A Syllabus in Data Warehousing

Edward A. Boyno†
Montclair State University

Abstract

Many texts dealing with database management or data mining include one or two short chapters on data warehousing. I believe that the subject is worthy of more thorough attention and have devised a syllabus for a course in data warehousing intended for students who already have a basic knowledge of traditional database functionality. The course includes a laboratory component that allows students to encounter first hand, and solve, some of the problems associated with building and using the warehouse.

Keywords: Data warehousing, databases, data Mining, OLAP,

1. INTRODUCTION AND OUTLINE.

The goal of the course is to give students a basic understanding of data warehouses. It is intended for advanced undergraduate students, but could, I believe, serve as the basis for a graduate course as well. In the introduction, besides learning basic terminology, students will learn the need for and uses of data warehouses and how they differ from traditional databases. Two sections follow in which the specialized data model and tools used in data warehouses are presented. Once students have acquired sufficient knowledge of the nature of data warehouses, the course presents an overview of the process by which data warehouses are designed and populated. The course concludes with a discussion of some existing technologies, including SQL, and the degree to which they address, or don't address, the requirements of data warehousing. Laboratory exercises are designed to demonstrate the difficulties present in some parts the warehousing process. Specifically addressed are the problems associated with extraction, transformation and loading data and the inadequacies of SQL for use in On-Line Analytical Processing (OLAP).

At the conclusion of the course students should have knowledge of the theoretical and practical foundations of data warehousing. They should be able to work with existing data warehouses and have a basic knowledge of how they are designed. Finally, they should be aware of the problems inherent in the data warehousing process. The outline for the course:

I Introduction (6 hours)
II The Multi-dimensional Data Model (9 hours)
III Specialized Techniques(3 hours)
IV Building the Data Warehouse (12 hours)
V Populating the data warehouse (12 hours)
VI Existing technologies (3 hours)
Examinations (3 hours)

2. THE SYLLABUS

I. Introduction:

I.1 A data warehouse is a special purpose database. Classic databases, so-called "Operational Databases" (OD's), are generally used to model some enterprise. Most often they are used to support transactions, a process that is referred to as On-Line Transaction Processing (OLTP). They are application oriented (tuned for efficient execution of the applications); are isolated (not integrated with other OD's inside an organization); are continuously updated; contain only current data values (don't usually contain historical data) and are rarely used for ad hoc queries (transactions applied to the database are by and large predictable). A data warehouse on the other hand is designed to support decision-making or so-called On-line Analytical Processing (OLAP). The general idea is to prepare data so that useful information can be abstracted from the data via statistical analysis, artificial intelligence or other special purpose techniques. This last process is called "Data Mining". The purpose of a data warehouse is to allow an analyst to view an entire database as if it were one huge spreadsheet (Chaudhury 1998) .

Bill Inmon, (Inmon, 1996) defined a data warehouse as a subject oriented, integrated, time-varying, non-volatile (i.e., containing stabilized data values) collection of data that is used primarily in organizational decision-making. We can add one more property to that list in Inmon's definition: A data warehouse is often subjected to ad hoc queries, which are frequently very complex.

We recognize three distinct sub-types:

1.An Operational Data Store, which is a

† boynoe@mail.montclair.edu

replicated OLTP database that is used for summary analysis.

2. A Data Mart, which is a functionally specialized data warehouse, usually containing a narrower scope of data, perhaps a single subject or business process (Sales or Purchasing, etc.). They usually contain only summary data although they may be linked to an OLTP DBMS.

3. An Enterprise Data Store (EDS), which contains information taken from throughout an organization. It is used for cross-departmental or cross-functional analysis, executive information systems and data mining applications.

Given these sub-types, there are four recognized topologies (Gardner, 1998):

1. A centralized data warehouse. This contains "properly conditioned" data that is relevant to all the business units within an enterprise in one location. It would be the topology of choice for an EDS. It allows centralized system management.

2. A combination of data marts and data warehouses. This organization contains three levels. *Clients* connect to specific *data marts*, which obtain their data from a *centralized warehouse.* This configuration provides optimal results for the individual data mart clients while still allowing cross-functional analysis via the background warehouse. Data mart servers are managed individually.

3. A distributed data warehouse. Individual data warehouses which are connected via a network. This approach has all the benefits and problems of a distributed OD: higher availability and local control of data vs. the expense and difficulty of distributed processing.

