Know your data(Chapter 2)

1.Type: Nominal：定类 Binary： Ordinal: 定序 *Numeric（**Interval-scaled：**定距，0 不代表没有 **Ratio-scaled：**定比，0 代表没有）

2.Unimodal 单峰： mean-mode=3*(mean-median)

3.Symmetric: mode=median=mean

positively skewed: < < **negatively skewed:** > >

4. Variance: sample(n): population(N):

$$s^2=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2=\frac{1}{n-1}\Big[\sum_{i=1}^{n}x_i^2-\frac{1}{n}\Big(\sum_{i=1}^{n}x_i\Big)^2\Big]\qquad \sigma^2=\frac{1}{N}\sum_{i=1}^{n}(x_i-\mu)^2=\frac{1}{N}\sum_{i=1}^{n}x_i^2-\mu^2$$

5.Q1:25% **Q3:**75% **Inter-quartile range:**IQR=Q3-Q1

6.Graphic Displays of Basic Statistical Descriptions: Boxplot: min Q1 Q2/median Q3 max max-Q3/A1-min=whisker **Histogram:** x values, y frequencies, show distributions of variables & binned quantitative data. But bar charts: compare variables & categorical data **Quantile plot:** Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences). **Quantile-q plot:** graphs the quantiles of one univariant distribution against the corresponding quantiles of another **Scatter plot:** Provides a first look at bivariate data to see clusters of points, outliers

7.Data Visualization: *Pixel oriented: m dimensions with m windows. To save space and show the connections among multiple dimensions, circle segment ***Geometric Projection: (Direct Data Visualization: Scatterplot Matrices(2D):** k*(k-1)/2 **Landscape: Parallel Coordinates:** axes [minimum, maximum]**)** ***Icon-based:** features of icons. Shape/color/tile bars(**Chernoff Faces/Stick Figures)** ***Hierarchical: (Dimensional Stacking**: Partitioning of the attribute value ranges into classes. **Words within worlds:** Assign the function and two most important parameters to innermost world **Tree-Map: InfoCube:) Tag cloud: Social Networks**

8.Similarity: *Z-score: $z=\frac{x-\mu}{\sigma}$

Minkowski Distance: $d(i,j)=\sqrt[p]{|x_{i1}-x_{j1}|^p+|x_{i2}-x_{j2}|^p+\dots+|x_{il}-x_{jl}|^p}$

Manhattan/City Block Distance: p=1 **Euclidean Distance:** p=2

Supremum D: $d(i,j)=\lim_{p\to\infty}\sqrt[p]{|x_{i1}-x_{j1}|^p+|x_{i2}-x_{j2}|^p+\cdots+|x_{il}-x_{jl}|^p}=\max_{f=1}^{l}|x_{if}-x_{jf}|$

9.JaccardCoefficient/Coherence(i,j): j: 1 0

$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|}$ $d(i,j)=\frac{r+s}{q+r+s+t}$ symmetric i:1 q r

$d(i,j)=\frac{r+s}{q+r+s}$ asymmetric 0 s t

Cosine Similarity: $cos(d_1,d_2)=\frac{d_1\bullet d_2}{\|d_1\|\times\|d_2\|}$

Chi-square(X2): $\chi^2=\sum_i^n\frac{(\overset{\text{observed}}{O_i}-E_i)^2}{\underset{\text{expected}}{E_i}}$

Null hypothesis: two distributions are independent 自由度=(行数-1) *(列数-1) 查表得到不相关的概率，卡方值越大说明关联性越强。

10.Variance: $\sigma^2=\mathrm{var}(X)=E[(X-\mu)^2]=E[X^2]-\mu^2=E[X^2]-[E(x)]^2$

Sample var: Est avg == real avg, N; Est avg != real avg, N-1

Covariance: $\sigma_{12}=E[(X_1-\mu_1)(X_2-\mu_2)]=E[X_1X_2]-\mu_1\mu_2=E[X_1X_2]-E[X_1]E[X_2]$

Sample cov: (Total Cov/n). >0, pos;独立则==0(反不可); <0, neg

11. Pearson Correlation: normalize covariance with standard deviation [-1, 1]

$\rho_{12}=\frac{\sigma_{12}}{\sigma_1\sigma_2}=\frac{\sigma_{12}}{\sqrt{\sigma_1^2\sigma_2^2}}$ >0, pos correlated; ==0, independent; <0, neg

