CS145 Lecture Notes #16

Beyond CS145: Data Warehousing, Data Mining, XML/XQL, Search Engines

Data Warehousing

Two types of database loads:

- *OLTP:* On-Line Transaction Processing
  - Lots of short, read/write transactions
  - Small, simple queries
  - Frequent updates
- *OLAP:* On-Line Analytical Processing
  - Long, read-only transactions
  - Huge, complex queries
  - Rare updates

Data warehousing: bring data from operational (OLTP) sources into a central warehouse to do OLAP

ROLAP—Relational OLAP

A grossly simplified example of a *star schema:*

- *Dimension tables:*
  - `Stores(`<u>`StoreID`</u>`, city, state)`
  - `Items(`<u>`ItemID`</u>`, name, description)`
  - `Custs(`<u>`CustID`</u>`, name, address)`
- *Fact table:*
  - `Sales(`<u>`StoreID`</u>`, `<u>`ItemID`</u>`, `<u>`CustID`</u>`, price)`
- *Star join:*

```
SELECT *
FROM   Sales, Stores, Items, Custs
WHERE  Sales.StoreID = Stores.StoreID
AND    Sales.ItemID = Items.ItemID
AND    Sales.CustID = Custs.CustID;
```

A simple OLAP query: total sales for each store in California

```
SELECT   Sales.StoreID, SUM(price)
FROM     Sales, Stores
WHERE    Sales.StoreID = Stores.StoreID
AND      Stores.state = 'CA'
GROUP BY Sales.StoreID;
```

Idea: materialize views to speed up query

- $V_{\{\mathtt{store,item}\}}$:

  ```
  SELECT   StoreID, ItemID, SUM(price) AS total
  FROM     Sales
  GROUP BY StoreID, ItemID;
  ```

$\leadsto$ Rewrite the query using $V_{\{\mathtt{store,item}\}}$?

- $V_{\{\mathtt{store}\}}$:

  ```
  SELECT   StoreID, SUM(price) AS total
  FROM     Sales
  GROUP BY StoreID;
  ```

$\leadsto$ Rewrite the query using $V_{\{\mathtt{store}\}}$?

- $V_{\varnothing}$:

  ```
  SELECT SUM(price) AS total FROM Sales;
  ```

$\leadsto$ Rewrite the query using $V_{\varnothing}$?

Problem: which views to materialize?

- Views with more `GROUP-BY` attributes:
  - $\leadsto$ Bigger, more detailed, benefit more queries
- Views with fewer `GROUP-BY` attributes:
  - $\leadsto$ Smaller, more summarized, benefit queries more

MOLAP—Multidimensional OLAP

A *data cube* based on the same example:

Coordinate system:

- Points inside the cube:
- Points on the store-item plane:
- Points on the store axis:
- Origin:

Operations:

- *Roll up:* detailed data $\longrightarrow$ summarized data
- *Drill down:* summarized data $\longrightarrow$ detailed data

Data Mining

Data mining: search for patterns and structure in large data sets
An example of *market basket* data: `Sales(basketID, item)`

Mining for *association rules:* conditional implications between sets of items
"$X \to Y$" means "if a customer buys X, then this customer will very likely buy Y as well" (e.g., $\{\text{bread}, \text{milk}\} \to \{\text{eggs}\}$, $\{\text{diapers}\} \to \{\text{beer}\}$)

- X must appear in many baskets

$$\text{Support}(X) = \frac{\text{\# of baskets containing } X}{\text{total \# of baskets}}$$

- Probability of Y appearing given that X is in the basket must be high

$$\text{Confidence}(X \to Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

XML & XQL

XML: Extensible Markup Language

- Future of Web?
- Two modes:
  - *Well-formed* XML: semistructured
  - *Valid* XML: structured
    $\leadsto$ A *DTD* (Document Type Definition) specifies the schema of a valid XML document

Well-formed XML:

- An *element* is enclosed by a pair of *tags*
  - Elements can be nested
  - Attributes can be specified inside element tags
- Elements form a hierarchy; at the top is a *root element*

Example: bookstore inventory database
⇝ Can be viewed as a tree where nodes are elements and edges are tags

```
<?xml version='1.0' standalone='yes'?>
<bookstore>
  <book>
    <title>A First Course in Database Systems</title>
    <author>
      <name><first-name>Jeff</first-name>
            <last-name>Ullman</last-name></name>
      <degree>PhD, Princeton</degree>
    </author>
    <author>
      <name><first-name>Jennifer</first-name>
            <last-name>Widom</last-name></name>
      <degree>PhD, Cornell</degree>
    </author>
  </book>
  <book>
    <title>Compiler Design:
           Principles, Tools, and Techniques</title>
    <author><name>Alfred Aho</name></author>
    <author><name>Ravi Sethi</name></author>
    <author><name>Jeff Ullman</name></author>
  </book>
</bookstore>
```

XQL: XML Query Language

- An XQL query returns a collection of elements
- The basic syntax mimics UNIX directory
  - There is a notion of current context "`.`"
    Example: suppose the current context is the first book element
  - Queries use the current context by default
    Example: find degree elements inside authors in the current context
  - We can also specify queries to use the root context instead
    Example: all authors of books inside the bookstore
  - "`//`" matches any sequence of tags
    Example: find name elements anywhere inside the current context
  - "`*`" matches any single tag
    Example: find all names that are grandchildren of books, anywhere inside the document

- Filters, enclosed in "`[]`", can be attached anywhere along the path
  Example: find titles of books where the book contains at least one author with a degree

  Example: find books written by Cornell Ph.D.'s

- Methods can be invoked using "`!`"
– A built-in method `text()` returns all text contained within an element and its descendents, minus any structure
  Example: find all books written by Jeff Ullman

Why XML?

- Simple, flexible
- Separation of presentation and content
  - Content: specified in XML
  - Presentation: specified in *XSL* (Extensible Stylesheet Language)

Search Engines

- Find pages with "cat" and "dog"
- Find pages with "cat" or "dog"
- Find pages with "cat" and "dog" close together

$\leadsto$ Rank pages according to how many times "cat" and "dog" appear
$\leadsto$ Rank pages according to how many hits they receive
$\leadsto$ Rank pages according to how many important pages link to them

Inverted lists:

– Sort each list according to page rank?
– Store the position of the word in the page?
– Store the context in which the word appears?