Data Mining, Database Tuning

Tuesday, Feb. 27, 2007

Outline

- Data Mining: chapter 26
- Database tuning: chapter 20

Data Mining

- Data mining is the exploration and analysis of large quantities of data in order to discover valid, novel, potentially useful, and ultimately understandable patterns in data.
- Example pattern (Census Bureau Data):
  - If (relationship = husband), then (gender = male). 99.6%

Data Mining

- Valid: The patterns hold in general.
- Novel: We did not know the pattern beforehand.
- Useful: We can devise actions from the patterns.
- Understandable: We can interpret and comprehend the patterns.

Why Use Data Mining Today ?

Human analysis skills are inadequate:

- Volume and dimensionality of the data
- High data growth rate

Availability of:

- Data
- Storage
- Computational power
- Off-the-shelf software
- Expertise

Types of Data Mining

- Association Rules
- Decision trees
- Clustering
- Niave Bayes
- Etc, etc, etc.

We'll discuss only association rules, and only briefly.

Association Rules

- Most studied mining method in db community:
  - Simple, easy to understand
  - Clever, scalable algorithm

We discuss only association rules in class

- Project Phase 4, Task 1:
  - Use association rules
  - You should be done in 10’
- Tasks 2, 3: may try something else
  - E.g Bayesian Networks
  - But need to read first

Association Rules

Market Basket Analysis

- Consider shopping cart filled with several items
- Market basket analysis tries to answer the following questions:
  - Who makes purchases?
  - What do customers buy together?
  - In what order do customers purchase items?

Market Basket Analysis

A database of customer transactions

- Each transaction is a set of items
- Example: Transaction with TID 111 contains items {Pen, Ink, Milk, Juice}

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Market Basket Analysis

Coocurrences

- 80% of all customers purchase items X, Y and Z together.

Association rules

- 60% of all customers who purchase X and Y also buy Z.

Sequential patterns

- 60% of customers who first buy X also purchase Y within three weeks.

Market Basket Analysis

We prune the set of all possible association rules using two interestingness measures:

- Confidence of a rule:
  - X -->Y has confidence c if $P(Y|X) = c$
- Support of a rule:
  - X -->Y has support s if $P(XY) = s$

We can also define

- Support of an itemset (a coocurrence) XY:
  - XY has support s if $P(XY) = s$

Market Basket Analysis

Examples:

- {Pen} => {Milk}

Support: 75%

Confidence: 75%

- {Ink} => {Pen}

Support: 100%

Confidence: 100%

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Market Basket Analysis

Find all itemsets with support >= 75%?

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Market Basket Analysis

Can you find all association rules with support >= 50%?

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Finding Frequent Itemsets

- Input: a set of “transactions”:

TID	**ItemSet**
T_1	Pen, Milk, Juice, Wine
T_2	Pen, Beer, Juice, Eggs, Bread, Salad
. . .	
T_n	Beer, Diapers

Finding Frequent Itemsets

- Itemset I; E.g I = {Milk, Eggs, Diapers}

TID	ItemSet
T_1	Pen, Milk, Juice, Wine
T_2	Pen, Beer, Juice, Eggs, Bread, Salad
. . .	
T_n	Beer, Diapers

Support of I = supp(I) = # of transactions that contain I

Finding Frequent Itemsets

- Find ALL itemsets I with supp(I) > minsup

TID	**ItemSet**
T_1	Pen, Milk, Juice, Wine
T_2	Pen, Beer, Juice, Eggs, Bread, Salad
. . .	
T_n	Beer, Diapers

Problem: too many I's to check; too big a table (sequential scan)

A priory property

$I \subset I' \Rightarrow \text{supp}(I) \geq \text{supp}(I')$ (WHY ??)

TID	**ItemSet**
T_1	Pen, Milk, Juice, Wine
T_2	Pen, Beer, Juice, Eggs, Bread, Salad
. . .	
T_n	Beer, Diapers

Question: which is bigger supp({Pen}) or supp({Pen, Beer}) ?

The A-priori Algorithm

Goal: find all itemsets I s.t. supp(I) > minsupp

- For each item X check if supp(X) > minsupp then retain I_1 = {X}
- K=1
- Repeat
  - For every itemset I_k, generate all itemsets I_{k+1} s.t. $I_k \subset I_{k+1}$
  - Scan all transactions and compute supp(I_{k+1}) for all itemsets I_{k+1}
  - Drop itemsets I_{k+1} with support < minsupp
- Until no new frequent itemsets are found

Association Rules

Finally, construct all rules $X \rightarrow Y$ s.t.