12.KL Divergence: measure the difference between two probability distributions over the same x. D(p(X)||q(X)), P true/observation, 1 model/approximation of p, from q(prior(to p(posterior). D>=0 and only p==q,D == 0. P:0,D:0;q:0,D:正无穷

$D_{KL}(p(x)\|q(x))=\sum_{x\in X}p(x)\ln\frac{p(x)}{q(x)}$ smoothing: not counting the possibility of unseen

Value Range jacc_coef,=[0,1]; cov=(-inf, inf); KL-div=[0, inf]; z=(-inf,inf); pearson cor coef=[-1, 1]; L_inf=[0, inf]=[min(s(ij)), max(s(ij))] s(ij)=supdist(I,j)

Data Processing(Chapter3)

1.Quality: accuracy, completeness, consistency, timeliness, believability, interpretability

Cleaning: Incomplete-ignore, Noisy(Binning-Equal-frequency), Regression, Clustering-outlier), Inconsistent, Intentional;

2.Integration: schema integration(feature 一样但不同名)

3.Data Reduction: Parametric(可以用 mode 表示)/Non-parametric Methods; dependent/response variable, independent/explanatory variable; Regression/ Log-Linear/ Histogram Analysis/ Clustering Analysis/ Sampling(random sampling-equal probability, without replacement-selected & removed, stratified-partition/cluster first then sampling)/ **Data Cube aggregation/ Data Compression4. Data Transformation: Normalization(min-max/z-score/decimal scaling)**

4.Discretization: Binning-unsupervised (Equal-width/distance partition Equal-depth (frequency) partition, Top-down **Histogram**-un,td **Clustering**-un,td or bottom-up **Classification(Decision-Tree**-su,td) **Correlation(x^2)**-su, bu **Concept Hierarchy Generation**

5.Dimensionality Reduction: Feature Selection: subset; **Feature extraction: 降维**（**PCA:** linearly uncorrelated variables called principal components. Find the eigenvectors of the covariance matrix, and these eigenvectors define the new space. **Attribute Subset Selection:** Redundant attributes. Irrelevant attributes. **Attribute Creation**: (**Attribute extraction-**domain-specific**, Mapping data to new space-Fourier/wavelet transformation, Attribute Construction**-Combining features**)**

Data Warehousing and OLAP for Data Mining (Chapter4)

1.Basic Concepts: Subject-oriented, Integrated, Time Variant, Nonvolatile(initial loading of data & access of data).

2.Type:Enterprise warehouse: collect all of the information about subject spanning the entire organization **Data Mart:** A subset of corporate-wide data that is of value to a specific groups of users **Virtual warehouse:** a set of views over operational databases

3.Operations:Data extraction/Data cleaning/ Load/ Data transformation(host format to warehouse format)/ **Refresh**

4.Dimension Tables-item; **Fact Table-**measures; **Data Cube**-(nD: base cuboid, 0D: apex cuboid)

5.Modeling:**Star Schema**: A fact table in the middle connected to a set of dimension tables **Snowflake schema:** dimensional hierarchy **Fact constellations:** Multiple fact tables

6.OLAP Operations: Roll up/Drill-up: summarize, dimension reduction; **Drill down/roll down:** new dimensions **Slice:** dimension's subset **Dice:** data's subset **Pivot/rotate: Drill Across:** involving across more than one fact table **Drill through:** Through the bottom level of cube to its back-end relational tabl

7.Measures: Distributive: if the result derived by applying the function to n aggregate values is the same as that derived by applying the function on all the data without partitioning E.g., count(), sum(), min(), max() **Algebraic:** if it can be computed by an algebraic function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate functionavg(x) = sum(x) / count(x), average, std, maxN, minN, CenterOfMass **Holistic:** if there is no constant bound on the storage size needed to describe a subaggregate .E.g., median(), mode(), rank(),Q1, Q3

8.Materialization: Full/No/Partial Materialization

9.How many cuboids in an n-dimensional cube with L levels:

$T=\prod_{i=1}^{n}(L_i+1)$ 3 dimensions: 3 levels + 4 levels + 5 levels T = (3+1) (4+1) (5+1)

