102046706 – Data Mining & Business Intelligence

# Unit-5
## Concept Description and Association Rule Mining

# Outline

- What is concept description?
- Market basket analysis
- Association Rule Mining
- Generating Rules
- Improved apriori algorithm
- Incremental ARM (Association Rule Mining)
- Associative Classification
- Rule Mining

# Concept description

- Data mining can be classified into two categories: **descriptive** data mining and **predictive** data mining.
- Descriptive data mining **describes the data set** in a **concise and summative manner** and **presents interesting general properties** of the data.
- Predictive data mining **analyzes the data** in order to **construct one or a set of models**, and attempts to **predict the behavior of new data sets**.
- Database is usually storing the large amounts of data in great detail. However users often **like to view** sets of **summarized data in concise, descriptive terms**.

# Concept description (Cont..)

The simplest kind of **descriptive data mining** is called **concept description.**

- A **concept** usually refers to a **collection of data** such as frequent_buyers, graduate_students etc.
- As a data mining task, concept description is **not** a simple **enumeration** (number of things done one by one) of the data.
- Concept description generates **descriptions** for **characterization and comparison** of the data it is also called class description.

# Concept description (Cont..)

- **Characterization** provides a **concise and brief summarization** of the data.
- While **concept or class comparison** (also known as discrimination) provides **discriminations** (inequity) comparing two or more collections of data.
- Example
  - Given the ABC Company database, for example, examining individual customer transactions.
  - Sales managers may prefer to view the data generalized to higher levels, such as summarized by **customer groups** according to **geographic regions, frequency of purchases per group** and **customer income**.

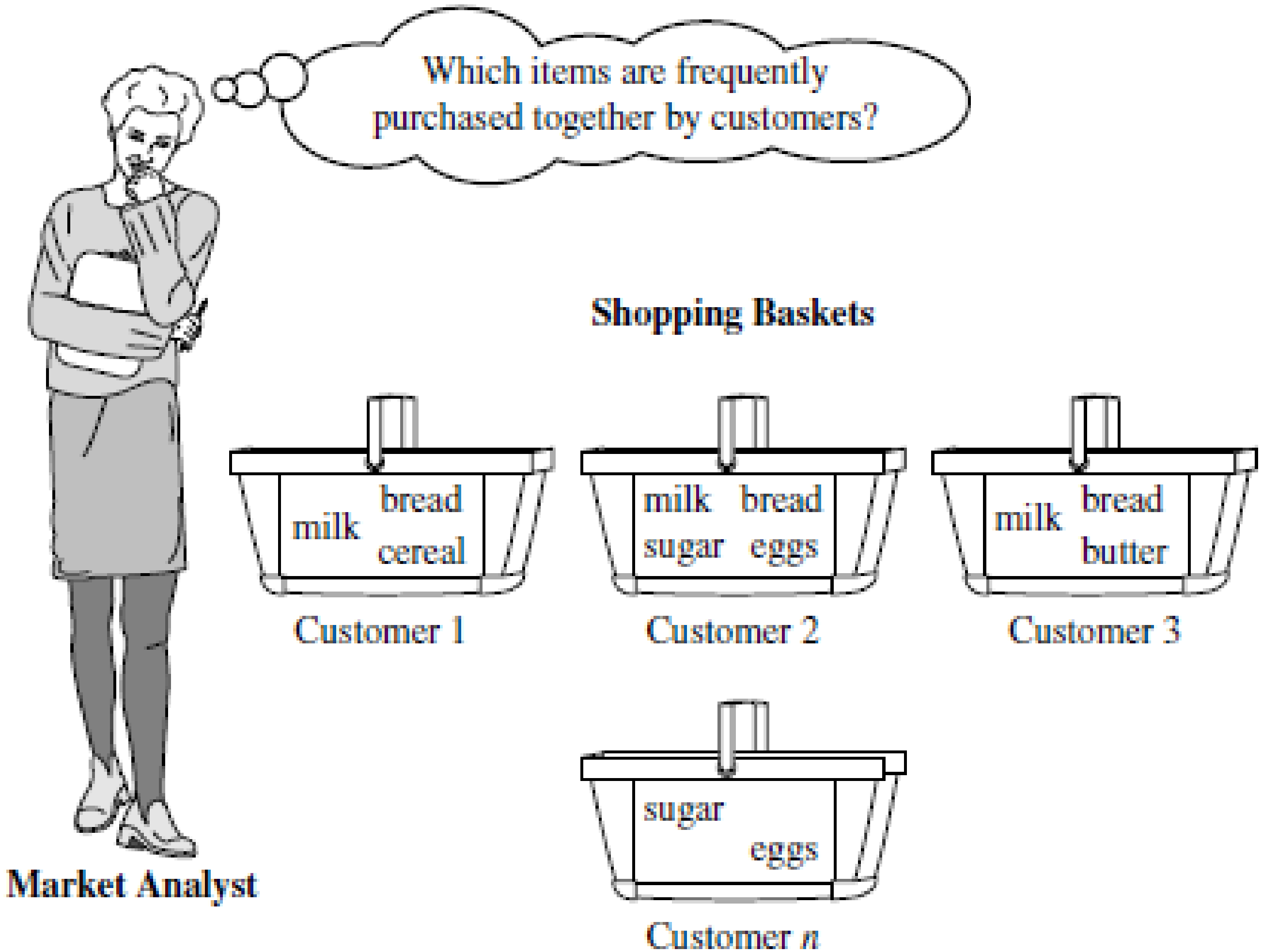
# Market basket analysis

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases.
- The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes such as
  - ✓ catalog design,
  - ✓ cross-marketing, and
  - ✓ customer shopping behavior analysis.

# Market basket analysis (Cont..)

- An example of frequent itemset mining is **market basket analysis.**
- This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.
- The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.
- For instance, if customers are buying milk, how likely are they to also buy bread on the same trip.

