## **FASE 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES**

### **Descarte de ideas no factibles**

En la siguiente tabla se presentan las ideas que fueron descartadas para el análisis de datos. Cabe mencionar se implementarán sin realizar una previa evaluación, estas son: El algoritmo de fuerza bruta y el algoritmo Apriori.

|  |  |
| --- | --- |
| *Idea no factible* | Justificación |
| *Idea 2-b*  *Partition* |  |
| *Idea 2-c*  *Eclat* |  |
|  |  |
| *Idea 8*  *Poda Alfa Beta* | Esta idea se descarta porque implica una técnica que está mucho más asociada a los juegos y por ende está más adaptada a ese tipo de problemas y no al que se tiene en cuestión. |

Para solucionar el problema de la presentación del análisis de los datos, se descartó la siguiente idea.

|  |  |
| --- | --- |
| *Idea no factible* | Justificación |
| *Idea 3*  *Graphics* | Se optó por descartar esta idea pues se considera muy tedioso realizar el dibujo de cada uno de los gráficos generados posteriormente del análisis de los datos de entrada. |

**Diseños preliminares**

Ahora bien, con las ideas que no fueron descartadas en el paso previo, se pretende realizar un diseño preliminar de cada una de estas, para así continuar con el proceso de análisis de las alternativas.

*Idea 1: Algoritmo de Fuerza Bruta:*

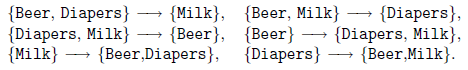
Es una técnica de búsqueda también llamada búsqueda exhaustiva que consiste en observar todos los posibles candidatos a una solución.

Considérese el siguiente ejemplo que consiste en un registro de compras.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T | Bread | Milk | Diapers | Beer | Eggs | Cola |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 1 | 1 | 1 | 0 | 0 | 1 |

En esta tabla, cada línea representa una compra. Por ejemplo, la segunda línea, ilustra una compra en la que se adquirió pan, pañales, cerveza y huevos, donde cada uno de estos es considerado un ítem dentro del problema..

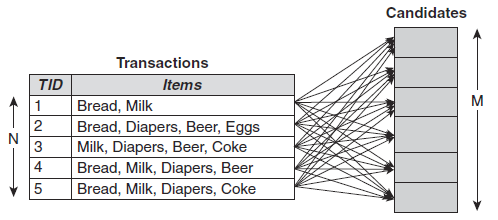
De la tabla anterior, se pueden extraer las siguientes hipótesis planteadas como dependencias entre ítems:



A partir de la generación de todos los itemsets, se puede concluir cuales son frecuentes y cuáles no.  
Por lo tanto, la idea general de este procedimiento es no solo generar los itemsets frecuentes, si no también generar una lista de reglas que sirvan para identificar rápidamente los itemsets. En general este proceso tiene un costo de recursos computacionales muy alto. Normalmente, un conjunto de datos que contenga *k* registros, puede llegar a generar *(2^k) -1* itemsets frecuentes.

Ahora bien, lo que se pretende con la aplicación de este algoritmo es tomar cada una de las combinaciones de ítems generadas y comprarlas con todas las transacciones que se encuentran registradas, esto con el objetivo de encontrar los candidatos más frecuentes.

Para exponer la idea anterior se opta por presentar la siguiente tabla, donde N es el número de transacciones, M la cantidad proyectada de itemsets [*(2^k) -1*] y w es el tamaño máximo de una transacción, en este caso w=4. Por ende, el costo computacional para una lista de transacciones más alta es mucho más caro, porque se deben realizar O(N M w) comparaciones.



*Idea 2: Algoritmo Apriori:*

Algoritmo utilizado en minería de datos, sobre bases de datos transaccionales, que permite encontrar de forma eficiente "conjuntos de ítems frecuentes", los cuales sirven de base para generar reglas de asociación. Procede identificando los ítems individuales frecuentes en la base y extenderlos a conjuntos de mayor tamaño siempre y cuando esos conjuntos de datos aparezcan suficientemente seguidos en dicha base de datos.

Considérese el mismo ejemplo expuesto en el algoritmo de fuerza bruta.

Ahora es necesario establecer cuándo un ítem es frecuente, en este caso un ítem se considerará frecuente si este aparece en por lo menos 3 transacciones. Posteriormente, se debe contar el número de ocurrencias de cada ítem en las transacciones, denominando este valor la confianza.

|  |  |
| --- | --- |
| ítem | Confianza |
| Bread | 4 |
| Milk | 4 |
| Diapers | 4 |
| Beer | 3 |
| Eggs | 1 |
| Cola | 2 |

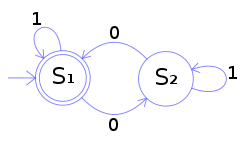
El próximo paso consiste en generar un listado de todos los pares de ítems frecuentes, en este caso no se tendrá en cuenta los siguientes ítems: Beer, Eggs, Cola.