10.Bitmap Index: speed up + reduce storage

11.Architectures: Relational(ROLAP):use relational or extended-relational DBMS **Multidimensional(MOLAP):** sparse array-based multidimensional storage engine **Hybrid(HOLAP):** low level: relational, high-level: array

*** P:** 和顺序有关 P(n,m) = n! / (n-m)!

*** C:** 和顺序无关 C(n,m)=n! / [(n-m)! * m!]

Data Cube Technology (Chapter 5)

1. **Base Cell / Aggregate Cell; Full Cube/Iceberg Cube**

2.Close Cube: a cell c is closed if there exists no cell d, such that d is a descendant of c, and d has the same measure value as c

3. Computation Methods:

- Example: A cube with 100 dimensions full $=2^{100}$
  - Suppose it contains only 2 base cells: $\{(a_1, a_2, a_3, \dots, a_{100}), (a_1, a_2, b_3, \dots, b_{100})\}$ only 2 records
  - How many aggregate cells if "having count >= 1"?
    - Answer: $(2^{101}-2)-4$ ~~(Why?!)~~

Let cube P have only 2 base cells: $\{(a_1, a_2, a_3 \dots, a_{100}):10, (a_1, a_2, b_3, \dots, b_{100}):10\}$

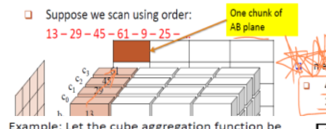
- How many cells will the iceberg cube contain if "having count(*) ≥ 10"?
  - Answer: $2^{101}-4$ (still too big!)

General Heuristics (share-sorts/share-partitions **Aggregate** may be computed from previously computed aggregates: **Smallest-child/Cache-results/Amortize-scans)**

Multi-Way Array Aggregation (Bottom-up): A>B>C for ABC

BUC(Top-down): **A Shell-Fragment Approach** (High-dimensional OLAP) **Online Query Computation with Shell-**

- Reducing memory and I/O
  - Suppose we scan using order: 13 – 29 – 45 – 61 – 9 – 25 – ...



Example: A: 4000, B: 400, C: 40 Chunk: 1000 x 100 x 10

memory: minimum

- 40 x 400 (BC) + 40 x 1000 (AC) + 100 x 1000 (AB) = 156,000 units

Example: Let the cube aggregation function be count

TID	A	B	C	D	E
1	a1	b1	c1	d1	e1
2	a1	b2	c1	d2	e1
3	a1	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

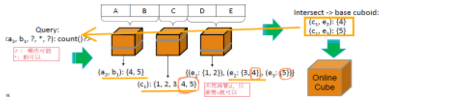
Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Divide the 5-D table into 2 shell fragments: (A, B, C) and (D, E)

Build traditional invert index or RID list (**1-D**)

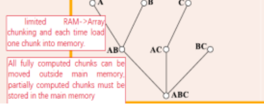
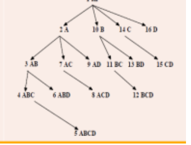
Fragments: Frag-Shells -> Offline & Online

总结：Multiway: dense full cube. no iceberg, no high-D; **BUC:** ice, no high-D; **shell:** high-D, no dense full cube, no large cube


A
B
C
D
E
Query: (a2, b1, ?, *, ?): count()?
Intersect -> base cuboid:
(c1, e2): {4}
(c1, e3): {5}
(a2, b1): {4, 5}
(c1): {1, 2, 3, 4, 5}
{(e1: {1, 2}), (e2: {3, 4}), (e3: {5})}
Online Cube

Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is: $\boxed{O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)}$

- For F < 5, the growth is sub-linear

	multiway (Partition arrays into chunks (a small subcube which fits in memory))	BUC (Divides dimensions into partitions and facilitates iceberg pruning)
Input format	Multi-dimensional array	Relational database
Good for	Full cube (for small number of dimensions)	Iceberg cube (large number of dimensions)
Key idea	Simultaneously Aggregation	Partition and sort
Calculation direction	 limited RAM->Array chunking and each time load one chunk into memory. All fully computed chunks can be moved outside main memory, partially computed chunks must be stored in the main memory	

