# Reglas de Asociación

Minería de datos

Master Universitario en Tecnologías de Análisis de Datos Masivos
Escola Técnica Superior de Enxeñaría (ETSE)
Universidade de Santiago de Compostela

# Contenidos de la presentación

- Introducción
  - Transacciones e itemsets
  - Regla de asociación
- Descubrimiento de reglas de asociación
  - Algoritmo Apriori
  - Algoritmo FP-Growth
- Generación de reglas de asociación
- Evaluación de reglas de asociación

# Introducción

El concepto de regla de asociación fue introducido [Agrawal et al., 1993] donde se aplicó al análisis de la cesta de la compra.

**Objetivo:**
Expresar **patrones de comportamiento** entre los datos en función de la aparición conjunta de valores de dos o más atributos.

A diferencia de los métodos de correlación, permite establecer **relaciones entre variables cualitativas**.

Aplicaciones:

- Análisis de la cesta de la compra.
- Estudio de textos.
- Búsqueda de patrones en páginas web.
- Diagnóstico médico y bioinformática.

# Transacciones e *itemsets*

Una **base de datos transaccional** hace referencia a una colección de transacciones, donde cada transacción queda representada por el conjunto de *items* (artículos) incluidos en la misma.

**Definición:**

Sea $I = \{i_1, i_2, \dots, i_d\}$ el conjunto de todos los *items* que pueden aparecer en una transacción.

- Un ***itemset*** es un conjunto de *items*.
- Si un *itemset* tiene k elementos se denomina ***k-itemset***.

**Definición:**

Sea $\mathrm{T} = \{t_1, t_2, \dots, t_N\}$ el conjunto de todas las transacciones.

- Una transacción está definida por el *itemset* que indica los *items* involucrados en la misma.

# Transacciones e *itemsets*

Existen diferentes formas de representar una base de datos transaccional:

- Horizontal

| TID | items |
|---|---|
| 1 | {Pan,Leche} |
| 2 | {Pan,Leche,Detergente,Cerveza,Huevos} |
| 3 | {Leche,Detergente,Cerveza,Cola} |
| 4 | {Pan,Leche,Detergente,Cerveza} |
| 5 | {Pan,Leche,Detergente,Cola} |

- Vertical

| Pan | Leche | Detergente | Cerveza | Huevos | Cola |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 3 |
| 2 | 2 | 3 | 3 | | 5 |
| 4 | 3 | 4 | 4 | | |
| 5 | 4 | 5 | | | |
| | 5 | | | | |

- Binaria

| TID | Pan | Leche | Detergente | Cerveza | Huevos | Cola |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 1 | 0 | 0 | 1 |

# Itemset

Sea $X$ un *itemset*. El número de transacciones en las que aparece $X$ se define como:

$$\sigma(X) = \{X \subseteq t_i, t_i \in T\}$$

**Definición (soporte de un itemset):**
Sea $X$ un *itemset*. El **soporte** de $X$ es la fracción de transacciones que lo incluyen:

$$supp(X) = \frac{\sigma(X)}{N}$$

**Definición (itemset frecuente):**
Un ***itemset* frecuente**, es un *itemset* cuyo soporte es superior a un mínimo establecido, ***MinSup***.

- Ejemplo:

| TID | items |
|---|---|
| 1 | {Pan, Leche} |
| 2 | {Pan, Leche, Detergente, Cerveza, Huevos} |
| 3 | {Leche, Detergente, Cerveza, Cola} |
| 4 | {Pan, Leche, Detergente, Cerveza} |
| 5 | {Pan, Leche, Detergente, Cola} |

| Itemsets | Soporte |
|---|---|
| {Leche, Detergente} | 4/5 |
| {Leche, Detergente, Cerveza} | 3/5 |
| {Cerveza, Detergente} | 3/5 |
| {Leche, Cerveza} | 3/5 |
| {Leche, Cola} | 2/5 |

# Regla de asociación

**Definición (regla de asociación):**
Una regla de asociación es una implicación de la forma $X \rightarrow Y$ donde:

- $X$ e $Y$ son ***itemsets*** disjuntos ($X \cap Y = \emptyset$).

- Ejemplo: $\{Detergente\} \rightarrow \{Cerveza\}$

Indica que existe una fuerte relación entre detergente y cerveza.

Una regla de asociación no implica causalidad, sino **coocurrencia**

**Definición (soporte de una regla de asociación):**
Sea la regla de asociación $X \rightarrow Y$. El soporte de una regla de asociación es la fracción de transacciones en las que están incluidos los *itemsets* tanto del antecedente como del consecuente ($X \cup Y$).

$$supp(X \rightarrow Y) = supp(X \cup Y)$$

El soporte de una regla nos indica cuántas veces es aplicable para el conjunto de transacciones.