# Market basket analysis (Cont..)



# Association rule mining

- Given a set of transactions, we need rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.
- **Market-Basket transactions**

| TID | Items |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Chocolate, Pepsi, Eggs |
| 3 | Milk, Chocolate, Pepsi, Coke |
| 4 | Bread, Milk, Chocolate, Pepsi |
| 5 | Bread, Milk, Chocolate, Coke |

**Example of Association Rules**

{Chocolate} → {Pepsi},
{Milk, Bread} → {Eggs, Coke},
{Pepsi, Bread} → {Milk}

# Association rule mining (Cont..)

- **Itemset**
  - A collection of **one or more items**
    - **E.g.** : {Milk, Bread, Chocolate}
  - k-itemset

    An itemset that contains **k** items
- **Support count (σ)**
  - **Frequency** of occurrence of an **itemset**
    - **E.g.** σ({Milk, Bread, Chocolate}) = 2
- **Support**
  - **Fraction of transactions** that **contain an itemset**
    - **E.g.** s({Milk, Bread, Chocolate}) = 2/5
- **Frequent Itemset**
  - An itemset whose **support is greater than or equal to a minimum support** threshold

| TID | Items |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Chocolate, Pepsi, Eggs |
| 3 | Milk, Chocolate, Pepsi, Coke |
| 4 | Bread, Milk, Chocolate, Pepsi |
| 5 | Bread, Milk, Chocolate, Coke |

# Association rule mining (Cont..)

- **Association Rule**
  - An **implication expression** of the form **X → Y**, where X and Y are itemsets
    - **E.g.**: {Milk, Chocolate} → {Pepsi}
- **Rule Evaluation**
  - Support (s)
    - **Fraction of transactions** that **contain both X and Y**
  - Confidence (c)
    - Measures **how often items in Y appear** in **transactions that contain X**

# Association rule mining (Cont..)

| TID | Items |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Chocolate, Pepsi, Eggs |
| 3 | Milk, Chocolate, Pepsi, Coke |
| 4 | Bread, Milk, Chocolate, Pepsi |
| 5 | Bread, Milk, Chocolate, Coke |

<u>Example:</u>

Find support & confidence for **{Milk, Chocolate} ⇒ Pepsi**

$$s = \frac{\sigma(Milk, Chocolate, Pepsi)}{|T|} = \frac{2}{5} = \mathbf{0.4}$$

$$c = \frac{\sigma(Milk, Chocolate, Pepsi)}{\sigma(Milk, Chocolate)} = \frac{2}{3} = \mathbf{0.67}$$

# Association rule mining - Example

| TID | Items |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Chocolate, Pepsi, Eggs |
| 3 | Milk, Chocolate, Pepsi, Coke |
| 4 | Bread, Milk, Chocolate, Pepsi |
| 5 | Bread, Milk, Chocolate, Coke |

Calculate **Support** & **Confidence:**

{Milk, Chocolate} → {Pepsi}
{Milk, Pepsi} → {Chocolate}
{Chocolate, Pepsi} → {Milk}
{Pepsi} → {Milk, Chocolate}
{Chocolate} → {Milk, Pepsi}
{Milk} → {Chocolate, Pepsi}

**Answer**

Support (s) : **0.4**

{Milk, Chocolate} → {Pepsi} c = **0.67**
{Milk, Pepsi} → {Chocolate} c = **1.0**
{Chocolate, Pepsi} → {Milk} c = **0.67**
{Pepsi} → {Milk, Chocolate} c = **0.67**
{Chocolate} → {Milk, Pepsi} c = **0.5**
{Milk} → {Chocolate, Pepsi} c = **0.5**

# Association rule mining (Cont..)

- A common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

  1. **Frequent Itemset Generation**
     - The objective is to find all the item-sets that satisfy the minimum support threshold.
     - These itemsets are called **frequent itemsets**.
  2. **Rule Generation**
     - The objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step.
     - These rules are called **strong rules**.

# Apriori algorithm

- **Purpose**: The Apriori Algorithm is an influential algorithm for mining **frequent itemsets** for Boolean **association rules**.
- **Key Concepts**:
  - **Frequent Itemsets**:

    The sets of item which has **minimum support** (denoted by $L_i$ for ith-Itemset).
  - **Apriori Property**:

    Any **subset of frequent itemset must** be **frequent**.
  - **Join Operation**:

    To find $L_k$, a set of candidate k-itemsets is generated by joining $L_{k-1}$ itself.

# Apriori algorithm (Cont..)

- **Find the frequent itemsets**
  - The **sets of items that have minimum support** and a **subset of a frequent itemset must** also be a **frequent itemset (Apriori Property).**
  - **E.g.** if **{AB}** is a frequent itemset, both {A} and {B} should be a frequent itemset.
  - Use the frequent item sets to generate association rules.
- **The Apriori Algorithm : Pseudo code**
  - **Join Step**: $C_k$ is generated by joining $L_{k-1}$ with itself
  - **Prune Step**: Any (k-1) itemset that is not frequent cannot be a subset of a frequent k-itemset

# Apriori algorithm steps (Cont..)

- **Step 1:**
  - Start with itemsets containing just a **single item (Individual items).**
- **Step 2:**
  - Determine the support for itemsets.
  - **Keep** the itemsets that **meet your minimum support threshold** and **remove** itemsets that **do not support minimum support**.
- **Step 3:**
  - Using the itemsets you have kept from Step 1, **generate all the possible itemset combinations**.
- **Step 4:**
  - **Repeat** steps 1 & 2 until there are **no more new itemsets**.

# Apriori algorithm - Pseudo code (Cont..)

$C_k$: Candidate itemset of size k

$L_k$: Frequent itemset of size k

$L_1$= {frequent items};

**for** (k = 1; $L_k$ != ∅; k++) **do begin**

$C_{k+1}$ = candidates generated from $L_k$;

**for each** transaction **t** in database do

Increment the count of all candidates in $C_{k+1}$

That are contained in **t**

$L_{k+1}$ = candidates in $C_{k+1}$ with min_support

**end**

return $\cup_k L_k$;

# Apriori algorithm - Example

**Minimum Support = 2**

| TID | Items |
|---|---|
| 100 | 1 3 4 |
| 200 | 2 3 5 |
| 300 | 1 2 3 5 |
| 400 | 2 5 |

Scan D →

$C_1$

| ItemSet | Min. Sup |
|---|---|
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {4} | 1 |
| {5} | 3 |

→

$L_1$

| ItemSet | Min. Sup |
|---|---|
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {5} | 3 |

$C_2$

| Itemset |
|---|
| {1 2} |
| {1 3} |
| {1 5} |
| {2 3} |
| {2 5} |
| {3 5} |

← Scan D

$C_2$

| Itemset | Min. Sup |
|---|---|
| {1 2} | 1 |
| {1 3} | 2 |
| {1 5} | 1 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

←

$L_2$

| ItemSet | Min. Sup |
|---|---|
| {1 3} | 2 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

# Apriori algorithm - Example

**Minimum Support = 2**

$L_2$

| ItemSet | Min. Sup |
|---|---|
| {1 3} | 2 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

Scan D →

$C_3$

| ItemSet | Min. Sup |
|---|---|
| {1 2 3} | 1 |
| {1 2 5} | 1 |
| {1 3 5} | 1 |
| {2 3 5} | 2 |

→ $L_3$

| Items | Sup |
|---|---|
| {2 3 5} | 2 |

**Rules generation**

| Association Rule | Support | Confidence | Confidence (%) |
|---|---|---|---|
| **2 ^ 3 → 5** | 2 | 2/2 = **1** | **100 %** |
| **3 ^ 5 → 2** | 2 | 2/2 = **1** | **100 %** |
| **2 ^ 5 → 3** | 2 | 2/3 = **0.66** | **66%** |
| **2 = 3 ^ 5** | 2 | 2/3 = **0.66** | **66%** |
| **3 = 2 ^ 5** | 2 | 2/3 = **0.66** | **66%** |
| **5 = 2 ^ 3** | 2 | 2/3 = **0.66** | **66%** |