1) Suppose a data relation has 100 attributes and 106 tuples. Each attribute has 50 distinct values. Suppose each cell takes 16 bytes of space, and the shell-fragments are all 4 dimensions. (i) [5] What is the size (in bytes) of one pre-computed shell-fragment of size 4? Answer: 16 * (1 + 4 *50 + 6*50^2 + 4* 50^3) Bytes. For the 0-D (apex) cuboid, there is 1 cell. For each 1-D cuboid, there are 50 *1*1*1 = 50 cells. There are 4 such cuboids. For each 2-D cuboid, there are 50*50*1*1 = 502 cells. There are 6 such cuboids. For each 3-D cuboid, there are 50*50*50*1 = 503 cells. There are 4 such cuboids. The total size of these cells is 16* (1 + 4*50 + 6*50^2 + 4* 50^3), or equivalently, 16* ((50 + 1)4- 50*4). Note that if we also count the base cells, since the number of tuples is smaller than 504, we can store the data in 106 cells. We will add 16*106 to the size. (ii) [4] If an OLAP query contains 2 instantiated variables and 6 inquired variables, what is the number of shell fragments this query must access in the best case and in the worst case, respectively? Answer: Best case: 2, worst case: 8. There are 8 dimensions we care about. In the best case, these 8 dimensions happen to be in 2 shell fragments since there are 4 dimensions each. In the worst case, all these 8 dimensions are in different shell fragments and we need to access 8 shell fragments.

2)Cannot support the following operations efficiently? i. Computing an iceberg cube [[Multiway cannot support, aggregation bottom-up and Apriori principle/pruning cannot. ii. Processing an OLAP query on 30 dimensions [Multiway and BUC – 30 dimensions is too many]

3)list one best method and another worst (a) computing a dense full cube of low dimensionality (e.g., less than 6 dimensions), B: multiway. W: shell-fragment (since most part of cube is not precomputed) (b)performing OLAP operations in a high-dimensional database (e.g., over 50 dimen-sions) B: shell-fragment, W: the other (c)computing a large iceberg cube of around 10 dimensions. B: BUC W: multiway

4)CubeSparse Sparse array compression: Use chuck to partition the data, and use (chunk id, offset) to store only those cells contain (nonempty) values.

5) ** Cube & Cuboid** Suppose the base cuboid of a data cube contains two cells(a1; a2; a3; a4; : : : ; a10) : 1, (a1; b2; a3; b4; : : : ; b10) : 1 where ai 6= bi for any I i. [3] How many nonempty cuboids are there in this data cube? Answer: 2^10. Since we have 10 dimensions with no concept hierarchy, there are 2^10 cuboids and all of them should not be empty. ii. [3] How many (nonempty) aggregate closed cells are there in this data cube? Answer: 1. There are 3 closed cells, including the two base cells and (a1; _; a3; _; a5; _; a7; _; a9; _). But only the latter one is a aggregated closed cell. iii. [3] How many (nonempty) aggregate cells are there in this data cube? Answer: 2014. For each base cell, there are 2^10 1 aggregated cells. However, there are 25 cells that are counted twice since there are 5 common dimensions. Therefore, the total number of nonempty aggregate cells is 2*(2^10-1)-2^5 = 2014. iv. [3] If we set minimum support = 2, how many (nonempty) aggregate cells are there in the corresponding iceberg cube? Answer: 2^5. These two base cells have common value in 5 dimensions; therefore, there are 2^5 nonempty cells with support = 2 and all of them are aggregate cells.

6) A cell c is closed if any descendent cell d have a smaller measure than c. measure(closed cell) <= #(base cell)

(a1,a2,c3,...,ck),(a1,b2,c3,...,ck),(b1,a2,c3,...,ck), (b1,b2,c3,...,ck)

* Cuboids = 2^k, given k dimensions

aggregate Closed cells: (a1,,c3,...,ck) : 2, (b1,*,c3,...,ck) : 2, (*,a2,c3,...,ck) : 2, (*,b2,c3,...,ck) : 2, (*,*,c3,...,ck) : 4

*aggregate cells: 4×(2^k-1)−4×2^(k−2)−3×2^(k−2)

Hint: "(a1,*,c3,...,ck) : 2" merges 2 × 2k−2 cells into 1 × 2k−2 cells and thus net loss is 2k−2 cells. There are 4 such cells. Thus their total loss is 4 × 2k−2 cells. "(*,*,c3,...,ck) : 4" merges 4 × 2k −2 cells into one 2k−2 cells and thus net loss is 3 × 2k−2.