# Regla de asociación

**Definición (confianza de una regla de asociación):**
Sea la regla de asociación $X \rightarrow Y$. La confianza de una regla de asociación es la fracción de transacciones en las que aparece el *itemset* $X$ y que también incluyen al *itemset* $Y$.

$$conf(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{supp(X \cup Y)}{supp(X)}$$

La confianza nos da una medida de lo frecuente que es encontrar a $Y$ en una transacción que incluye a $X$.

- Ejemplo: Dada la siguiente tabla de transacciones, calcular el soporte y confianza de:

$$\{Leche, Detergente\} \rightarrow \{Cerveza\}$$

$$supp(\{Leche, Detergente\} \rightarrow \{Cerveza\}) = supp(X \cup Y) = supp(\{Leche, Detergente, Cerveza\} = 3/5$$
$$supp(\{Leche, Detergente\} = 4/5$$

$$conf(\{Leche, Detergente\} \rightarrow \{Cerveza\}) = \frac{3/5}{4/5} = 3/4$$

# Descubrimiento de reglas de asociación

**Objetivo:**
Dados un conjunto de transacciones $T$ encontrar todas las reglas de asociación que tengan:

- Un soporte $\geq$ minsup, y
- Una confianza $\geq$ minconf.

Por lo tanto, para encontrar reglas de asociación debemos definir un soporte y una confianza mínimos.

**Aproximación naïve:**

- Calcular el soporte y confianza de todas las posibles reglas

Impracticable, ya que el número de posibles reglas para d *items* es: $R = 3^d - 2^{d+1} + 1$

**Aproximación más eficiente:**
Dado que la confianza es directamente proporcional al soporte, una aproximación puede ser solo **generar reglas para *itemsets* frecuentes**. Algoritmo en 2 pasos:

- Generación de *itemsets* frecuentes.
- Generación de reglas.

Si tenemos un conjunto de datos con $k$ *items* se podrían generar $2^k - 1$ *itemsets* frecuentes

La aproximación por fuerza bruta presenta un coste computacional de $O(NMw)$, donde $N$ es el número de transacciones, $M = 2^k - 1$ y $w$ es la longitud máxima de todas las transacciones.

# Descubrimiento de reglas de asociación

Existen dos estrategias básicas para reducir la carga computacional:

1) **Reducir el número de *itemsets* candidatos**.
   - Esta estrategia nos permitirá eliminar en número de candidatos posibles sin necesidad de calcular su soporte (principio Apriori).

2) **Reducir el número de comparaciones**.
   - Esta estrategia se basa en la utilización de estructuras de datos avanzadas que permiten reducir el número de comparaciones necesarias para la determinación de los *itemsets* frecuentes.

Veremos 2 aproximaciones:

- Apriori
- FP-Growth

# Apriori

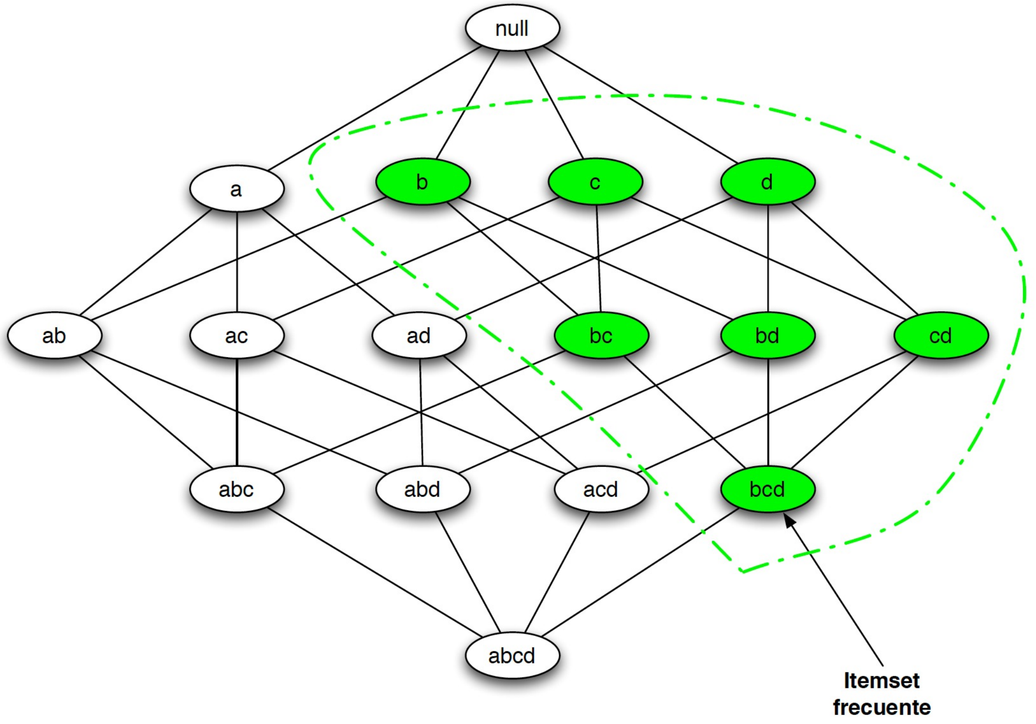
**Principio Apriori:**

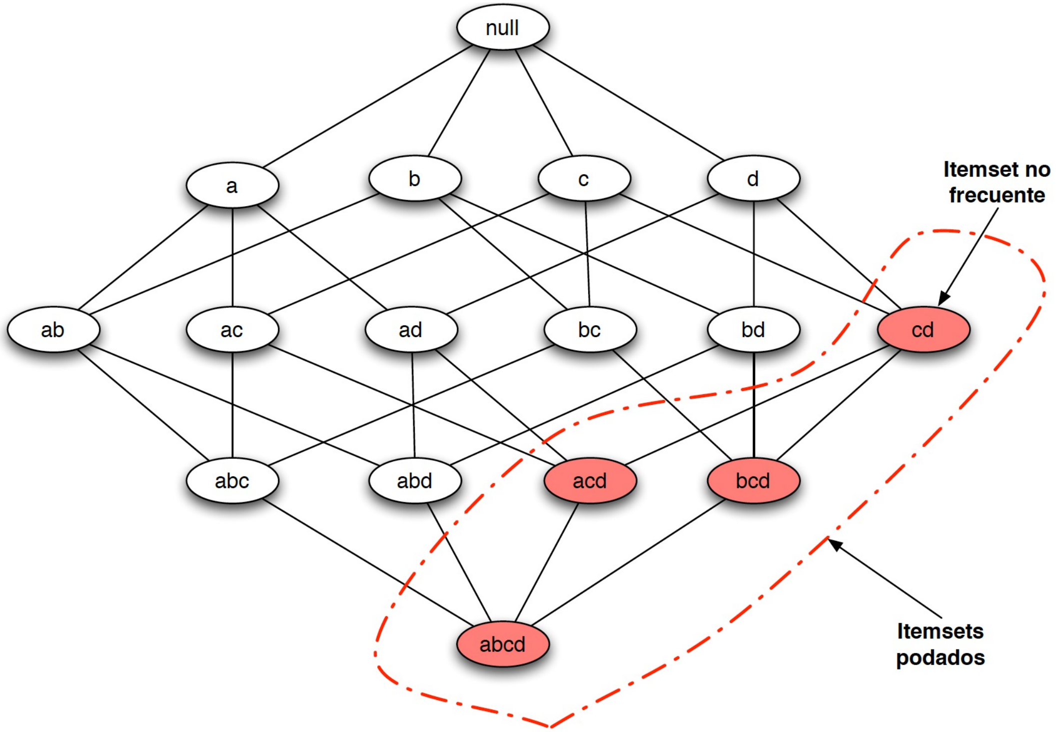
Todos los subconjuntos no vacíos de un *itemset* frecuente son también frecuentes:

- Si se añaden *items* a un *itemset* $I$ que es frecuente, éste seguirá siendo frecuente (al menos aparecerá en todas las transacciones en las que aparece $I$).
- De la misma forma, si un *itemset* $I$ no es frecuente, seguirá siendo no frecuente independientemente de cualquier adición de *items* al mismo (no puede aparecer en más transacciones en las que aparece $I$).
    - Esta propiedad se conoce como **antimonotonicidad** y puede ser utilizada para realizar una poda del espacio de búsqueda basada en el soporte.

# Descubrimiento de reglas de asociación

- Ejemplos (principio Apriori): *itemset* frecuentes y poda para el retículo con los items $\{a, b, c\}$



# Generación de itemsets

**Algoritmo** Apriori: Generación de itemsets frecuente

1: $k = 1$;
2: $F_k = \{i | i \in I \wedge \sigma(\{i\}) \geq N \times minSup\}$ ▷ Encontrar itemsets frecuentes
3: **repetir**
4: $k = k + 1$;
5: $C_k = Apriori\_Gen(F_{k-1})$; ▷ Generar itemsets candidatos
6: **para cada** transacción $t \in T$ **hacer**
7: $C_t = subset(C_k, t)$ ▷ itemsets candidatos en $t$
8: **para cada** itemset $c \in C_t$ **hacer**
9: $\sigma(c) = \sigma(c) + 1$ ▷ Incrementar el contador de soporte
10: **fin para**
11: **fin para**
12: $F_k = \{c | c \in C_k \wedge \sigma(c) \geq N \times minSup\}$
13: **hasta que** $F_k = \emptyset$
14: **Devolver** $\bigcup F_k$

- Sea $C_k$ el conjunto de $k$-*itemsets* candidatos y $F_k$ el conjunto de $k$-*itemsets* frecuentes.
- El algoritmo realiza un primer barrido para determinar los 1-itemsets frecuentes, F1 (L2).
- Seguidamente, y de forma iterativa, se van generando $k$-*itemsets* candidatos a partir de los $(k-1)$ *itemsets* frecuentes (L5).
- En el siguiente paso (L8-L10) se vuelven a recorrer los datos para calcular el soporte de cada $k$-*itemsets* candidato.
- Del conjunto $C_k$ se seleccionan aquellos $k$-*itemsets* que son frecuentes, conjunto $F_k$ (L12).
- El proceso termina cuando yo no se generan más $k$-*itemsets* frecuentes (L13) y se devuelven todos los *itemsets* frecuentes de tamaños 1 hasta $k$.