- XY has high support
- Supp(XY)/Supp(X) > min-confidence

Database Tuning

- Goal: improve performance, without affecting the application
  - Recall the “data independence” principle
- How to achieve good performance:
  - Make good design choices (we’ve been studying this for 8 weeks…)
  - Physical database design, or “database tuning”

The Database Workload

- A list of queries, together with their frequencies
  - Note these queries are typically parameterized, since they are embedded in applications
- A list of updates and their frequencies
- Performance goals for each type of query and update

Analyze the Workload

- For each query:
  - What tables/attributes does it touch
  - How selective are the conditions; note: this is even harder since queries are parameterized
- For each update:
  - What kind of update
  - What tables/attributes does it affect

Physical Design and Tuning

- Choose what indexes to create
- Tune the conceptual schema:
  - Alternative BCNF form (recall: there can be several choices)
  - Denormalization: may seem necessary for performance
  - Vertical/horizontal partitioning (see the lecture on views)
  - Materialized views
- Manual query/transaction rewriting

Guidelines for Index Selection

- Guideline 1: don’t build it unless someone needs it !
- Guideline 2: consider building it if it occurs in a WHERE clause
  - WHERE R.A=555 --- consider B+-tree or hash-index
  - WHERE R.A > 555 and R.A < 777 -- consider B+ tree

Guidelines for Index Selection

- Guideline 3: Multi-attribute indexes
  - WHERE R.A = 555 and R.B = 999 --- consider an index with key (A,B)
  - Note: multi-attribute indexes enable “index only” strategies
- Guideline 4: which index to cluster
  - Rule of thumb: range predicate $\Rightarrow$ clustered
  - Rule of thumb: “index only” $\Rightarrow$ unclustered

Guidelines for Index Selection

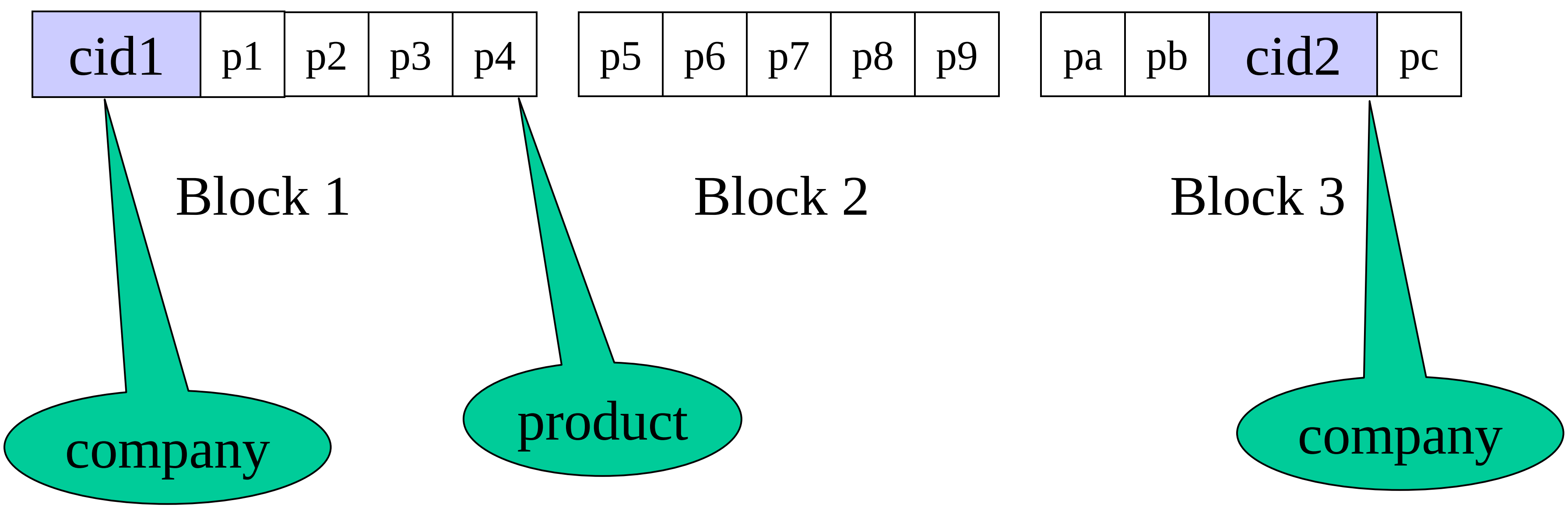
- Guideline 5: Hash v.s. B+ tree
  - For index nested loop join: prefer hash
  - Range predicates: prefer B+
- Guideline 6: balance maintenance cost v.s. benefit
  - If touched by too many updates, perhaps drop it

Clustered v.s. Unclustered Index

- Recall that when the selectivity is low, then an unclustered index may be less efficient than a linear scan.
- See graph on pp. 660

Co-clustering Two Relations

Product(pid, pname, manufacturer, price)
Company(cid, cname, address)



We say that Company is unclustered

Index-Only Plans

```
SELECT Company.name
FROM Company, Product
WHERE Company.cid = Product.manufacturer
```

```
SELECT Company.name, Company.city,Product.price
FROM Company, Product
WHERE Company.cid = Product.manufacturer
```

How can we evaluate these using an index only ?

Automatic Index Selection

SQL Server -- see book

Denormalization

- 3NF instead of BCNF
- Alternative BCNF when possible
- Denormalize (I.e. keep the join)
- Vertical partitioning
- Horizontal partitioning