*If we set min_support = 2, how many aggregate cells are there in the corresponding iceberg cube? 5 × 2^(k−2) since there are 5 aggregate closed cells, each will cover 2k−2 aggregate cells.

7) Explain why good intra-cuboid expansion may enhance the quality of drill-down in sampling cube, but a bad intra-cuboid expansion may reduce its quality. [Enhance quality: if we expand on a dimension that is not correlated with the target measure, we have more data available (so we can have tighter confidence bounds) Reduce quality: if we expand on a dimension correlated with the target measure, the new data will skew the target measure. We effectively end up answering a different question than the one posed]

8) What are the major challenges to support OLAP on sampling data? [Outline a method that may support such operations effectively. Major challenges: Some drilling down cells may contain few or no data. Method: Intra or inter-cuboid expansion to combine with other cells whose attributes has low correlations with the dimensions interested.]

Mining Frequent Patters, Association and Correlations(char6)

1.Transactional Database(TDB): itemset, k-itemset, absolute/relative support. Frequent: if the support of X is no less than a minsup threshold. **Association Rule:**

- We first compute the following two metrics, s and c.
  - Support of $X \cup Y$
    - Ex. s{Diaper, Beer} = 3/5 = 0.6 (i.e., 60%)
  - Confidence of $X \rightarrow Y$
    - The *conditional probability* that a transaction containing X also contains *Y* ≠
    - $c = sup(X \rightarrow Y) / sup(X)$
    - Ex. *c* = sup{Diaper, Beer}/sup{Diaper} = ¾ = 0.75

Solution 1: **Closed patterns**: A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ⊃ X, *with the same support* as X

Ex. TDB$_1$: T$_1$: {a$_1$, …, a$_{50}$}; T$_2$: {a$_1$, …, a$_{100}$} reconstruct

Suppose *minsup* = 1. How many closed patterns does TDB$_1$ contain?

- Two: P$_1$: "{a$_1$, …, a$_{50}$}: 2"; P$_2$: "{a$_1$, …, a$_{100}$}: 1"

X: {a}   {a, b}, {a, c} super-pattern

Solution 2: **Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X , ~~same support~~

Difference from close-patterns?

Do not care the real support of the sub-patterns of a max-pattern \ lss the supo

Let Transaction DB TDB$_1$: T$_1$: {a$_1$, …, a$_{50}$}; T$_2$: {a$_1$, …, a$_{100}$}

Suppose *minsup* = 1. How many max-patterns does TDB$_1$ contain?

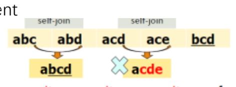
- One: P: "{a$_1$, …, a$_{100}$}: 1"

{a$_5$, a$_{55}$} is freq

closed pattern: lossless compression/ Max-pattern: lossy

2.Downward Closure Property Apriori: any subset of a frequent itemset must be frequent

```
// Step 1: Joining
for each p in $F_{k-1}$
    for each q in $F_{k-1}$
        if $p.item_1 = q.item_1, …, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$ {
            c = join(p, q)
// Step 2: pruning
            if has_infrequent_subset(c, $F_{k-1}$)
                continue    // prune
            else add c to $C_k$
    }
```

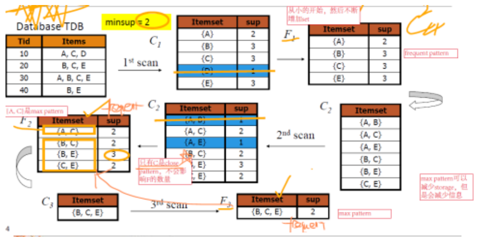


因为cd是新的，所以我们是希望仍然保留cd在一起

zabc zabd

zabcd

因为cd是新的，所以希望保留。然后寻找zabcd的subset
zacd zbcd zbcd



Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

C$_1$ (1st scan)

Itemset	sup
{A}	2
{B}	3
{C}	3
~~{D}~~	~~1~~
{E}	3

F$_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C$_2$

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

C$_2$ (2nd scan)