4. A hybrid topology. A combination of bottom-up development of data marts together with a top-down, high-level data model. This technique may actually inhibit cross-functional analysis since the individual data marts may be developed with no regard to each other. However, it is also much quicker and cheaper to develop and can serve as an easy way for organizations to begin development. Finally, legacy systems are relatively easy to embed in such a topology.

I.2 The three-tiered architecture. Data warehouses normally implement what is called the three-tiered architecture (Chaudhury, 1998):

The bottom tier is usually a relational database server, or perhaps flat files.

The middle tier is comprised of the OLAP server, either ROLAP or MOLAP (below).

The top tier consists of clients, query and reporting tools, analysis tools and data mining tools.

I.3 Separation of the warehouse and the ODS. A data warehouse should be kept separately from the operational database for both performance and functional reasons.

Performance:

1.The ODS is tuned for transaction processing. Tables are highly normalized, indices and physical considerations like buffering schemes are all designed for transaction processing. Queries generated in an OLAP environment are often very complex, involving large multitable joins for which the set up of the ODS is not suitable. Attempting to do OLAP on the ODS would be inefficient and would probably degrade the OLTP performance as well.

2. Special data organization, access and methods are needed for OLAP. Data is often de-normalized and/or replicated and specialized schemata and indexes are used to make processing more efficient.

3. Concurrency control (lock contention) in a mixed OLTP, OLAP environment can seriously degrade performance.

Functional:

1. Decision support typically requires historical data which is not usually kept in an ODS.

2. Decision support requires aggregation from multiple heterogeneous sources: the operational database, external sources, the web, etc.

3. Different data sources typically use different data representations, which must be reconciled before OLAP processing.

II The Multi-dimensional Data Model:

In the multi-dimensional data model we visualize a record as if it were a point in some multi-dimensional space

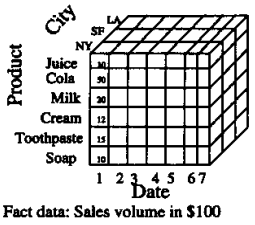

Fig. 1. (Chaudhuri 1998)

Each axis represents a different "dimension" and it is easy to imagine many more dimensions than three. One can always represent a multi-dimensional data set as a relation.

Each point in a multi-dimensional data model representation is called a "fact" and the table itself a fact table. We sometimes call the multi-dimensional data structure a "cube". It is usually appropriate to maintain more detailed information on one or more of the dimensions in a multi-dimensional database. In the above example we might certainly want to know more about the items, the location and the customers and would naturally construct tables to do so. All of these so-called "dimensional" tables would be related to the fact table by the presence in the fact table of the key attribute of the dimensional table as a foreign key. The appearance of the schema, with the fact table at the center and the dimensional tables surrounding it, has caused people to call this a "star" schema. If the "rays" of the "star" themselves have subordinate tables, the schema is rather poetically named a "snowflake". In

large warehouses, fact tables may be shared by multiple dimension tables. These schemata are often called "constellations".

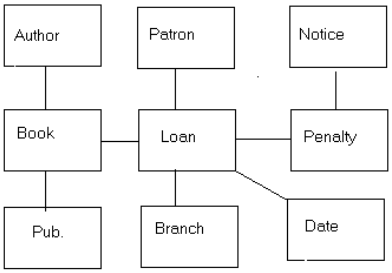

Fig. 2 A snowflake schema with fact table "Loan"

The basic Multi-dimensional operations are:

Aggregation of detailed data to create summary data, sometimes called dimension reduction.

a. Simple aggregation: Find total items sold by location.

b. Multiple aggregations as the name implies, aggregate data along multiple dimensions. There are two recognized sub-types:

"Rollup" is a multiple aggregation operation in which the order of the dimensions makes a difference in the way the aggregate is performed;

rollup total items sold by item, location and color produces the following aggregations:

total items grouped by item, location, color
total items grouped by item, location
total items grouped by item

while

rollup total items sold by location, color and item produces

total items grouped by location, color, item
total items grouped by location, color
total items grouped by location

"Cube" treats all dimensions the same and produces all possible aggregations;

cube total items by item, location and color produces the following aggregations:

total items grouped by item, location, color
total items grouped by item, location
total items grouped by location, color
total items grouped by item, color
total items grouped by item
total items grouped by location
total items grouped by color

(The term rollup is sometimes used to describe any aggregation operation)

Navigation (often called "drill down") moves from an aggregate to detailed data, i.e. , to navigate is to "explode" aggregated data by unaggregating it. For example, given the aggregate sales for a store, display the monthly sales figures.

