# CS352 Lecture - Analytics

Last revised March 5, 2021

*Objectives:*

1. To introduce basic OLAP concepts (cubes, rollups, ranking)
2. To introduce the notion of a data warehouse
3. To introduce the notion of a decision tree

*Materials*

1. Projectable of ER diagram for a transaction in a transactional database
2. Projectable of Figure 11.1 from book
3. Projectable of Figure 11.3 from book
4. Projectables of four SQL queries that get individual components of a crosstab
5. Projectable of slice Figure 11.4 from book
6. Projectable of Figure 11.5 from book
7. Projectable of slice from the above
8. Projectable of SQL query that generates raw data for cube except summaries
9. Projectable of SQL query that gives item name versus color for all sizes summary
10. Projectable Eight SQL queries equivalent to one group by cube query
11. Projectable of equivalent single query using cube
12. Online access to sample database; electronic form of various (6) OLAP queries for cut and paste

## I. **Introduction**

A. One of the largest categories of use for a database system is the storing of <u>transactional data.</u>

1. The term "transaction" here is used in a different sense from the way we used it earlier, though the concepts are related.

2. In this context, a transaction refers to a single interaction between a "customer" and an organization - such as:

a)A single deposit, withdrawal, transfer transaction at a bank.

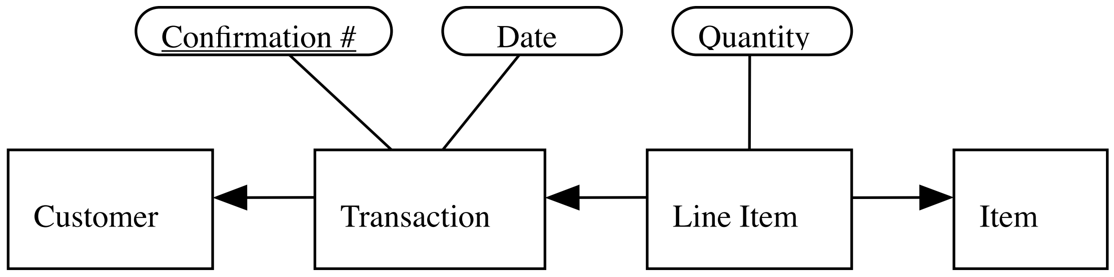
b) A single purchase at a store, or from an ecommerce site

etc.

3. For large firms, especially when the web is used, thousands of transactions may take place in a single hour.

4. Each transaction results in one or more rows being added to tables in the database, or to a similar structure in a non-relational database.

   Example: An ecommerce site might use a structure like this (shown as a simplified ER diagram without showing relationships explitly, where the arrows mean "1" in the relationship and imply the presence of a foreign key in the datatabase schema:



   PROJECT

   A new row is added to the transaction table for each purchase. Associated with this row are one or more rows in the line item table, each describing "line item" of the purchase - i.e. the purchase of some quantity of some particular item.

5. Regardless of how the database is implemented (whether relationally or using some other model), the tables used to store transactional data can be huge of course.

B. While transactional data needs to be stored for its own sake (e.g. to support shipping of an order or listing of a bank transaction on a monthly statement), it also has relevance to organizational decision-making.

1. Manufacturers can use data from their own sales transactions to help them adjust their product offerings to be a better match to what customers are buying.

   Using transactional data in this and similar ways may is called analytics, and the process of drawing conclusions from it may be called OLAP - online analytical processing.

2. There are many ways in which transactional data - frequently aggregated from multiple sources - can be used for predicting certain kinds of behavior.

   Using transactional data to make predictions is called data mining, and sits on the border between Database Systems and AI.

   a. Advertisers use historical data from a person's own browsing history or the behavior of other customers to decide what ads to show to a customer.

      i. Example: "You might like" or "Others customers viewed" on sites like Amazon.

      ii. Google offers a service that recommends ads based on individual browsing history on multiple sites to recommend ads to sites that buy their services in order to increase the likelihood of click-through or conversion. (You no doubt have experienced this!)

   b. Credit bureaus's use historical data from many sources (banks, credit card companies etc.) to predict the credit-worthiness of a customer based on similarity between the customer's income, outstanding debt, etc, and historical data on past customers. (The so-called "credit score").

Banks considering a loan application might also do something similar by comparing the more detailed information in a loan application to their own historical data.

c) Medical data from multiple sources may be analyzed to identify things like risk factors for disease.