Itemset	sup
~~{A, B}~~	~~1~~
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

F$_2$

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C$_3$

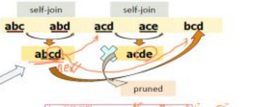
Itemset
{B, C, E}

F$_3$ (3rd scan)

Itemset	sup
{B, C, E}	2

How to generate candidates?
- Step 1: self-joining F_k
- Step 2: pruning

Example of candidate-generation
- F_3 = {*abc, abd, acd, ace, bcd*}
- Self-joining: F_3*F_3
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *acde* is removed because *ade* is not in F_3
- C_4 = {*abcd*}



上一页的实现：
AC BC BE CE
（AC：alone，被排除
BC + BE -> BCE
CE 不可以 也排除）

AC BC BE CE → BCE

Partitioning: any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB

Direct Hashing and Pruning: different itemsets may have the same hash value, can speed up while the size of buckets 重要

3.Vertical Data Format:

Using diffset to accelerate mining reduce storage
- Only keep track of differences of tids
- t(e) = {T$_{10}$, T$_{20}$, T$_{30}$}, t(ce) = {T$_{10}$, T$_{30}$} → Diffset (ce, e) = {T$_{20}$}

Method: Scan DB twice (A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*)

Scan 1: Partition database so that each partition can fit in main memory (why?)
- Mine local frequent patterns in this partition

Scan 2: Consolidate global frequent patterns
- Find global frequent itemset candidates (those frequent in at least one partition)
- Find the true frequency of those candidates, by scanning TDB$_i$ one more time

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

4.FP-Tree

1. Scan DB once, find single item frequent pattern:
   Let min_support = 3
   f:4, a:3, c:4, b:3, m:3, p:3
2. Sort frequent items in frequency descending order, f-list   F-list = f-c-a-b-m-p
3. Scan DB again, find the ordered frequent itemlist for each transaction

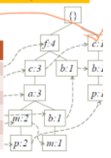
TID	Items in the Transaction	Ordered, frequent itemlist
100	{f, a, c, d, g, i, m, p}	f, c, a, m, p
200	{a, b, c, f, l, m, o}	f, c, a, b, m
300	{b, f, h, j, o, w}	f, b
400	{b, c, k, s, p}	c, b, p
500	{a, f, c, e, l, p, m, n}	f, c, a, m, p

4. For each transaction, insert the ordered frequent itemlist into an FP-tree, with shared sub-branches merged, counts accumulated

After inserting all the frequent itemlists

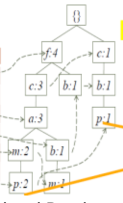
Header Table

Item	Frequency	header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



min_support = 3

Item	Frequency	Header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



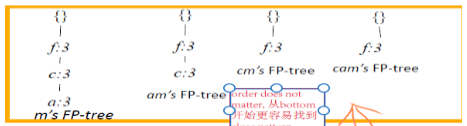
Conditional database of each pattern

Item	*Conditional database*
c	f:3
a	fc:3
b	fca:1, f:1, c:1
m	fca:2, fcab:1
p	fcam:2, cb:1

f m:3 m=3

pc=3

Mine Each Conditional Database recursively:



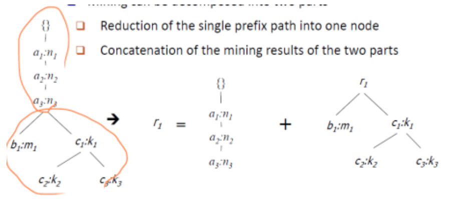
order does not matter, 从bottom开始更容易找到close pattern

Actually, for single branch FP-tree, all the frequent patterns can be generated in one shot

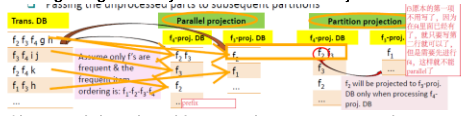
m: 3
fm: 3, cm: 3, am: 3
fcm: 3, fam:3, cam: 3
fcam: 3

Single Prefix Path in FP-tree;

- Reduction of the single prefix path into one node
- Concatenation of the mining results of the two parts



