 Other Components of the Project

Next to the applications surrounding DB2 databases is the development of WebSphere applications. There are several application areas that WebSphere can play a major role.

1. As mentioned earlier, all tools developed for an instructor to generate a data warehouse from the medical insurance database are web based applications. Java Servlets provides a good tool for database intensive applications.
2. The target users of this service are business students who usually do not have profound knowledge in Linux operating system and programming. In order to shield the technical details from the students, a web interface supported by WebSphere as the middleware allows students to focus on analytical and business issues rather than struggling with programming details.
3. We would also like to create some templates for student to easily deploy a virtual company with online transaction capability. This will first link to the database created by students to operate their e-store. Step by step, it can be developed into an online trading community where there are buyers, sellers and transaction facilitating media such as banks and shippers.

The third component in this project covers the area of group decision support, which is a key element in knowledge management. An open source, XML-based software, Jabber [3], can be used as a building block for creating and testing a group decision support environment. Other miscellaneous supporting tasks like developing tools to assist instructors managing student accounts are also intended. This includes programs for setting up the accounts, granting appropriate privileges, and monitoring usage.

8 Conclusion

In this paper, we introduce the service offered by the IBM Linux in Academia program and describe a project designed to take advantages of the service provided by the program. The objective of the project is to develop instructional resources including a data warehouse generator using a web based user interface and the Account Data Model. The software allows instructors to carve out a portion of a huge database and create a customized data warehouse for their instructional needs. The data warehouse generator along with other web based resources are designed to facilitate the offering of e-Business and knowledge management related courses in a business program. We hope the software and tools produced from this project can help a business program enhance its capacity and capability to train students on both strategic and analytical issues related to e-Business and knowledge management. We would also like to invite participation from other institutions to contribute to the process of developing and testing more instructional resources.

References

[1] Chen, P. P., "The Entity-Relationship Model: toward a Unified view of data", *ACM Trans. on Database Systems*, Vol. 1, 1, 1976, 9-36.

[2] CSU/IBM Linux Hub Home Page, http://204.146.24.13

[3] Jabber Software Foundation, http://www.jabber.org

[4] Kaplan, Robert S., Norton, David P., *The Balanced Scorecard - Translating strategy into Action*, Harvard Business School Press, Boston, 1996.

[5] Kimball, Ralph, A Dimensional Modeling Manifesto, *DBMS*, vol. 10, no. 9, 1997, pp. 58-72.

[6] Olavsrud, Thor, "IBM Offers Processing Power As Utility", *InternetNews*, July 1, 2002.

[7] Pletch, A., Tsai, C-Y., and Matula, C., "The Account Data Model", in S. Spaccapietra, S.T. March, Y. Kambayashi, (Eds) *Lecture Notes in Computer Science* for the *21st International Conference on Conceptual Modeling (ER2002)*, October 7-11, 2002, Tampere, Finland, Vol 2503, pp. 263-275.

[8] Todman, Chris, *Desiging a Data Warehouse– Supporting Customer Relationship Management*, Prentice Hall PTR, New Jersey, 2001.

[9] Transaction Processing Performance Council, http://www.tpc.org.