# Improve apriori’s efficiency

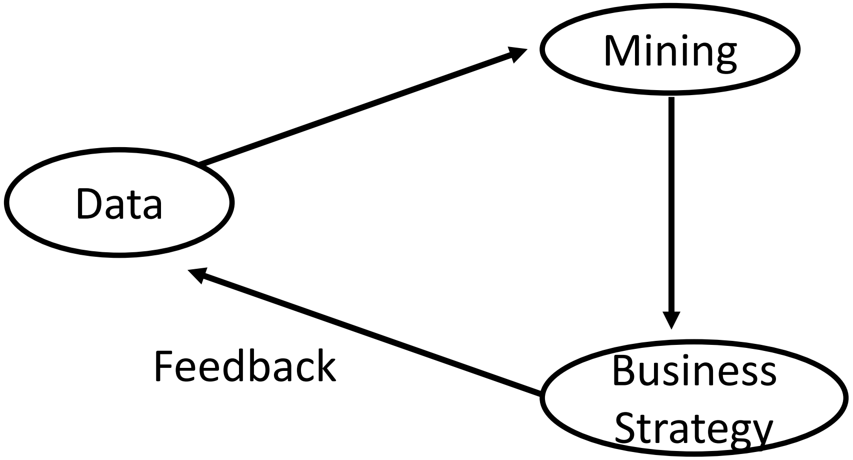
- **Hash-based itemset counting**: A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- **Transaction reduction**: A transaction that does not contain any frequent k-itemset is useless in subsequent scans.
- **Partitioning**: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.
- **Sampling**: Mining on a subset of given data, lower support threshold + a method to determine the completeness.
- **Dynamic itemset counting**: Add new candidate itemsets only when all of their subsets are estimated to be frequent.

# Incremental Mining of Association Rules

- It is noted that analysis of past transaction data can provide very valuable information on customer buying behavior, and thus improve the quality of business decisions.
- With the increasing use of the record-based databases whose data is being continuously added, updated, deleted etc.
- Examples of such applications include Web log records, stock market data, grocery sales data, transactions in e-commerce, and daily weather/traffic records etc.
- In many applications, we would like to mine the transaction database for a fixed amount of most recent data (say, data in the last 12 months).

# Incremental Mining of Association Rules

- Mining is not a one-time operation, a naive approach to solve the incremental mining problem is to re-run the mining algorithm on the updated database.



# FP-Growth Algorithm

- The FP-Growth Algorithm is proposed by Han.
- It is an **efficient and scalable** method for **mining the complete set of frequent patterns**.
- Using prefix-tree structure for storing information about frequent patterns named frequent-pattern tree (**FP-tree**).
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent item sets.

# FP-Growth Algorithm (Cont..)

- **Building the FP-Tree**
  1. Scan data to determine the support count of each item.
     - **Infrequent** items are **discarded**, while the frequent items are **sorted in decreasing support counts**.
  2. Make a second pass over the data to construct the FP-tree.
     - As the transactions are read, before being processed, **their items are sorted according to the above order**.

# FP-Growth Algorithm - Example

**FP-Tree Generation**

| TID | Transactions |
|---|---|
| 1 | A B C E F O |
| 2 | A C G |
| 3 | E I |
| 4 | A C D E G |
| 5 | A C E G L |
| 6 | E J |
| 7 | A B C E F P |
| 8 | A C D |
| 9 | A C E G M |
| 10 | A C E G N |

**Step:1**
Freq. 1-Itemsets.
**Min_Sup ≥ 2**

| Transactions |
|---|
| A : 8 |
| C : 8 |
| E : 8 |
| G : 5 |
| B : 2 |
| D : 2 |
| F : 2 |