Selection (often called "slicing") defines a sub-cube. It the equivalent of the geometric projection operation and is accomplished by fixing one or more dimensions as in " Find all sales where city = 'Reno' ". A variant of selection, which involves a range query, is sometimes called "Dicing". An example of dicing might be "Find the sales for products whose advertising budget is greater than $100".

Calculation also has two variations:

a. within a dimension...e.g., calculate (sales - expenses) by office, i.e., within the office dimension

b. across dimensions…e.g., divide the cost of advertising for a family of products by market share of the individual products...note that these numbers probably exist at different levels of the cube.

Ranking or the presentation of ordinal data. For example find the top 3% of offices by average income.

Visualization is the ability to modify the view of the data.

a.Nesting, which is the act of viewing multi-dimensional data in two dimensions. Fig. 1 can be represented in two dimensions as a tree:

```
        1                2                3
    /   |   \        /   |   \        /   |   \
  LA   SF   NY    LA    SF    NY    LA    SF   NY
  /
J C M Cr T S
Etc.
```

b. pivoting or rotating the cube which is the act of presenting a different "face" of the cube.

II.2 Multi-dimensional data models come in two basic flavors:

ROLAP: Extended relational DBMS' that map multi-dimensional operations to standard relational ones, and

MOLAP: Array based storage structures that directly implement multi-dimensional operations.

III. Specialized OLAP tools and Data Manipulation:

III.1 Bit mapped indices. Given an attribute that can assume only a small number of values, a bit-mapped index is a collection of vectors, one for each possible value of the attribute. For every row, the corresponding position in the bit vectors will be 1 in the vector whose value occurs in that row and 0 in the others.

Bit-mapped indices are extremely useful in answering two common kinds of queries: Disjunctive queries such as "Find all female employees in the NY office" and certain aggregate queries such as "How many employees are there in NY?" In the first case, the query processor can perform a bit-wise AND with the F vector and the NY vector to isolate those rows. The second, of course, is merely the number of 1's in the NY vector.

III.2 Join Indexes. Efficient joins are desirable in any database. One technique that can be used to this end is the construction of indices especially for this purpose. The idea is simple; for every pair of

joinable rows we place a pair consisting of their respective row identifiers (rids) into the index. If , for example, we knew that we would frequently be joining the Products table with the Customers table, we might construct an index whose entries have the form (p, c) where p is the rid of a row in Products and c is a customer rid and the corresponding rows participate in the join. The net effect of a join index is to consciously "pre-join" the tables. If a selection condition exists on one or both of the tables involved in the join, we can incorporate it in the choice of rids, excluding those rows which do not meet the condition.

The idea is easy to extend to more than two tables and may be particularly useful in the case of star schemata.

Unfortunately, the number of join indices can get quite large as one is required for every possible combination of joined tables and selection conditions. A solution to this problem is to form a join index for every attribute of every dimension table, which occurs in a selection condition in the following way:

Suppose that dimension table, D, joins with the fact table, F, subject to some selection condition involving attribute A of D. Let v be a value of A occurring in a row of D that joins to a row in F and let r be the rid of that row in F. We place the pair (v, r) in the join index. Given a join that involves multiple tables and multiple selections on multiple rows, we can take the intersection of the rid sets produced by the indices to discover the required rows of F. If it happens that the attributes described above have only a few values, this index can be bit-mapped and the resulting intersection of rid sets would be very efficient to calculate.

III.3 Materialized views. Creation and maintenance of materialized views can be a significant benefit to performance. Materialization of all possible views would be enormously expensive in terms of space, so one must carefully choose which views to materialize. When designing such a scheme one must keep in mind the type, frequency and cost of the queries to be run. The scheme must also take into account the broader context of physical design such as the existence of indices.

III.4 Star query optimizing. Queries on a star schema usually involve joins in which the fact table is joined with multiple dimension tables. Since the fact table is often very much larger than the dimension tables, traditional join techniques may not yield the best result. Special join techniques and optimization are necessary.