El algoritmo utiliza una estratégica de generación y prueba:

- En cada iteración se generan un conjunto de *itemsets* candidatos a partir de los *itemsets* frecuentes de iteraciones anteriores.
- Después se calcula el soporte para todos los *itemsets* candidatos para buscar los frecuentes.

# Generación de candidatos y poda

**Objetivo:**

La generación de candidatos debe cumplir los siguientes requisitos:

- Se debe evitar la generación de candidatos no necesarios (antimonotonicidad).
- Se debe garantizar que el conjunto de candidatos sea completo: $\forall k: F_k \subseteq C_k$ .
- Se debe de evitar la repetición de candidatos.

Estrategias para la generación de candidatos: Fuerza bruta, método $F_{k-1} \times F_1$, método $F_{k-1} \times F_{k-1}$.

**Fuerza bruta**

- En este caso, se consideran que todos los $k$-*itemsets* posibles como candidatos.
  - El número de posibles candidatos es $\binom{d}{k}$, siendo $d$ el número de *items*.
- El proceso de poda eliminará los candidatos no frecuentes.

- El cálculo de los $k$-*itemsets* candidatos es trivial, pero el proceso de poda es muy costoso
- El cálculo del soporte para cada candidato es $O(k)$.
- La complejidad total sería:

$$O(\sum_{k=1}^{d} \times \binom{d}{k}) = O(d \cdot 2^{d-1})$$

# Generación de candidatos y poda

$F_{k-1} \times F_1$

- La idea básica es la de extender cada (k-1)-*itemset* frecuente con un 1-itemset frecuente.
  - El número de posibles candidatos es $O(|F_{k-1}||F_1|)$

- La complejidad total sería:

$$O\left(\sum_{k=1}^{d} k \times |F_{k-1}||F_1|\right)$$

Inconvenientes:

- Se pueden generar *itemsets* duplicados. **Se hace necesario ordenar alfabéticamente los *items* en los *itemsets*.**
- Puede producir una gran cantidad de ***itemsets* innecesarios.**

$F_{k-1} \times F_{k-1}$

- Se fusionan dos (k-1)-*itemsets* frecuentes cuyos (k-2) items son idénticos, bajo la asumpción de que están ordenados alfabéticamente.
  - El número de candidatos es $O(|F_{k-1}||F_{k-1}|)$

- El orden alfabético asegura no generar itemsets duplicados.
- Se generan menos candidatos, pero sigue siendo necesaria la poda.
- La complejidad total sería:

$$O\left(\sum_{k=1}^{d} k \times |F_{k-1}||F_{k-1}|\right)$$

# Generación de candidatos y poda

- Ejemplos (Generación de candidatos $F_{k-1} \times F_1$):

$C_1$

| 1-itemsets. | Sop. |
|---|---|
| *Cerveza* | 3 |
| | |
| *Cola* | 2 |
| *Detergente* | 4 |
| | |
| *Huevos* | 1 |
| *Leche* | 5 |
| *Pan* | 4 |

$F_1$

| 1-itemsets. | Sop. |
|---|---|
| *Cerveza* | 3 |
| *Detergente* | 4 |
| *Leche* | 5 |
| *Pan* | 4 |

$C_2 \leftarrow (F_1, F_1)$

| 2-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente\}$ | 3 |
| $\{Cerveza, Leche\}$ | 3 |
| | |
| $\{Cerveza, Pan\}$ | 2 |
| $\{Detergente, Leche\}$ | 4 |
| $\{Detergente, Pan\}$ | 3 |
| $\{Leche, Pan\}$ | 4 |

$F_2$

| 2-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente\}$ | 3 |
| $\{Cerveza, Leche\}$ | 3 |
| $\{Detergente, Leche\}$ | 4 |
| $\{Detergente, Pan\}$ | 3 |
| $\{Leche, Pan\}$ | 4 |

$C_3 \leftarrow (F_2, F_1)$

| 3-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente, Leche\}$ | 3 |
| | |
| $\{Cerveza, Detergente, Pan\}$ | 2 |
| | |
| $\{Cerveza, Leche, Pan\}$ | 2 |
| $\{Detergente, Leche, Pan\}$ | 3 |