**Remaining all O,I,J,L,P,M & N is with min_sup = 1**

**Step:2**
Transactions **with items sorted** based on frequencies, and **ignoring the infrequent items**.

**A C E B F**
**A C G**
**E**
**A C E G D**
**A C E G**
**E**
**A C E B F**
**A C D**
**A C E G**
**A C E G**

# FP-Tree after reading 1st transaction

**A C E B F**

A C G

E

A C E G D

A C E G

E

A C E B F

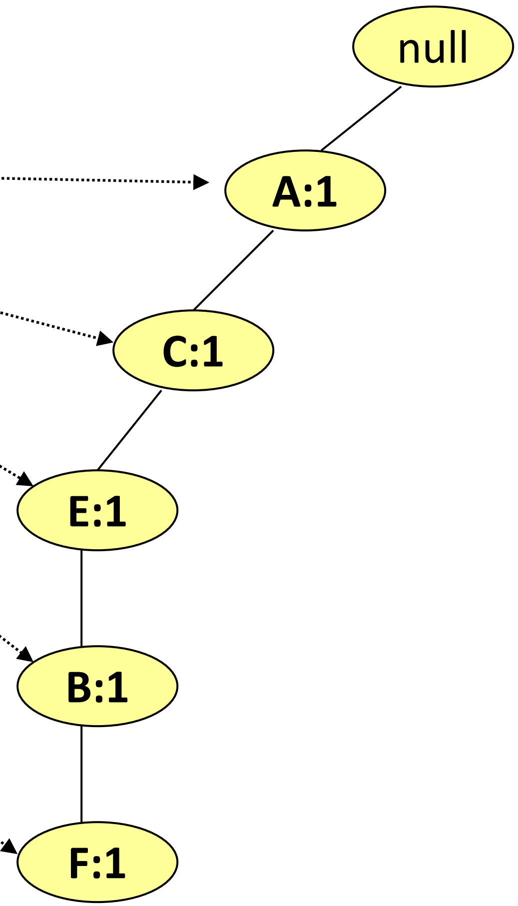
A C D

A C E G

A C E G

Header

| | |
|---|---|
| A:8 | |
| C:8 | |
| E:8 | |
| G:5 | |
| B:2 | |
| D:2 | |
| F:2 | |



# FP-Tree after reading 2nd transaction

A C E B F