III.5 Partitioning. It is often advisable to divide the warehouse into smaller more manageable pieces. These partitions are created in essence by applying "select-from-where" clauses to a warehouse object. The partitions themselves can be sized so as to minimize direct I/O. Transactions applied to the pieces individually may result in substantial performance gains.

IV Building the Data Warehouse:

There are as many techniques for designing a data warehouse as there are authors on the subject. (Bieber 1999; Chaudhury 1997 ; Debevoise 1999 and Gardner 1998). The outline presented here is a synthesis of these.

IV.1 The process.

A. Plan the warehouse

1. Identify Users.
2. Perform a requirements analysis, that is uncover business rules and data requirements
3. Establish the warehouse architecture
4. Model the data:
   a. Produce a high level model using a structured tool such as ERD's or UML
   b. Identify attributes
   c. Identify transactions

B. Design and implement the warehouse

1. Identify the location of the data, design and implement scripts for data extraction, loading and refreshing.
2. Connect sources: gateways, ODBC drivers, and wrappers
3. Design the physical warehouse including placement of data, partitions and access methods.
4. Design and implement applications including data mining application.
5. Integrate database and OLAP servers, storage and client tools.
6. Design and implement the warehouse support and management tools.
7. Roll out the warehouse and the applications

C. Use, Support and allow the warehouse to evolve.

1. Provide hardware and software support.
2. Optimize Performance.
3. Expand the system by including new applications, users or by increasing the use of current applications.
4. Update the system and provide for system growth.

IV.2 Metadata Through all phases of the data warehouse design process, the maintenance of metadata, data about the data, is essential. It is usually maintained and managed in a separate "repository." The repository should provide searchable, understandable access to the metadata. Several kinds of metadata should be maintained (Chaudhury 1998):

1. Administrative information including ODS schemata; gateway descriptions; the warehouse schema, including view definitions; the partition scheme; logical and physical data mapping; transformation rules and defaults; data refreshing and purging rules; a dictionary of applications; security and any other data necessary for using the warehouse. The repository may also contain data relating to system performance.
2. Business information including business terms and definitions; data ownership information and charging policies.

3. Operational information that is, information collected during the operation of the warehouse, including data lineage, currency of data and monitoring information.

V Populating the data warehouse:

There are significant challenges to be overcome when data is drawn from multiple heterogeneous sources into a central repository (Bieber 1999 ; Chaudhury 1998 and 1999 ; Han 2001; Kimball 1997; Moss 1998 and Pyle 1999).

V. 1. The challenges include:

1. Differences in data type. This can occur across products or even within various releases of a single product. The string "ISECON" stored in v6 of Oracle with a data type char(20) will not match the exact same string in v7 of Oracle because the latter's char(20) data type right pads the string with blanks (Corey 1998)
2. Data may be encoded differently in different ODS or in different parts of the same organization. The sex field in an employee record may have domain {'M','F'} in one file and {'Male', Female'} or even { 0,1 } in another.
3. Records may have the same primary key but might have different data. This can occur if primary keys are reused or when one entity acquires another. It can also happen if the same primary key used by different parts of an organization for different purposes.
4. There are often multiple ways to denote a name. "UVA", "Virginia, Univ. of " and "University of Virginia" are all the same. Similarly, some legacy applications may use "Calif" in addresses rather than "CA".
5. Multiple primary keys for the same entity. This can arise when different segments of an entity design databases independently of one another.
6. Invalid Data. A POS terminal may require a customer's phone number, and, if the customer refuses to give it, a clerk may enter 999-999-9999.
7. Same name for a field but different meaning of the data. "Yearly Sales" may mean total sales for a fiscal year to one part of the user population and total sales for a calendar year to another.
8. Required fields left blank.
9. Different formats for primary keys. At my institution, social security numbers are sometimes represented with imbedded dashes, sometimes not.
10. Erroneous data. For example when the state field is "CA" but the zipcode is for New York.
11. Dealing with Null values presents a special set of problems. Missing data can occur for a wide variety of reasons and can be represented in an OD in several ways: using system nulls; using default values or using user-defined nulls.