$F_3$

| 3-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente, Leche\}$ | 3 |
| $\{Detergente, Leche, Pan\}$ | 3 |

**Itemsets frecuentes:**
$F_1 \cup F_2 \cup F_3$

# Generación de candidatos y poda

- Ejemplos (Generación de candidatos $F_{k-1} \times F_{k-1}$):

$C_1$

| 1-itemsets. | Sop. |
|---|---|
| *Cerveza* | 3 |
| | |
| *Cola* | 2 |
| *Detergente* | 4 |
| | |
| *Huevos* | 1 |
| *Leche* | 5 |
| *Pan* | 4 |

$F_1$

| 1-itemsets. | Sop. |
|---|---|
| *Cerveza* | 3 |
| *Detergente* | 4 |
| *Leche* | 5 |
| *Pan* | 4 |

$C_2 \leftarrow (F_1, F_1)$

| 2-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente\}$ | 3 |
| $\{Cerveza, Leche\}$ | 3 |
| | |
| $\{Cerveza, Pan\}$ | 2 |
| $\{Detergente, Leche\}$ | 4 |
| $\{Detergente, Pan\}$ | 3 |
| $\{Leche, Pan\}$ | 4 |

$F_2$

| 2-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente\}$ | 3 |
| $\{Cerveza, Leche\}$ | 3 |
| $\{Detergente, Leche\}$ | 4 |
| $\{Detergente, Pan\}$ | 3 |
| $\{Leche, Pan\}$ | 4 |

$C_3 \leftarrow (F_2, F_2)$

| 3-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente, Leche\}$ | 3 |
| $\{Detergente, Leche, Pan\}$ | 3 |

$F_3$

| 3-itemsets. | Sop. |
|---|---|
| $\{Cerveza, Detergente, Leche\}$ | 3 |
| $\{Detergente, Leche, Pan\}$ | 3 |

**Itemsets frecuentes:**
$F_1 \cup F_2 \cup F_3$

# Cálculo del soporte

Es necesario comparar cada transacción con cada *itemset* candidato.

- Computacionalmente costoso

Se necesita optimizar la estructura de búsqueda (tablas hash, ...)

Una forma de optimizarlo:

- Calcular todos los *itemsets* de cada transacción y buscar en esta lista a los *itemsets* candidatos.

Otra forma:

- Eliminar del conjunto de *k-itemsets* candidatos a aquellos en los que algunos de esos subconjuntos no estén en $F_{k-1}$.

La más utilizada:

- No calcular el soporte en todas las fases de generación de candidatos.
  - **Fase Forward**. Se van generando los distintos conjuntos de *k-itemsets* candidatos. Sólo se detectan los frecuentes para determinadas longitudes.
  - **Fase Backward**. Se calcula el soporte para el resto *itemsets* no considerados, eliminando previamente aquellos itemsets que son subconjuntos de algún *itemset* frecuente.
    - Este último proceso encuentra los *itemsets* maximales.

# Descubrimiento de reglas de asociación

**Definición (*itemset* maximal frecuente):**
Un *itemset* maximal frecuente es un *itemset* frecuente para el que ninguno de sus superconjuntos inmediatos es frecuente.

Constituyen el conjunto más pequeño de *itemsets* frecuentes a partir de los que el resto de *itemsets* frecuentes se pueden derivar:

- Proporcionan una representación compacta en el caso de tener muchos itemsets frecuentes (existen algoritmos eficientes para detectarlos).

**Definición (*itemset* cerrado):**
Un itemset es cerrado si ninguno de sus superconjuntos tiene exactamente su mismo soporte.

Dicho de otra forma, un *itemset* no es cerrado si al menos uno de sus superconjuntos inmediatos tiene su mismo soporte.

- Todos los *itemsets* maximales frecuentes son también cerrados.
- Existen métodos eficientes para calcular, a partir de los *itemsets* cerrados frecuentes, el soporte de los *itemsets* no cerrados

# Cálculo del soporte

- **Ejercicio**: Generar los itemset frecuentes de la siguiente tabla de transacciones utilizando el algoritmo Apriori e identificar cuáles son cerrados y cuáles son maximales.

| TID | Items |
|---|---|
| T1 | {A,B,C,D} |
| T2 | {A,B,C,D} |
| T3 | {A,B,C} |
| T4 | {B,C,D} |
| T5 | {C,D} |

# FP-Growth

El método **frequent-pattern growth (FT-Growth)** intenta evitar los dos inconvenientes más importantes del algoritmo Apriori:

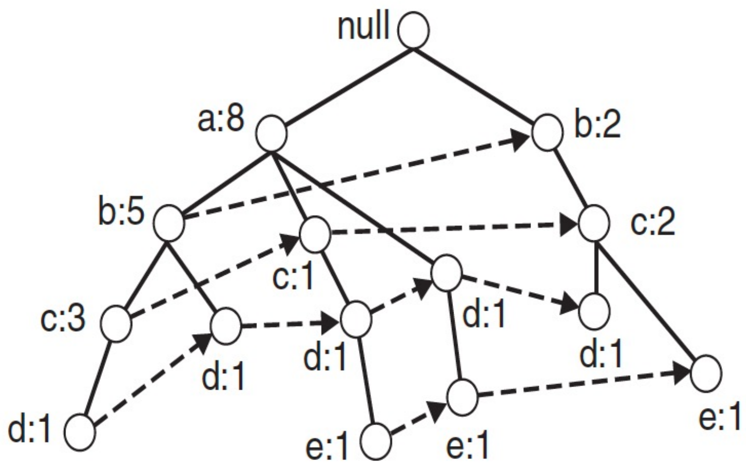
- Puede generar un número grande de *itemsets* candidatos.
- Puede necesitar realizar varias pasadas a la base de datos.

Se basa en una **estrategia divide y vencerás**:

- Se compacta la base de datos utilizando un FP-tree (**frequent pattern tree**).
- Se divide la BD compactada en BD condicionales y se extraen los *itemsets* frecuentes de ellas.

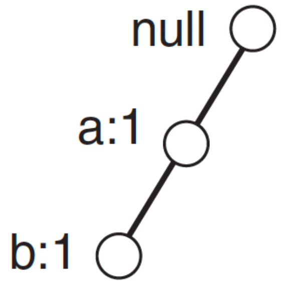
- Cada nodo del árbol-FP representa un ***item*** y tiene asociado un **contador** con el número de transacciones en las que el camino hasta el nodo está presente.
- El **nodo raíz** es el nodo nulo.
- También existen enlaces entre los nodos que representan ***items* idénticos**.

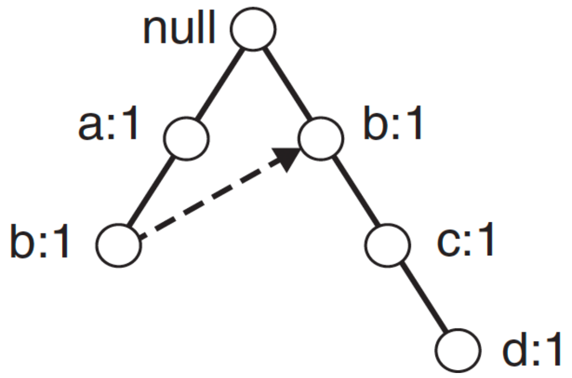
| TID | Items |
|---|---|
| 1 | {a,b} |
| 2 | {b,c,d} |
| 3 | {a,c,d,e} |
| 4 | {a,d,e} |
| 5 | {a,b,c} |
| 6 | {a,b,c,d} |
| 7 | {a} |
| 8 | {a,b,c} |
| 9 | {a,b,d} |
| 10 | {b,c,e} |



# Constucción del árbol

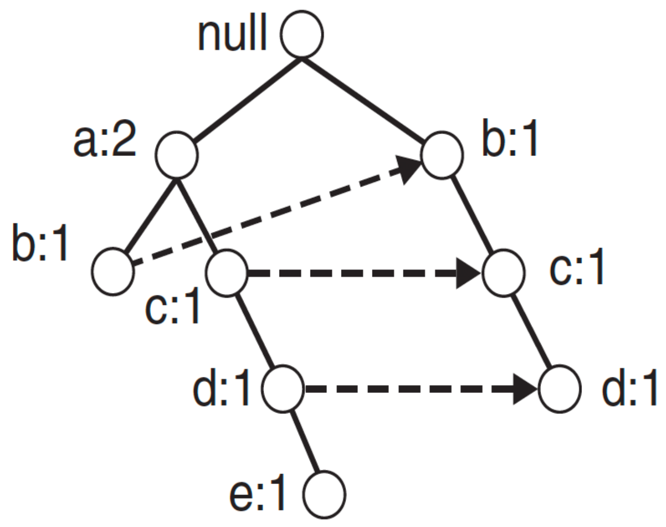
1. Se determina el soporte de cada *item* y se eliminan los no frecuentes.
2. Se ordenan los *items* frecuentes en orden descendente según su soporte en cada transacción.
3. Ahora se realiza otro barrido de la base de datos para analizar cada transacción.
4. La primera transacción $\{a, b\}$ da lugar al camino $null \rightarrow a \rightarrow b$. A cada uno de los contadores se les asigna el valor 1.
5. La segunda transacción $\{b, c, d\}$ da lugar a una nueva rama $null \rightarrow b \rightarrow c \rightarrow d$. Los contadores también se inicializan a 1.
   - Aunque comparten el *item* $b$, se generan ramas diferentes al no compartir el prefijo.
   - También se crea un enlace entre los nodos que representan el *item* $b$.