**A C G**

E

A C E G D

A C E G

E

A C E B F

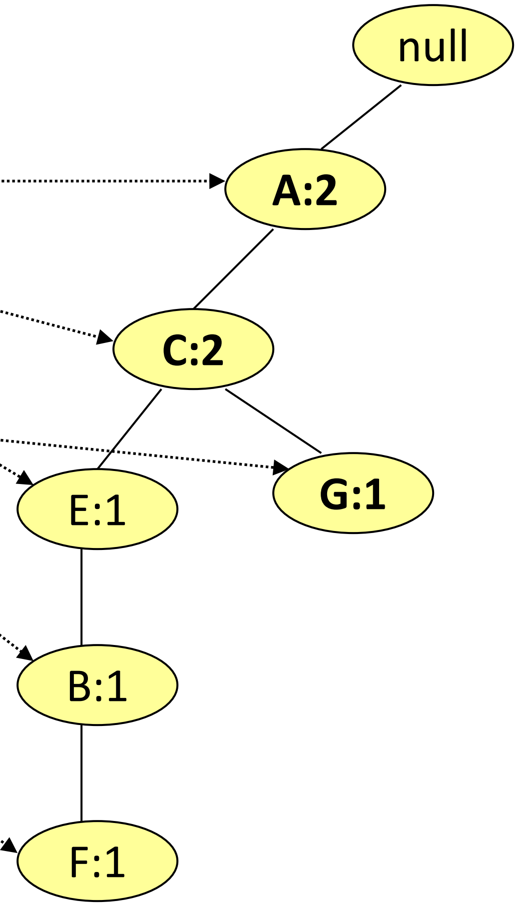
A C D

A C E G

A C E G

Header

| | |
|---|---|
| A:8 | |
| C:8 | |
| E:8 | |
| G:5 | |
| B:2 | |
| D:2 | |
| F:2 | |



# FP-Tree after reading 3rd transaction

A C E B F

A C G

$\boxed{E}$

A C E G D

A C E G

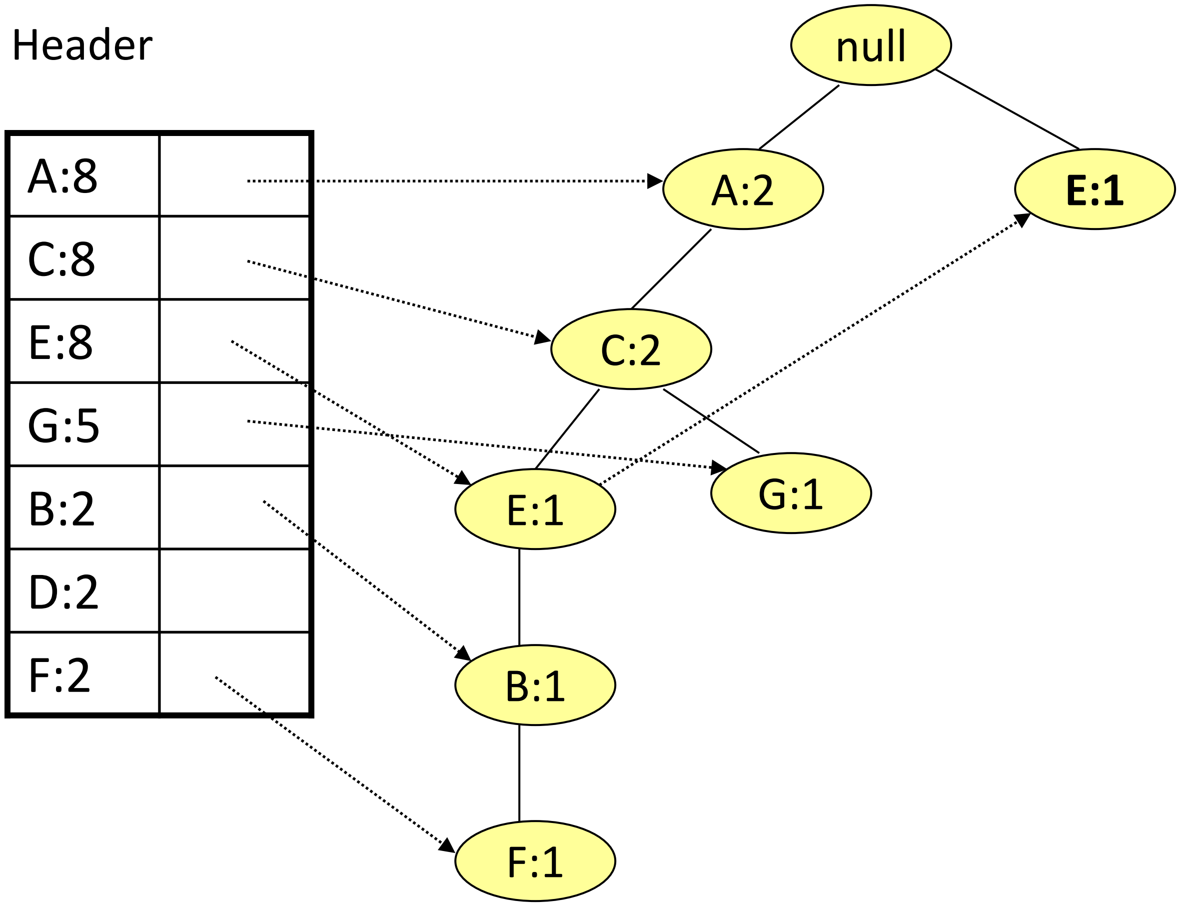
E

A C E B F

A C D

A C E G

A C E G



# FP-Tree after reading 4th transaction

A C E B F

A C G

E

**A C E G D**

A C E G

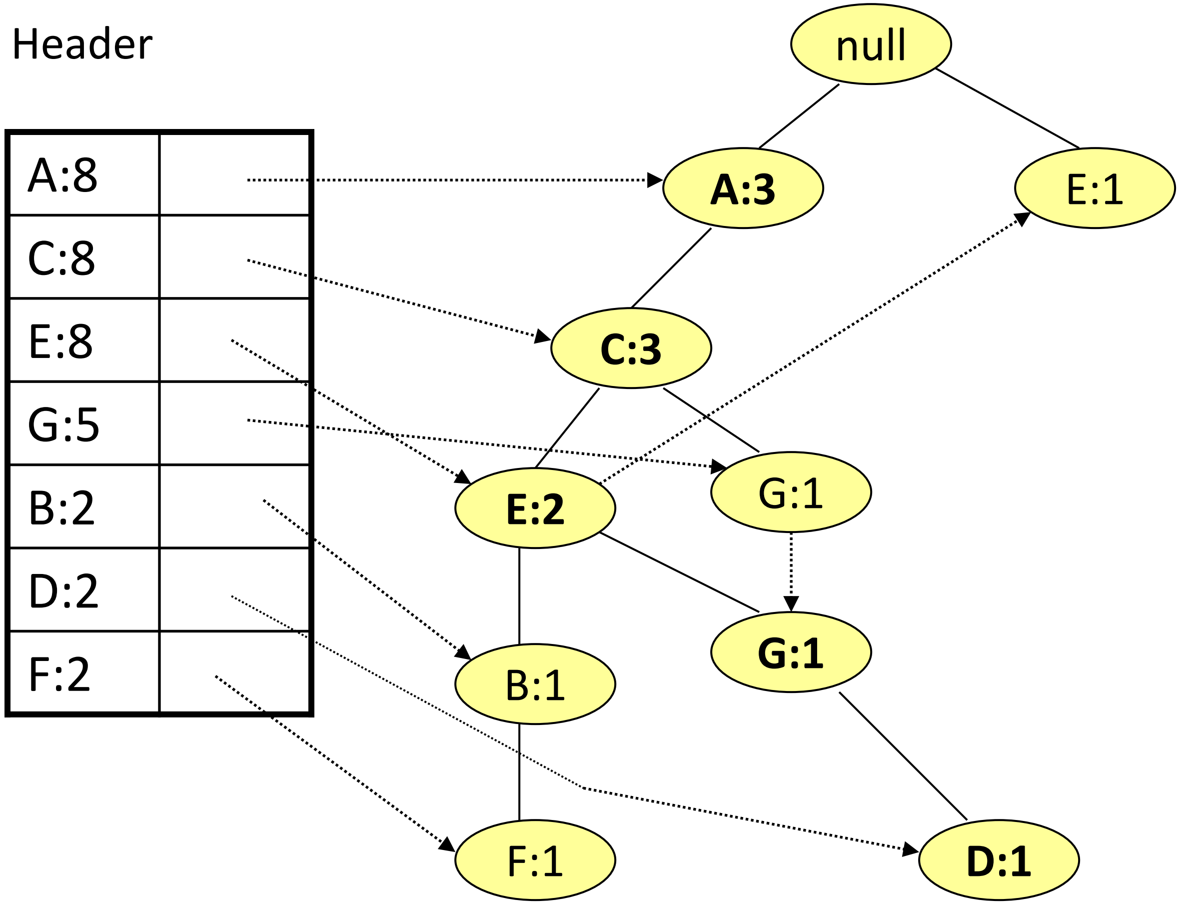
E

A C E B F

A C D

A C E G

A C E G



# FP-Tree after reading 5$^{th}$ transaction

A C E B F