V. 2. We can identify five distinct phases in the loading process:

1. Extraction: Collection of data from the native format of an ODS. Sources of data include ASCII files, legacy mainframe data perhaps in VSAM files or other proprietary systems and/or commercial DBMS format.
2. Conditioning (data transformation) : Conversion of data from the source data type to the target data type. This step may include aggregation, smoothing (removing "noise" from the data) and normalization (scaling attribute values to fall within a specified range).
3. Scrubbing (data cleansing): Making sure that the data meets all the validation rules that have been decided by the warehouse designers. This includes handling null or missing data; violations of data type; uniform date formats and validating data.

Techniques for this part include: using domain experts to decide on the proper representation of data; parsing and fuzzy matching to decide matches; designating a preferred source, and database rules and triggers.

4. Loading: The actual placement of the data in the target warehouse. This step includes some or all of the following operations: computation of views, integrity checking, index building and partitioning. These operations are usually accomplished via regular operations and/or utilities supplied by the commercial database.

Some issue that need to be addressed during the loading phase are:

- The huge volume of data that may need to be loaded
- The need to take the warehouse off line during the loading process. There is usually a very small window when this can happen (weekends and nights)
- When can or should summary tables and indices be built?
- Transaction management. During a normal transaction, the transaction manager would log every change made to the database. During a load the log file used for this purpose would rapidly overflow causing a system shutdown, so one would normally disable logging during the load but this means that a failure during the load might leave the database in an inconsistent state.

It would be very useful if the system administrator were able to monitor the status of the load, suspend, resume, cancel it or change the load rate.

Some solutions to these problems include loading the data in parallel and/ or loading it incrementally.

5. Refreshing: Propagating changes from the source databases into the data warehouse. There are two basic considerations: How to refresh and when to do it. Options for the first step are:

- fully reloading a table with updated data. This would take as long as the original load and the old table must be maintained until the updated table is ready and the transaction commits. This may be the only option for legacy databases.
- use of incremental techniques (below).

As to when to refresh, there are again two options:

- on every update. This is very expensive but might be necessary if OLAP requires current data, as in

stock market data

•periodically. That is every hour, every day, or after "significant events"

Generally, refresh policy should be set by the system administrator, based on user needs, and may in fact be different for different users. We also note that during refreshment it may be difficult to maintain correctness of derived tables and that optimization could be effected

V 3. Incremental processing. A load may be broken into a sequence of shorter transactions, perhaps after some number of records or some amount of time.

Refresh via full table refresh may be accomplished by the same technique or by only updating changes made to the tuples of the base tables. To do this, such changes must be detected and propagated to the warehouse. This may be accomplished via "Replication servers" used by various DBMS' , including Oracle and IBM, to manage distributed databases or via the use of "triggers." Sybase has a feature called "transaction shipping" that can also be used. There are some problems associated with this approach:

1. The sequence of transactions may interfere with queries.
2. It is difficult to ensure consistency between the base tables, derived tables and indices.

V. 4. Other operations during loading. There are a couple of other operations that can be applied during the population process or to the data itself to make other operations easier:

Auditing. Attempting to uncover unusual facts.

Merging. If a single occurrence of a target datum is found in multiple sources, a single schema must be chosen.

Validating. Making sure that the data has maintained its integrity during the transformation process.

VI. Existing technologies:

VI.1. The insufficiency of SQL. Many of the operations required for a data warehouse are difficult if not impossible to express in SQL: Consider the following "natural" warehouse-type queries:

Assuming the schema Sales(Sales_id, Store_id, Product_id, Sales_Rep, amount, date)

1. Comparisons (with aggregation).Compare last year's sales with this year's sales for each product

2. Multiple Aggregation. Given the sales schema above, find the total sales by store and by product.

3. Ordinal Data. Given the schema above, find the top 25% of sales reps by total sales.

4. Statistical data: Find the 30 day moving average of an product's sales.

5. Time series: Given the schema Transactions(Tx_id, begin_time, end_time, commit). Find all Tx's that began while T1 was executing and ended while T2 was executing.