C. To support OLAP, another related concept is the notion of a Data Warehouse.

1. For a large organization, the transactional data may be stored in multiple databases (e.g. for different subsidiaries, or at different physical branches, or in different departments)

   a) These databases may have different models or schemes

   b) No one person may have the ability to access all of these databases.

   c) The databases may store only relatively current data

2. To support corporate decision-making, an organization may create a data warehouse - a separate database which stores data collected from a variety of sources - specifically to support decision-making activities.

   PROJECT Figure 11.1 from book

   a) Data is propagated from the various transactional databases (called data sources) to the warehouse on a regular basis (perhaps daily). (Of course, this means that the data in the warehouse is always slightly out of date - but this is not an issue when the data is used for decision making that typically looks at historical trends over a period of months or years)

   b) The data may be transformed in various ways to accomodate differences in models and schemes in the various sources and to eliminate duplicate information, etc.

c) Historical data may be converted to a summarized form to keep down the overall size.

PROJECT Figure 11.3 from book

3. The book discusses data warehouses in the assigned material for today, but we will not discuss this topic further here.

D. The use of data to support various forms of prediction falls into the area that is often called Big Data.

E. We will first discuss OLAP, then Data Mining.

## II. **Basic OLAP Concepts**

A. Various sorts of summary information can be useful in corporate decision-making.

1. Simple summary information is often useful

   Example: Total number sold of a particular item can help a firm decide whether to continue to carry it.

   Such data can be obtained by a SQL query like the following:

```
select item_code, sum(quantity)
   from line_item
   group by item_code
```

   PROJECT

   (or the equivalent accessing multiple aggregates in a non-relational model - but in this lecture we'll develop all our examples using SQL)

2. But often times a further breakdown is useful

Example: Consider clothing items, which come in different colors. It may be that some color is selling poorly, and should be discontinued, even though other color(s) are doing well. (Suppose that the same `item_code` is used for all the colors of a given item, with color being an attribute of `line_item`). This would call for a query like:

```
select item_code, color, sum(quantity)
   from line_item
   group by item_code, color
```

PROJECT

3. Suppose the firm sells many different items in a given color. Then it might be interesting to know not only how popular a particular item is in each color, but also how popular a color is for all items. (This might lead to a decision to begin producing some item in a color that is not currently used for that item, but is popular for similar items)

   The needed summary information could be obtained by using a query like the following :

```
select color, sum(quantity)
   from line_item
   group by color
```

   PROJECT

4. Sometimes items fall into broader categories - e.g. different kinds of pants or shirts or whatever. It may be beneficial to look at summary data for broad categories, rather than detailed information for a particular code - e.g.suppose the item table stores various pieces of information for each item_code, including an item_type. Then a query like the following may be useful:

```
select item_type, color, sum(quantity)
   from line_item natural join item
   group by item_type, color
```

   PROJECT

5. All of these kinds of information might be obtained from a cross-tab like the following:

   PROJECT Figure 11.4 from book

   Such a table cannot be directly produced by a SQL query, of course. However, we will meet some SQL extensions that support extracting the necessary data which can then be converted to a suitable display by additional software

6. The construction of tabulations like the above (sometimes with more than two dimensions) is referred to as Online Analytical Processing (OLAP). Since generating a cross-tab from a huge transactional database "from scratch" can involve massive computation, systems that facilitate OLAP often pre-compute certain summary data to facilitate quick responses to interactive user queries (hence the term "online").

   a. We will not talk about OLAP software per se, but we will talk about some of the SQL facilities that support OLAP-style operations.

   b. This is a place where the facilities available in Db2 differ in some ways from the SQL:2003 standard discussed in the book because - as is often the case - database vendors like IBM developed facilities in their products before a SQL standard was created (and, in fact, the pivot facility itself does not seem to be common in commercial DBMS's, though the equivalent result can be obtained other ways.)

B. Attributes in a transactional scheme fall into two broad categories.

1. Dimensional attributes- e.g. [ for clothing ] item type, color, size.

   Actually, a dimensional attribute can be an attribute that is explicitly stored, or one that is computed from an explicitly stored value.

Example: We may actually store date of birth, but use age (calculated using today's date) as a dimensional attribute.

Example: sometimes it is helpful to use ranges. For example, in an earlier homework, you worked with the employee table in the sample database, which has a "years of education column". You were asked to write a SQL case statement that converted this value into one of three strings (GRADUATE, COLLEGE, HIGH SCHOOL). If one were analyzing some piece of information (e.g. average salary) across this dimension, using just these three categories rather than a distinct category for each different value for years of education would likely be helpful.

2. Measure attributes - a value that can be summarized using some sort of aggregate operation (e.g. sum, average, maximum). It may be an attribute in the table or a value that can be computed from attributes - e.g.. [ for clothing ] quantity sold, monetary value (= quantity * price), or something like count() - a simple count of the number of occurrences.