# Construcción del árbol

6. La tercera transacción $\{a, c, d, e\}$ comparte el prefijo $a$ con la primera transacción. Por lo tanto, la rama $null \rightarrow a \rightarrow c \rightarrow d \rightarrow e$ compartirá el nodo etiquetado como $a$. Esto hace que el contador del nodo a se incremente en 1.
   - También se crearán enlaces entre los nodos c y los nodos d.
   - El proceso continuaría hasta que se hallan procesado todas las transacciones.



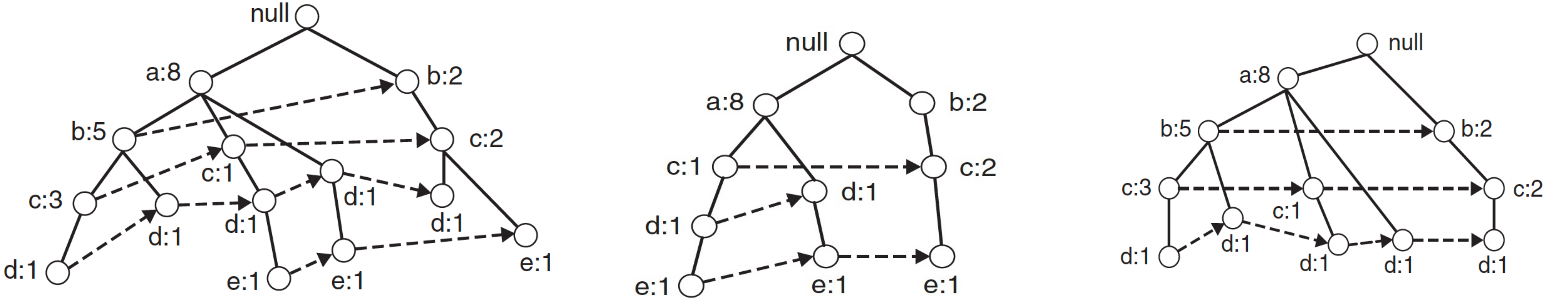

**Consideraciones:**

- El tamaño del árbol-FP es menor que la base de datos original.
- En el caso más favorable el árbol-FP contiene sólo una rama.
  - Todas las transacciones tienen los mismos *items*.
- En el caso más desfavorable, el árbol-FP tiene una rama por transacción.
- Depende del orden en el que se recorran los *items*.
  - Con un orden descendente del soporte se obtienen árboles más compactos.

# Generación de itemsets frecuentes

La generación de *itemsets* frecuentes **se realiza recorriendo el árbol-FP en sentido ascendente**:

- Se empieza buscando los *itemsets* frecuentes acabados en $e$, después los acabados en $de$, en $ce$, ... en $d$, en $cd$, ...
- Para localizar todas los *itemsets* acabados en $e$ sólo hace falta recorrer las ramas que acaban en $e$, proceso eficiente gracias a los enlaces entre todos los nodos $e$.




**Árbol-FP completo** **Subárbol-FP con ramas con el item e** **Subárbol-FP con ramas con el item d**

# Generación de itemsets frecuentes

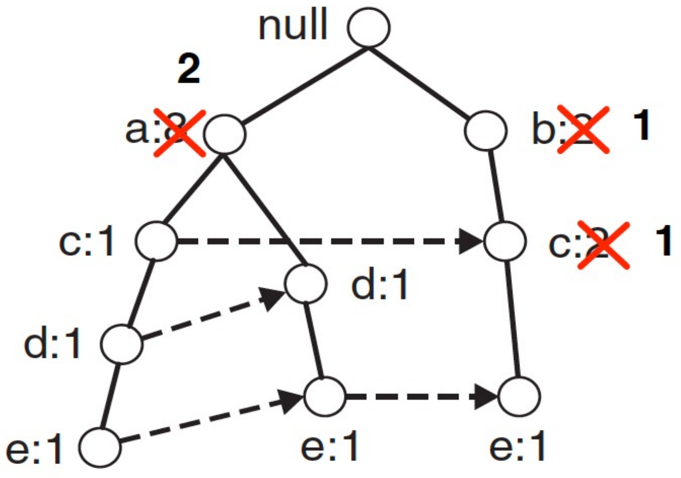
Supongamos que queremos calcular todos los *itemsets* frecuentes que acaban en $e$:

- Primero debemos determinar si el *item* $e$ es frecuente. Para ello sólo debemos sumar los contadores asociados a los nodos $e$.
- Si suponemos un soporte de 2, el *item* $e$ es frecuente al ser la suma de sus contadores 3.
- Por lo tanto, $\{e\}$ es un *itemset* frecuente.
- Seguidamente buscaríamos los *itemsets* frecuentes acabados en $e$, después los acabados en $de$, seguidos por los acabados en $ce$, $be$ y $ae$.