A C G

E

A C E G D

**A C E G**

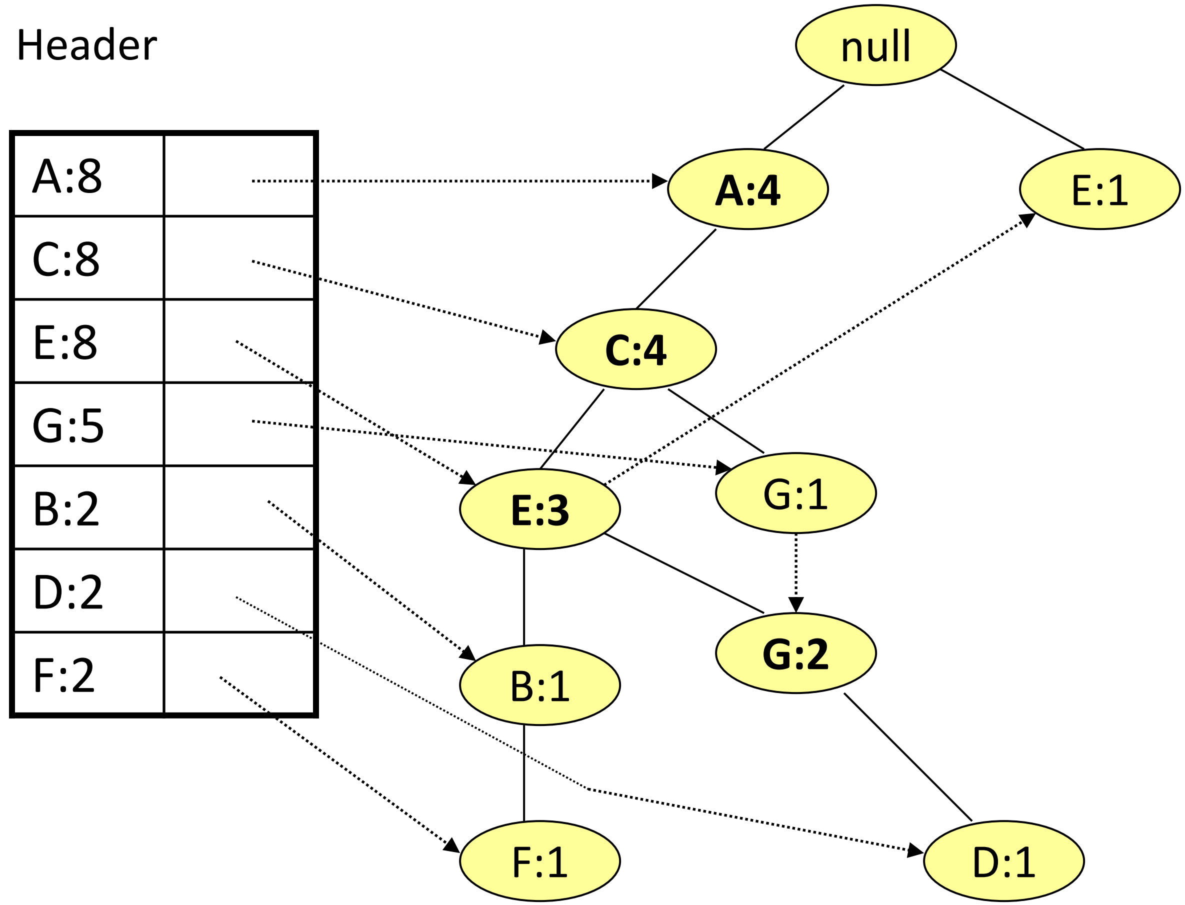
E

A C E B F

A C D

A C E G

A C E G



# FP-Tree after reading 6$^{th}$ transaction

A C E B F

A C G

E

A C E G D

A C E G

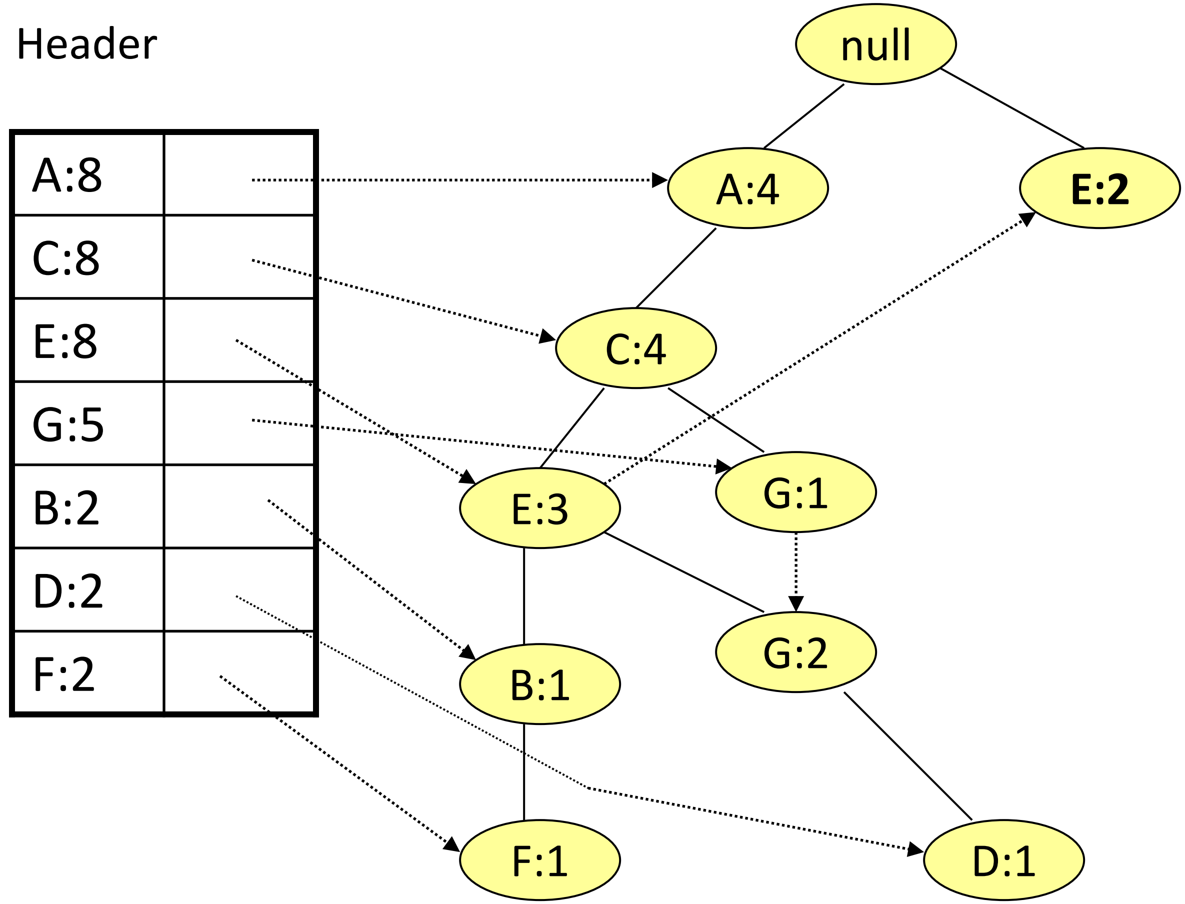
$\boxed{E}$

A C E B F

A C D

A C E G

A C E G



# FP-Tree after reading 7th transaction

A C E B F

A C G

E

A C E G D

A C E G

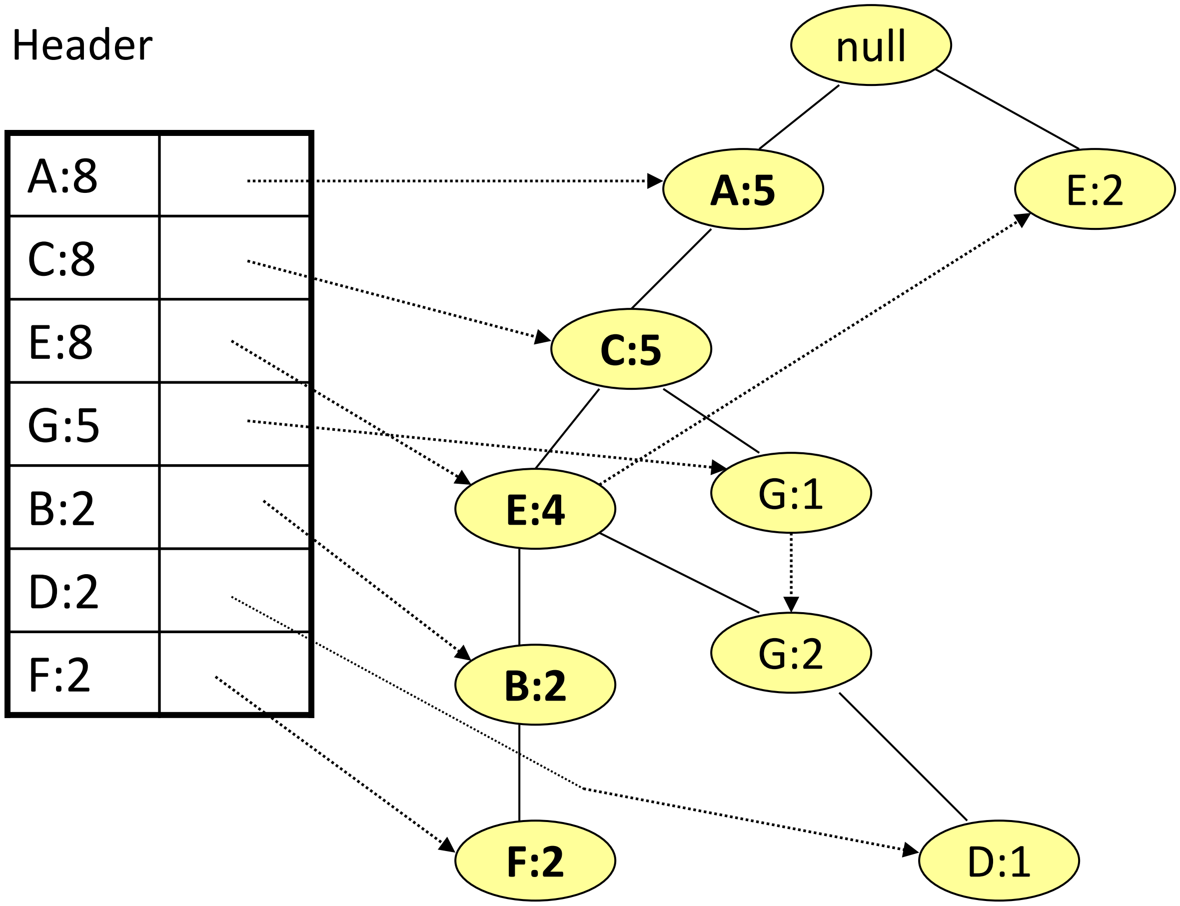
E

**A C E B F**

A C D

A C E G

A C E G



# FP-Tree after reading 8th transaction

A C E B F

A C G

E

A C E G D

A C E G

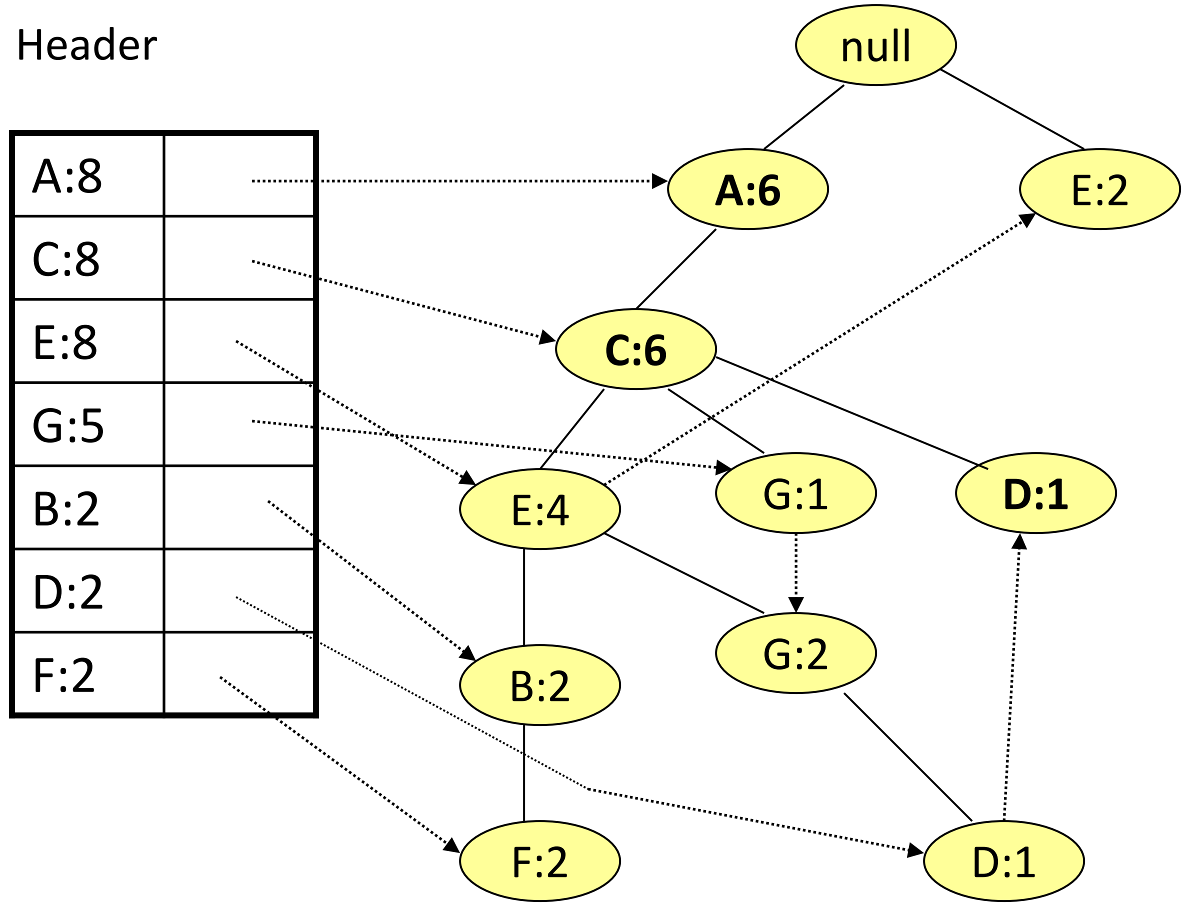
