COMPARATIVE ANALYSIS OF FP-GROWTH AND ECLAT ALGORITHMS FOR FREQUENT ITEMSET MINING

RICHARD YONATHAN JULIO CLAY HENG - X25EC3019
NANDINI VENGAT - X25EC3021
NIVEETHITA A/P PANDIA RAJAN - A24CS0148
KALAITHARAN A/L PALANYVELU - A24CS0091

INTRODUCTION

Association rule mining finds relationships between items in large transaction datasets, such as “customers who buy X also tend to buy Y.” Algorithms FP-Growth are used because they avoid expensive candidate generation and multiple database scans, making the mining process faster and more scalable. FP-Growth compresses the data into a compact FP-Tree and then directly mines frequent itemsets from this structure, which is efficient for large, real-world applications like market basket analysis and recommendations

COMPARISON BETWEEN FP-GROWTH AND ECLAT

Feature	FP-Growth	ECLAT
Data Representation	Uses Horizontal Database Format (transactions stored as item lists)	Uses Vertical Database Format (items stored with Transaction ID sets)
Main Structure	Frequent Pattern Tree	Transaction ID Set
Searching / Mining Approach	Builds FP-tree then mines using conditional pattern base and recursive tree generation	Uses Depth-First Search and frequent itemset generation through set intersection
Memory Usage	Lower memory usage due to compressed tree structure	Higher memory usage because all TID sets must be stored
Efficiency	Faster for large & dense datasets	Faster for small or medium sparse datasets
Advantages	No candidate generation, compact structure, reduces database scans	Simple implementation, fast intersection operations, efficient vertical mining
Limitations	FP-tree construction becomes complex for huge datasets	Memory intensive when dataset becomes large
Suitable Applications	Market basket analysis, recommendation systems, customer behavior analysis	Clickstream analysis, web usage mining, telecom transaction analysis

FP-GROWTH MINING

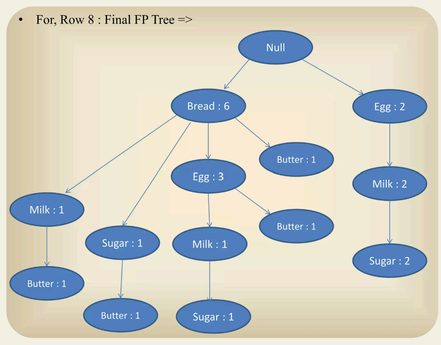
Mining Frequent Patterns from FP-Tree

Example: Find all frequent itemsets in the database using FP-growth algorithm. Take minimum support = 2

Transaction Id	Items
T1	Milk, Sugar, Bread, Egg
T2	Sugar, Bread, Butter
T3	Milk, Egg, Sugar
T4	Bread, Butter, Egg
T5	Bread, Butter, Milk
T6	Bread, Butter
T7	Milk, Sugar, Egg
T8	Bread, Egg

How Frequent Itemsets are Mined

1. Build the FP-Tree from the transaction database.
2. Start with the least frequent item.
3. Extract all paths containing that item.
4. Build a Conditional Pattern Base.
5. Create a Conditional FP-Tree.
6. Repeat recursively until all frequent itemsets are found.



Conditional Pattern Base & Conditional FP-Tree

Conditional Pattern Base

- Collection of prefix paths ending with a specific item.
- Used to identify related frequent patterns.

Conditional FP-Tree

- A smaller FP-Tree built from the Conditional Pattern Base.
- Used to efficiently mine additional frequent itemsets.

Benefits of FP-Tree Structure

Completeness	Compactness
Stores all frequent item information without losing patterns.	Compresses the database by sharing common prefixes.
Produces accurate frequent itemsets.	Uses less memory and speeds up mining.
Preserves transaction relationships.	No candidate item generation is needed.

ELCAT (EQUIVALENCE CLASS TRANSFORMATION) ALGORITHM

- An algorithm that mines frequent itemsets from a transaction database using vertical, depth-first approach instead of horizontal breadth-first search
- In association rule mining it efficiently identifies frequent itemset, which are used to generate association rules.

VERTICAL DATA FORMAT

- Each item is listed with the set of transaction IDs (TIDs) in which it appears.
- This set is called a TID-set or TID-list.
- This format makes it easy to compute support directly by counting the size of a TID-set, without rescanning the whole database.

Horizontal Format

Cake = {T1}
Coffee = {T1, T2}

Vertical Format

T1 = {Cake, Coffee}
T2 = {Coffee}

MINING FREQUENT ITEMSETS USING ECLAT

ECLAT finds frequent itemsets by intersecting TID-sets of items that **share a common prefix** rather than scanning the database repeatedly.

Start with the TID-sets of individual items (1-itemsets) → **Compute the support of each item as the size of its TID-set (keep only those meeting the minimum support threshold)** → **Combine pairs of frequent items sharing prefix by intersecting TID-sets** → **It produces a candidate 2-itemset and its TID-set.** → **Repeat this process to generate (k+1)-itemsets from frequent k-itemsets through intersection until no further frequent itemsets can form**

ECLAT avoids repeated database scans like Apriori because support is found by simply intersecting TID-sets which makes it faster for dense data and depth-first search

EXAMPLE

Items = Cake, Strudel, Cheesetart, Coffee | **Min support** = 2

TRANSACTIONS
T1: {Cake, Coffee}
T2: {Strudel, Coffee}
T3: {Cake, Strudel}
T4: {Cake, Coffee}

→

VERTICAL TID-SETS
Cake: {T1,T3,T4} (support 3)
Strudel: {T2,T3} (support 2)
Coffee: {T1,T2,T4} (support 3)

→

INTERSECT PAIRS
Cake ∩ Coffee = {T1,T4} → support 2 ✓ frequent
Cake ∩ Strudel = {T3} → support 1 ✗
Strudel ∩ Coffee = {T2} → support 1 ✗

→

RESULT
frequent itemsets = {Cake}, {Strudel}, {Coffee}, {Cake, Coffee}

REFERENCE

Bansal, A., Khare, A., & Moriwal, R. (2020). ECLAT algorithm for frequent item set generation with association rule mining algorithm. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. https://ijsrcseit.com/CSEIT206247

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. ACM SIGMOD
Record, 29(2), 1–12. https://doi.org/10.1145/335191.335372

CONCLUSION

FP-Growth and ECLAT are both efficient frequent itemset mining algorithms used in association rule mining. FP-Growth is generally more suitable for large and dense datasets because it compresses data into an FP-tree structure and avoids candidate generation. ECLAT performs better in smaller datasets by using vertical transaction ID intersections but requires more memory. The choice of algorithm depends on dataset size, structure, and application requirements.