Scaling FP-growth by Item-Based Data Projection


Trans. DB
f2 f3 f4 g h
f3 f4 i j
f2 f4 k
f1 f3 h
Assume only f's are frequent & the frequent item ordering is: f1-f2-f3-f4
Parallel projection
f4-proj. DB
f3-proj. DB
Partition projection
f4-proj. DB
f3-proj.
prefix
f2 will be projected to f3-proj. DB only when processing f4-proj. DB
f3原本的第一项不用写了，因为在f4里面已经有了，就只要写第二行就可以了，但是需要先进行f4，这样就不能parallel了

Closet" mining closed itemsets by pattern-growth

5.Pattern Evaluation: Strong Association Rule

6.Interstingess Measure:

Lift(dependent/correlated)

$$lift(B,C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$

X2

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

Lift(B, C) may tell how B and C are correlated
- Lift(B, C) = 1: B and C are independent
- > 1: positively correlated
- < 1: negatively correlated

	B	¬B	Σ_{row}
C	400 (450)	350 (300)	750
¬C	200 (150)	50 (100)	250
Σ_{col}	600	400	1000

7.Null-Invariance: value does not change with the # of null-transactions。改变 null-transactions 时 lift 和 x2 会变，所以不是 null-invariant. Not good to evaluate data.

8. Imbalance Ratio with Kulczynski Measure:

$$IR(A,B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)}$$

****Null invariant**** Suppose a transaction database contains N transactions, ct transactions contain both coffee and tea, ct transactions contain coffee but not tea, ct transactions contain tea but not coffee, and ct transactions contain neither tea nor coffee. At what condition that Lift may not be a good measure to indicate correlations between coffee and tea? and why? 【】

Condition: If the number of ct transactions is substantially larger than the remaining # of transactions, LIFT is not a good measure. / Reasoning: Lift is not null-invariant. That is, although LIFT is affected by the number of ct transactions, the correlation should not be (because transactions that do not contain c or t tell us nothing about the correlation - so they shouldn't affect the measure).

** **Null invariant** **Explain why null-invariance property is important in the study which authors are"correlated? 【】 Since most authors are not coauthoring papers, null value is very high. Null-invariance is critical otherwise the "correlation value could be greatly influenced by null values.

(1) mine FPs from the new DB; (2) get counts of those patterns frequent in new DB but not in old DB by scanning the old DB once, (3) get counts of those that are frequent in old DB by not in new DB. The merge counts will be used to just which patterns are globally frequent

Measure	Definition	Range	Null-Invariant?
$\chi^2(A,B)$	$\sum_{i,j}\frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0, \infty]$	No
$Lift(A,B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$	$[0, \infty]$	No
$Allconf(A,B)$	$\frac{s(A \cup B)}{max\{s(A), s(B)\}}$	$[0, 1]$	Yes
$Jaccard(A,B)$	$\frac{s(A \cup B)}{s(A)+s(B)-s(A \cup B)}$	$[0, 1]$	Yes
$Cosine(A,B)$	$\frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$	$[0, 1]$	Yes
$Kulczynski(A,B)$	$\frac{1}{2}\left(\frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)}\right)$	$[0, 1]$	Yes
$MaxConf(A,B)$	$max\{\frac{s(A \cup B)}{s(A)}, \frac{s(A \cup B)}{s(B)}\}$	$[0, 1]$	Yes

Let n be the total number of transactions, and $count(\neg(A \cup B))$ be the number of null-transactions.

$$lift(A,B) = \frac{s(A \cup B)}{s(A) \times s(B)} = \frac{\frac{count(A \cup B)}{n}}{\frac{count(A)}{n} \times \frac{count(B)}{n}} = \frac{count(A \cup B) \times n}{count(A) \times count(B)} = \frac{count(A \cup B) \times (count(A \cup B) + count(\neg(A \cup B)))}{count(A) \times count(B)}$$

$$cosine(A,B) = \frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}} = \frac{\frac{count(A \cup B)}{n}}{\sqrt{\frac{count(A)}{n} \times \frac{count(B)}{n}}} = \frac{count(A \cup B)}{count(A) \times count(B)}$$

We can clearly see that *cosine* is invariant with the number of null-transactions, while *lift* is not.