C. The term "cube" refers to a structure that represents an aggregation of the value of some measure attribute for each possible combination of values of some dimensional attributes, with the number of dimensions of the cube equal to the number of dimensional attributes used. For each dimension, there is usually also a summary value ("all") which represents the aggregation of the values for all the possible values of the dimensional attribute.

The examples we will use are based on three dimensional cubes because they're easier for us to visualize - but in practice "cube" refers to any multidimensional structuring of the data, not just a three-dimensional structure.

Example: PROJECT Figure 11.5 from book

1. Of course, for a cube of three or more dimensions, it is not possible to meaningfully display all the data at one time. Instead, the user can ask to see a particular two-dimensional <u>slice</u> of the cube, formed by using specific values for all but two of the attributes.

Example: for the cube just projected, the user might ask to see a slice showing all the different values for color and item_name for a particular size (e.g. medium).

PROJECT Slice from above cube for all colors, item_name for size medium

The process of creating such a slice is called "slicing" or "slicing and dicing"

2. Note that a cube is quite different from a relational table. In a table, the columns are specified when the table is created. In a cube, the “columns” are determined by the number of different values that occur in the database for a particular attribute.

   Example: if the database included data for striped clothing as well as dark, pastel, and white, then the cube would have one more vertical level.

D. It turns out that cubes cannot be constructed using features of SQL we have covered thus far.

1. We could, however, create some of the data by a query like the following - assuming a table that has columns item_name, color, size (dimension attributes) and number (measure attribute):

```
select item_name, color, size, sum(number)
   from sales
   group by item_name, color, size
```

PROJECT

a) What data needed for the cube would be missing?

   ASK

   The “all” faces

b) Of course, we could use additional SQL queries to generate the missing summaries - e.g.item name versus color for all sizes could be produced by:

```
select item_name, color, sum(number)
 from sales
 group by item_name, color;
```

PROJECT

c) How many queries in all would be needed to get all the different ways of slicing the data (including the original query)?

ASK

(8) PROJECT queries

```
select item_name, color, size, sum(number)
 from sales
 group by item_name, color, size;
select item_name, color, sum(number)
 from sales
 group by item_name, color;
select item_name, size, sum(number)
 from sales
 group by item_name, size;
select color, size, sum(number)
 from sales
 group by color, size;
select item_name, sum(number)
 from sales
 group by item_name;
select color, sum(number)
 from sales
 group by color;
select size, sum(number)
 from sales
 group by size;
select sum(number)
 from sales;
```

2. However, SQL:1999 added a `cube` operator which makes it easy to create such structures.

   The following is a SQL query that would construct the needed data (which would then have to be displayed appropriately):

   ```
   select item_name, color, size, sum(number)
      from sales
      group by cube(item_name, color, size)
   ```

   PROJECT

   This produces the same result as we would get from the above queries

3. An example of using SQL `cube`.

   a) Recall the employee table in the sample database we have used on various homework problems.

      DEMO `select * from employee` (in sample)

   b) Now suppose we want to explore how average salary (a measure attribute) varies with various dimensional attributes (such as department, length of service, job title, education level, gender, age).

      We could, for example, use a query like:

      ```
      select sex, avg(salary)
           from employee
           group by sex;
      ```

      To see how average salary varies by gender.

      DEMO

   c) We could see how average salary varies over several different attributes (job, education level, gender) by using a cube query like:

      ```
      select job, edlevel, sex, avg(salary)
           from employee
           group by cube(job, edlevel, sex)
           order by job, edlevel, sex;
      ```

      DEMO

(BTW: Note the use of `order by` to control the final order of printing. DEMO w/o this).

d) Actually, it might be better to look at the edlevel dimension by broad groups (as in homework 2) rather than by individual values. To do this, we could use something like:

```
select job, education, sex, avg(salary)
  from (select job, case
             when edlevel >= 18 then 'GRADUATE'
             when edlevel >= 16 then 'COLLEGE'
             else 'HIGH SCHOOL'
          end as education, sex, salary
        from employee) as e
   group by cube(job, education, sex)
   order by job, education, sex;
```

e) Actually, another issue may be involved here that one can see when looking at experience as well as formal education

```
select education, experience, sex, avg(salary)
        from (select case
               when edlevel >= 18 then 'GRADUATE'
               when edlevel >= 16 then 'COLLEGE'
               else 'HIGH SCHOOL'
             end as education,
             case
               when hiredate < '01-01-1960' then 'OVER 20'
               when hiredate < '01-01-1970' then '10-20'
               when hiredate < '01-01-1975' then '5-10'
               else 'NEW'
             end as experience,
             sex,
             salary
           from employee) as e
        group by cube(education, experience, sex)
        order by education, experience, sex;
```

E. SQL also supports another similar OLAP operation called `rollup`.

1. The difference is basically this: with cube, if you specify n dimensions, you form $2^n$ groups, representing all the possible combinations of the various dimensions and “all”. With rollup, you get n+1 groups - all the dimensions, all the dimensions except the last, all the dimensions except the last and second to the last ...