VI.2. DB2 DB2 provides the following additional functionality for OLAP: Intelligent partitioning; parallel database operations including: table and index scans, joins, backup and recovery, specialized indices and index processing, roll up and cube, star query processing, support for de-normalization via outer-joins and predicate transitive closure. (Bontempo, 1998)

VI.3 Oracle Oracle allows bit-mapped indices, hash joins and partitions and partition-wise joins. It also supports star schema optimization and provides additional tools for building the warehouse (Oracle Warehouse Builder) and data mining (Oracle Express). The server contains some extensions to SQL such as a "rank by" clause. (Corey, 1998)

VI.4. Redbrick, from Informix. Redbrick provides various data warehousing tools. The server manages partitions and supports some specialized indexing and join techniques. Redbrick's Intelligent SQL (RISQL) extends SQL with scalar functions for moving sum and average, percentile and rank. (Redbrick 1998)

3. EDUCATIONAL TECHNOLOGY

By and large the course uses conventional education technology. The exercises are assigned at appropriate times during the course so that students can learn first hand of the difficulties in the warehousing process and gain experience in solving the problems that arise.

Students have access to an existing data warehouse implemented in Oracle 8 running under Solaris. Students use this warehouse, which deals with a library circulation system, to perform a collection of exercises designed to illustrate some techniques of data manipulation and some elementary data mining. Students also design a modest warehouse and implement it in this Oracle instance. Finally students gain experience by populating the practice warehouse with data taken from three existing OLTP databases: another oracle instance running under Windows NT, an Ingres database running under VMS, and an Access database running under Windows 98.

4. EXERCISES

Specialized Warehouse Operations (performed on the practice data warehouse)

1. Roll up. Find total loans by branch. Find total loans by branch and publisher.

2. Slicing. Find all facts for branch 1.

3. Calculation. Find the average penalty per loan by branch

Populating the warehouse

1. Use the DBMS' utilities to copy a small table from one Oracle database to another.

2. Use the DBMS' utilities to copy a large table from an Ingres database to an Oracle database.

3. Develop and implement a strategy for copying and inserting data from a local Access database.

Data Cleansing

1. Handling different data types. Map Ingres' and Access' character and numeric data types to Oracle's

2. Dealing with Null or missing Values: Develop and implement a strategy for loading an Oracle table with data that contains system nulls, user defined nulls and default null values.

3. Dealing with duplicate keys: Construct and implement a strategy for insertion of rows into an Oracle table that contain key values that duplicated existing key values.

Data Mining

1.Write an SQL query to see if there is a correlation between a patron's age and the number of times they have books overdue, i.e., calculate the correlation coefficient.

2.Time Series: Find library patrons who borrow additional books before they return formerly borrowed books.

3.Ranking: Find the five most expensive books. (Without using Oracle's rank by clause

5. REFERENCES

Bieber, M. " Data Warehousing Workshop", CA-World, New Orleans LA, 1999

Bontempo, C., Zagelow, G. "The IBM Data Warehouse Architecture", Communications of the ACM, vol 41 no 9 Sept 1998: p38.

Chaudhuri, S., Dayal U. "An overview of Data Warehousing and OLAP Technology", SIGMOD Record, March 1997: p65

Chaudhuri, S., Dayal U. Decision Support Technologies: OLAP, Data Warehousing & Data Mining. International Conference on Data Engineering, Orlando, FL, 1998.

Corey, M., Abbey, M. , Abramson, I. and Taub, B. Oracle 8 Data Warehousing. Berkeley CA, 1998 Osborne,/McGraw Hill.

Debevoise, N. The Data Warehouse Method Upper Saddle River NJ 1999, Prentice Hall

Elmasri R. Navathe S. Fundamentals of Database Systems 3rd Ed. Redwood City CA 2000, Benjamin/Cummings

Gardner, S. "Building the Data Warehouse". Communications of the ACM, vol 41 no 9 Sept 1998: p52

Han, J. Kimball M. Data Mining, Concepts and Techniques. San Francisco 2001, Morgan Kaufman.

Inmon, W. H. Building the Data Warehouse (2nd Ed.)New York, 1996, John Wiley.

Kimball, R. *Preparing for Data Mining*. DBMS November, 1997, p14

Moss, L. Data Cleansing: *A Dichotomy of Data Warehousing?* DM Review, February 1998

Pyle, D. Data preparation For Data Mining. San Francisco, 1999, Morgan Kaufmann,.

Ramakrisnan, R. Gehrke, J Database Management Systems 2nd ed. Boston, 2000, McGraw-Hill,

Redbrick Warehouse SQL Reference Guide, Redbrick Systems, Los Gatos CA, 1998