*Idea 6: Autómatas finitos:*

Un autómata finito es un modelo computacional que realiza un proceso de forma automática para producir una salida desde cierta entrada. Para implementar un análisis de datos usando este método, la idea será crear un autómata finito que compute automáticamente un proceso que siempre lleve a analizar unos datos de entrada.

Para ahondar en el tema, un autómata finito es una 5-tupla que se compone de la siguiente manera:

(*Q*, Σ, *q0*, δ, *F*), donde:

* *Q* es el conjunto finito de estados por el que está compuesto el autómata.
* Σ es el alfabeto con el que actúa (dominio).
* *q0* pertenece a *Q.*
* δ es la función que define las transiciones entre estados.
* *F* es el conjunto que contiene a los estados del autómata en los que termina (estados de aceptación).

Por ejemplo, en la figura a mano izquierda se observa un autómata en el cual:

Q = {S1,S2}, Σ = {0,1}, *q0* = S1.

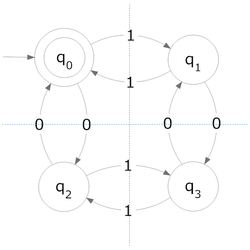
δ = S1 x 1 = S1, S1 x 0 = S2, S2 x 1 = S2, S2 x 0 = S1.

*F* = S2.

De la misma forma, un autómata finito puede representarse con una tabla de transiciones, siguiendo con el autómata anterior, su tabla de transiciones sería la siguiente:

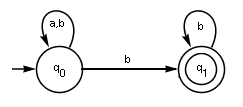
|  |  |  |
| --- | --- | --- |
| Salida qϵQ | Símbolo sϵΣ | Llegada q’ϵQ |
| S1 | 0 | S2 |
| S1 | 1 | S1 |
| S2 | 0 | S1 |
| S2 | 1 | S2 |

El ejemplo que se mostró anteriormente, se trata de un autómata finito determinista. No obstante, los autómatas finitos, varían en su tipo. Existen los AFD (autómatas finitos deterministas) y los AFN (autómatas finitos no deterministas). A continuación se va a profundizar más sobre la diferencia entre estos dos tipos de autómatas y cual se escogería para una futura implementación:

* Autómata finito determinista:   
  

Es un autómata finito que además es un sistema determinista; es decir, para cada estado *q* ∈ *Q* en que se encuentre el autómata, y con cualquier símbolo *s* ∈ Σ del alfabeto (dominio) leído, existe siempre una transición posible de un estado a otro. Por ejemplo, el autómata a mano izquierda es un AFD.

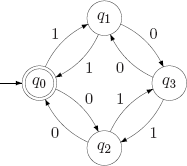
Como se ve en el diagrama, para todos los estados del autómata (q0, q1, q2, q3) existe máximo una transición por cada símbolo que reciba. Es decir, todos los estados van hacia algún lado cuándo reciben el símbolo 0 y el símbolo 1 y existe solo una transición por cada símbolo.

* Autómata finito no determinista:  
  

Es un autómata finito que, a diferencia de los autómatas finitos deterministas (AFD), posee al menos un estado *q* ∈ *Q*, tal que para un símbolo *s* ∈ Σ del alfabeto, exista más de una transición posible. Por ejemplo, en el diagrama a la izquierda se observa que para el estado q0, existen dos transiciones posibles generados por el símbolo b. Y además el estado q1, no posee ninguna transición para el símbolo a.

Después de observar todo lo pasado, se concluye que lo más apropiado para implementar un autómata que automatice un procedimiento que analice los datos que se tienen, sería un Autómata finito determinista. En efecto, este permite tener todos los posibles resultados para cualquier tipo de dato que entre (símbolo), a diferencia del AFN.

A continuación se mostrará cómo se implementaría esta idea con respecto al AFD de la figura 1.



*Figura 1.*

La idea sería la siguiente:

* Se tiene un conjunto (C) de datos. El estado inicial (podría ser q1 en el ejemplo) del autómata representaría el subconjunto (S0) de datos con el que comienzo a analizar.
* En seguida entra otro subconjunto de datos (S1) tal que S1 pertenece a C, es decir los subconjuntos son los datos que voy a ir analizando. El AFD, que por el momento se encuentra en el estado inicial, decide que procedimiento realizar dependiendo de la clasificación que tenga S2. (esta clasificación estaría representada por los símbolos posibles que tenga el autómata). En el diagrama a continuación estos serían los símbolos: 0 y 1.
* Se continuaría de esta forma hasta que se llegue a un estado final. En el diagrama sería: q0

Por ejemplo. En el caso hipotético de que se tenga el primer subconjunto S0 (q1) y entre otro subconjunto de datos S1 de tipo 1, entonces el autómata procedería a realizar las operaciones de análisis que lleven directamente al estado final q0. En el cual la información que se tiene se puede considerar un análisis relevante.