2. We can illustrate this using a query similar to the one we just did with cube.

```
select job, education, sex, avg(salary)
  from (select job, case
             when edlevel >= 18 then 'GRADUATE'
             when edlevel >= 16 then 'COLLEGE'
             else 'HIGH SCHOOL'
          end as education, sex, salary
        from employee) as e
   group by rollup(job, education, sex)
   order by job, education, sex;
```

DEMO

In this case, the groups we get are based on all three dimensions; just workdept and job; just workdept, and an overall summary

F. SQL also supports OLAP operations related to ranking

1. For example, suppose we were interested in looking at the relationship between salary and education level. We might formulate a query like the following:

```
select firstnme, lastname, salary,
        rank() over (order by edlevel desc) as edrank
   from employee
   order by salary desc;
```

DEMO (Note that “1” means the greatest amount of formal education, etc.)

2. Do you notice anything funny about the values reported for EDRANK?

   ASK

   Several values appear multiple times (e.g. 3, 12) and other values don't appear at all (e.g. 4, 13 ...)

   This is because, when there are ties, the rank() function gives all the tied values the same rank, and then skips over as many rank values as there are duplicates (e.g. since four people are tied for third, the next person is given rank 7)

3. An alternate function called `dense_rank` does not skip rank numbers

   ```
   select firstnme, lastname, salary,
      dense_rank() over (order by edlevel desc) as edrank
      from employee
      order by salary desc;
   ```

   DEMO

4. There are other facilities available as well e.g. GROUPING and PARTITION by.

## III. **Data Mining**

A. Data mining involves looking at a body of data to see what patterns may be present in it, without any predetermined notions as to what might be discovered. It can be thought of as a form of machine learning in which the learning is based on a large body of data in a database.

   1. Example: one often hears about medical "risk factors" for various conditions - e.g. belonging to a certain ethnic group, taking a certain medicine, or smoking, or ... Here, a database containing information on patients with certain conditions is mined. Note that risk factors may include both obvious and non-obvious attributes [e.g. smoking, ethnicity ].

2. Example: If advertisers know that a particular product is especially attractive to a particular group of customers, they may target their advertising toward that demographic in terms of choosing which programs to air their commercials on. Here, a database of information on demographics and purchases is mined to discover associations between certain demographic characteristics and products they purchase.

3. Example: It may turn out that people who purchase some particular product are more likely to purchase another particular product. In that case, a sales person or e-commerce system may suggest the second product to the purchaser of the first product, or a coupon printer at store checkout may print a coupon for the second product when a person purchases the first product. Here, a database of historical records concerning of purchases by individuals is mined to find such associations.

4. Data mining can easily be the subject of graduate-level courses, We will look at it only briefly here.

B. Data mining differs from OLAP in that OLAP requires a human user to explicitly specify the groupings to be considered, while data mining looks for patterns a human user might not anticipate.

C. In data mining, we are looking for rules of the general form

if these conditions are present in the data, then this conclusion is likely

1. Such rules are seldom 100% accurate, of course.

2. But they are valuable to the extent that they are generally true.

D. One important use of data mining is for <u>prediction.</u> For example, a bank might want to predict whether or not an individual is a good credit risk. Data mining techniques approach a problem like this as follows:

1. We begin with a collection of historical data which includes the value of various variables that were known at the time an individual applied for credit plus an indication of how the applicant actually performed as a credit recipient (did the applicant default on a loan, was the applicant habitually late with payments ...)

   This set of historical data is known as the training instances.

2. The training instances are mined to find correlations between variables known at application time and the actual credit-worthiness of an applicant. The goal of the mining process is to discover rules which, given variables known at application time, will successfully classify an individual as an excellent, good, average, or poor credit risk. - e.g

   if this pattern is present in the input data, then it is likely that this individual will be a ______ credit risk

   A desirable rule is one which, when applied to the various training instances, with high reliability is able to predict what actually transpired.

3. These rules are then used to "score" future applications - assuming that the same patterns that were present in the historical data will also be true in the future.