(b) [4'] Suppose that a data array has 3 dimensions A, B, C with the following sizes: $|A| = 400$, $|B| = 100$ and $|C| = 80$. The 3-D data array is divided into small chunks. Dimension A is divided into 4 equally sized partitions. Dimensions B and C are divided into 2 equally sized partitions respectively. Thus, there are totally 16 3-D chunks. The sizes of each 3-D chunk on dimensions A, B, and C are 100, 50, and 40 respectively. See Figure 1.

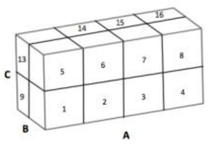


Figure 1: A 3-D data array with dimensions A, B and C. This data array is divided into 16 smaller chunks.

Now we want to use **Multiway Array Aggregation Computation** to materialize the 2-D cuboids AB, AC and BC. If we scan the chunks in the order of 1, 2, 3, 4, 5…., 15, 16 when materializing the 2-D cuboids AB, AC and BC, to avoid reading a 3-D chunk into memory repeatedly, what is the minimum memory requirement for holding all the related 2-D planes? Show your result with important intermediate steps in calculation. (*Hint: When calculating the minimum memory requirement, you are* **not** *required to consider the memory cost for the* ***3-D chunk*** *which is read into the memory for scanning.*)

Solution.
If we scan the cube base on the order of 1, 2, 3...16, we have: 400×80 (for the whole AC plane) + 400×50 (for one row of the AB plane) + 50×40 (for one BC plane chunk) = 54,000 memory units. (Do not consider the 3-D chunk)

2. [30] Frequent Pattern and Association Mining

(a) [8'] The price of each item in a store is nonnegative. For each of the following cases, identify the type of constraint they represent and briefly discuss how to mine such association rules efficiently with frequent pattern mining algorithms.

i. [4'] Containing at least one Nintendo game.
ANSWER: **The constraint is succinct and monotonic. This constraint can be mined efficiently using FP-growth as follows.**
• All frequent Nintendo games are listed at the end of the list of frequent items L.
• Only those conditional pattern bases and FP-trees for frequent Nintendo games need to be derived from the global FP-tree and mined recursively.

ii. [4'] Containing one free item and other items the sum of whose prices is at least \$200.
ANSWER: **The constraint is monotonic. (Or, subconstraints "containing one free item" and "the sum of whose prices is less than \$200" are succinct and monotonic, respectively.) This constraint can be mined efficiently using FP-growth as follows.**
• Put all frequent free items at the end of the list of frequent items L.
• Only conditional pattern bases and FP-trees for frequent free items need to be derived from the global FP-tree and mined recursively. Other free items should be excluded from these conditional pattern bases and FP-trees.
• Once a pattern with items the sum of whose prices is at least \$200, no further constraint checking for total price is needed in recursive mining.
• A pattern as well as its conditional pattern base can be pruned if the sum of the price of items in the pattern and the frequent ones in the pattern base is less than \$200.

(b) [6] Suppose the base cuboid of a data cube contains only four cells

$(a_1, a_2, a_3, a_4, a_5, \ldots, a_{2k})$
$(b_1, a_2, b_3, a_4, b_5, \ldots, a_{2k})$
$(c_1, a_2, c_3, a_4, c_5, \ldots, a_{2k})$
$(d_1, d_2, d_3, d_4, d_5, \ldots, d_{2k})$

where $k \in \mathbb{N}_+$ (k is positive), $a_i \neq b_i \neq c_i \neq d_i$, $\forall i = 1, \ldots 2k$.

i. [2] If we set minimum support = 2, how many nonempty aggregate cells are there in the corresponding *iceberg cube*?
ii. [2] How many closed cells are there in the *iceberg cube*?

Note: Please show essential calculation steps.
Answer:
i. 2^k;
ii. 2, which are $(*, a_2, *, a_4, \ldots, a_{2k}), and(*, *, *, *, \ldots, *)$.

□

(c) [6]
i. [3] For the following tasks, which cube implementation method is better? multiway array cubing, BUC (bottom-up computation), or **neither**? Briefly explain.
A. Considering a data cube about sales in a small store, with five dimensions (customer, time, product, unit, price), fully materialize this data cube.
B. Computing a large iceberg cube of around 830 dimensions.
Answer:
A. multiway array cubing.
B. Neither. It is a problem with high dimension.