E

A C E B F

**A C D**

A C E G

A C E G



# FP-Tree after reading 9$^{th}$ transaction

A C E B F

A C G

E

A C E G D

A C E G

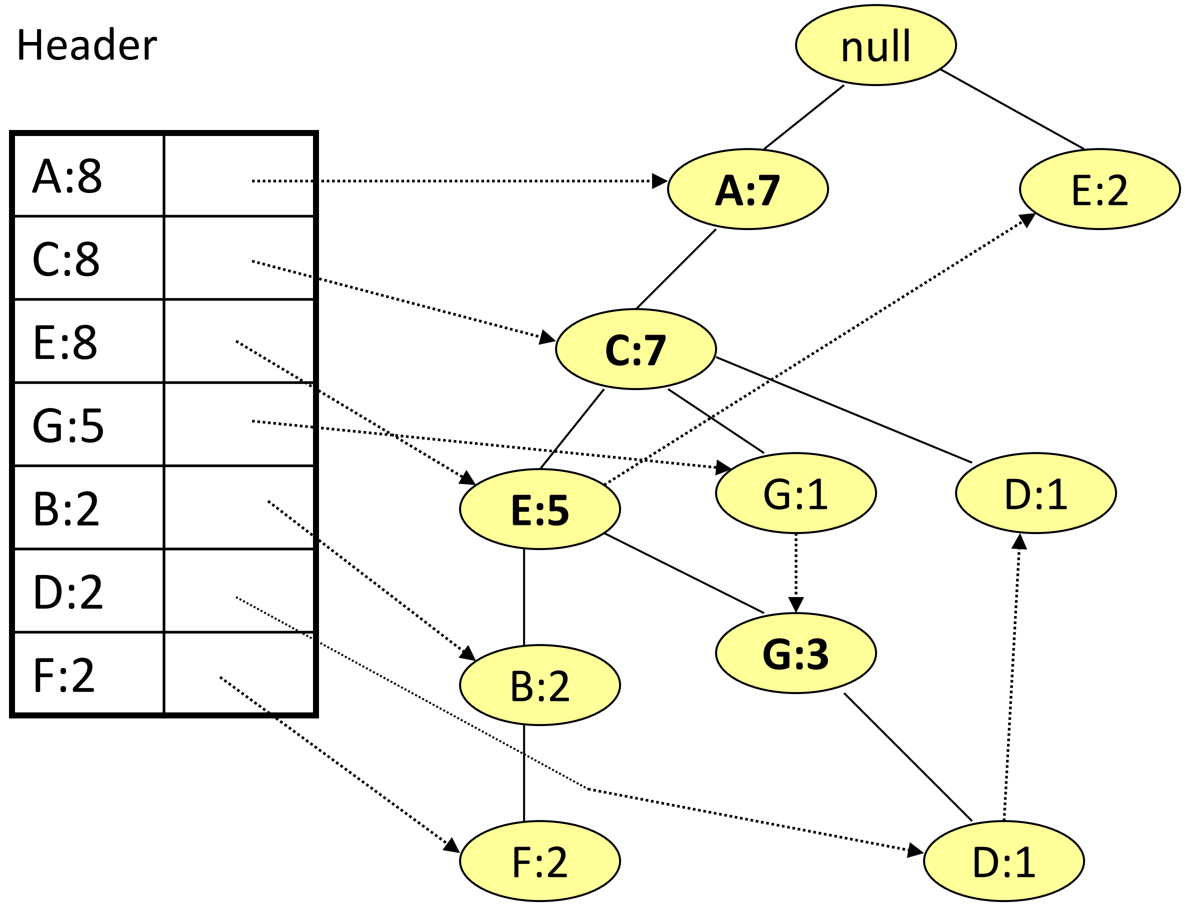
E

A C E B F

A C D

**A C E G**

A C E G



# FP-Tree after reading 10$^{th}$ transaction

A C E B F

A C G

E

A C E G D

A C E G

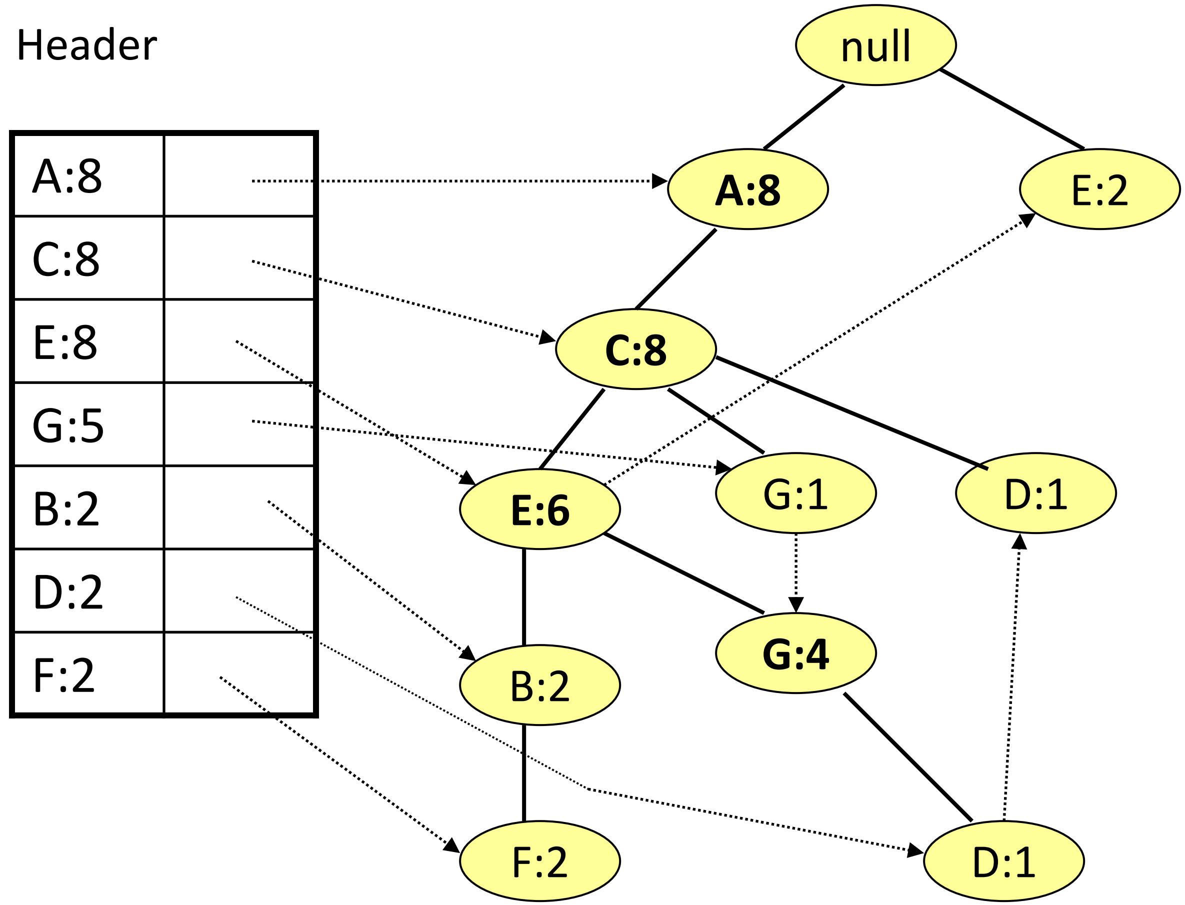
E

A C E B F

A C D

A C E G

**A C E G**



# FP-Growth algorithm - Example

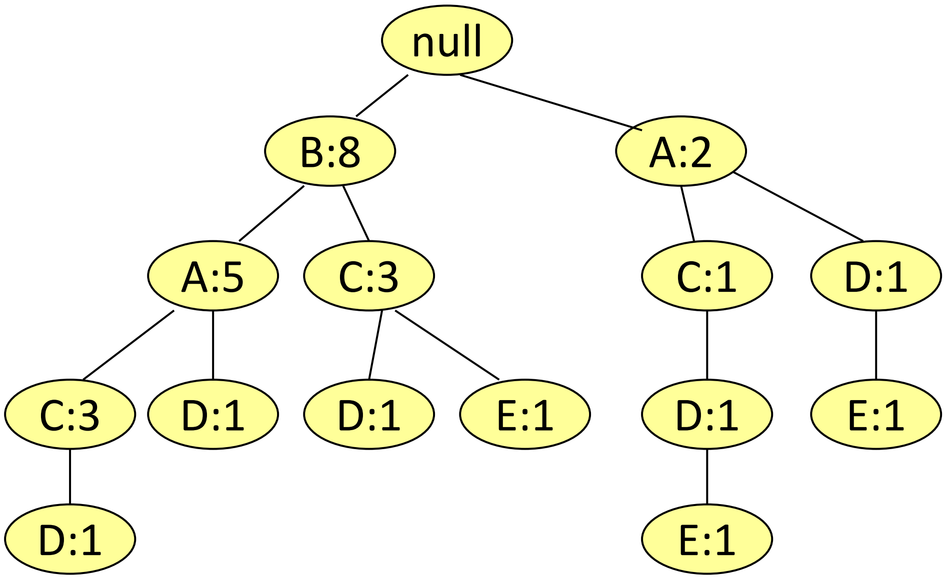
**Minimum Support >= 2**

| TID | Items |
|---|---|
| 1 | A B |
| 2 | B C D |
| 3 | A C D E |
| 4 | A D E |
| 5 | A B C |
| 6 | A B C D |
| 7 | B C |
| 8 | A B C |
| 9 | A B D |
| 10 | B C E |

**Header**

| Item | Support |
|---|---|
| B | 8 |
| A | 7 |
| C | 7 |
| D | 5 |
| E | 3 |

**FP-Tree**



# FP-Growth Example (Try it)

**Minimum Support = 3**

| TID | Items Bought |
|---|---|
| 100 | F A C D G I M P |
| 200 | A B C F L M O |
| 300 | B F H J O W |
| 400 | B C K S P |
| 500 | A F C E L P M N |

# FP-Growth Example - Answer

**FP-Tree Construction**

Header Table

| *Item* | *frequency* |
|---|---|
| ***f*** | *4* |
| ***c*** | *4* |
| ***a*** | *3* |
| ***b*** | *3* |
| ***m*** | *3* |
| ***p*** | *3* |