4. There are a number of different approaches that can be used to discover such patterns.

   a. These were briefly introduced in the portion of the chapter 11 part 4 that was not assigned - but you're welcome to look at this material if you wish.

   b. At Gordon, the machine learning course which Dr. Senning taught this fall (and hopefully will teach again) goes into these approaches in much more detail - hence we will not go into them today.

E. Another use of data mining is to discover <u>associations.</u>

1. As an example, book sellers like Amazon look for associations between books people bought - i.e. they are seeking to discover rules of the form "a person who bought X is also likely to buy Y". How do they use this kind of information?

   ASK

   a) Most of their pages describing products include a section near the bottom that reads "Customers who bought this item also bought ...." (The hope is that, if you are interested in some item - even if you don't buy it, you may be interested in these other items as well, and may buy one of them).

   b) When a regular customer connects to Amazon, they furnish a section labelled "Today's recommendations for you" - which lists titles Amazon thinks a customer might buy based on previous purchases that customer has made.

2. Companies like grocery stores, drug stores, etc. look for associations between products a customer purchases. This may affect placement of the products in the store, as well as generation of checkout coupons.

3. While associations within a single visit are important, companies are also interested in associations over time - e.g. how does what associations are there between what the customer buys on one visit to the store and what the customer buys on an other visit.

   a) One way of seeing how important this is to various businesses is the proliferation of various "frequent shopper programs". Many stores will let you register and offer significant discounts if you consistently use your frequent shopper tag when you make purchases.

   b) For the most part, such programs are <u>not</u> designed to generate mailing lists for email or snail mail. Rather, they are used to facilitate data mining, by allowing the company to build a record of purchases a given customer makes over time. (Absent such a program, the store would have records of myriads of transactions, but no way to associate a given transaction with previous transactions serving the same customer.)

4. Associations may be expressed as rules of the form

   some pattern => some other pattern

   (e.g. for a grocery store: includes(T, bread) => includes(T, milk)) - i.e. “if some transaction T includes bread, then it also includes milk”

   a) The left hand side of the rule is called the antecedent (e.g., in the above, includes(T, bread))

   b) The right hand side of the rule is called the consequent (e.g. in the above includes(T, milk))

   c) For each possible rule, two measures can be calculated. For any given problem, we are only interested in rules for which both measures are sufficiently high (where “sufficiently high” depends on the application)

      (1)The <u>support</u> for a rule is the percentage of all cases in which both the antecedent and the consequent are present. Rules with low support are uninteresting for two reasons:

         (a)The evidence for them is weak, and may be statistically insignificant

         Example: the support for the rule

         name(s, bjork) => teaches(s, cs) is less than 0.01 (even when my wife and I both worked here, less than 1% of Gordon employees had name Bjork and teach CS).

(b)The number of cases a rule applies to might be so small that it doesn't warrant attention on the company's part

(2)The <u>confidence</u> for a rule is the percentage of cases where the antecedent is true where the consequent is also true. Only rules with high confidence are of significant predictive value\

Example: For the rule: name(s, bjork) => teaches(s, cs), was 0.5 before my wife retired, since there were only 2 employees at Gordon with last name bjork. (Now the confidence is 1!).

(3)Rules that are interesting are ones that have both sufficiently high support and sufficiently high confidence.

(a)Thus, the rule name(s, bjork) => teaches(s, cs) is actually not very interesting because of its low support.

(b)We'll see an example of a rule that has both high support and confidence shortly

5. An example: we will use a database describing hypothetical members of the Jets and Sharks gangs of Westside story fame.

a) Connect to westside database

b) `select * from gangsters;`

c) Consider the association between age and gang membership. It turns out that - in this particular set of data - the Jets tend to be younger than the Sharks. Consider the following pair of rules:

age(X, in20s) => gang(X, jets)
gang(X, jets) => age(X, in20s)

To calculate support and confidence for these rules, we could use a query like the following:

```
select gang, age, count(*)
 from gangsters
 group by cube(gang, age)
 order by gang, age;
```

DEMO

d) It turns out that 9 of the 27 gangsters are in their 20's and Jets - so the support for both rules is 0.33

e) Of the 10 gangsters in their 20's, 9 are jets, so the confidence of the first rule is 9/10 or 0.9. (So if you were a police officer and you saw a young gangster, you'd be on pretty safe ground to assume he was a Jet!)

f) Of the 15 Jets, 9 are in their 20's, so the confidence of the second rule is 9/15 or 0.6.

F. Data mining turns out to be a place where there is considerable overlap between the areas of artificial intelligence and database systems - in fact, you see this as a topic in both kinds of textbooks, and the database we just used is also used a standard example in teaching about neural networks in AI!