Para ello, se construye un **árbol-FP condicional** a partir del subárbol-FP con las ramas que acaban en $e$:

- Se actualizan los contadores ya que estos incluyen transacciones que no contienen $e$. La transacción $\{b, c\}$ y algunas que empiezan por $a$ no contienen al item $e$.

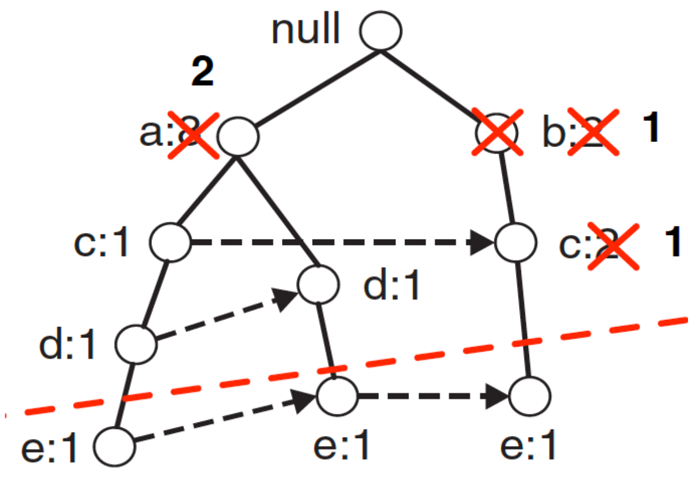
Es el árbol-FP que se hubiera construido si se eliminan todas las transacciones que no contienen $e$ y, del resto, eliminamos el item $e$



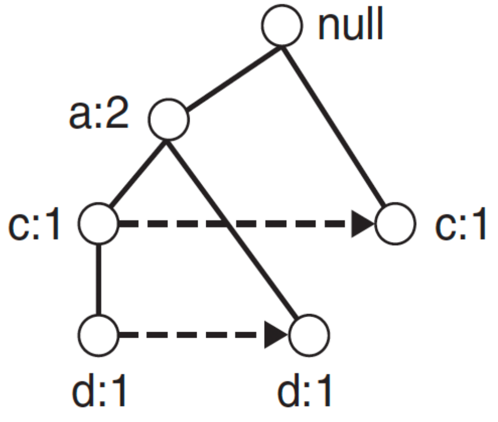

$$F_e = \{e\}$$

# Generación de itemsets frecuentes

- Se eliminan los nodos que contienen al item $e$ (ya no son necesarios).
- Se eliminan los nodos no frecuentes, en este caso $b$.




- FP-Growth usa este árbol-FP condicional para $e$ para entontrar los itemsets frecuentes acabados en $de$ y $ce$.

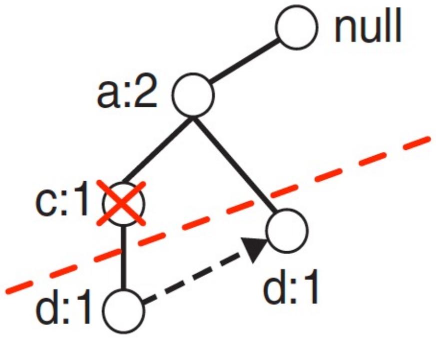



$$F_e = \{\{e\}\}$$

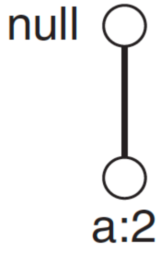
# Generación de itemsets frecuentes

- Para encontrar los itemsets frecuentes acabados en $de$ hay que generar el subárbol-FP acabado en $d$ a partir del árbol-FP condicional acabado en $e$.

- Sumando el soporte de los nodos $d$, que es igual a 2, determinamos que $\{d, e\}$ también es frecuente.



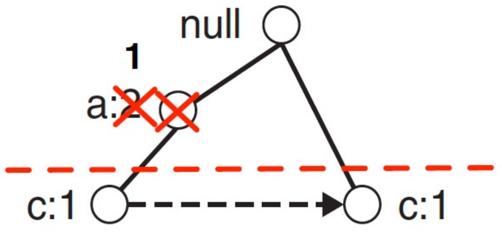

- Ahora se construye el subárbol-FP condicional acabado en $ade$. Como este árbol condicional sólo tiene el nodo $a$ con soporte mayor que minSup, el itemset $\{a, d, e\}$ también es frecuente.




$$F_e = \{\{e\}, \{d, e\}, \{a, d, e\}\}$$

# Generación de itemsets frecuentes

- A continuación se procedería a buscar los acabados en $ce$, creando el subárbol-FP acabado en $c$ a partir del árbol condicional de $e$.
- Se determina que el nodo $c$ es frecuente, siendo el itemset $\{c, e\}$ frecuente.
- A partir de este subárbol-FP se genera el árbol-FP condicional para la terminación $ce$.
- Al ser un árbol vacío se procedería encontrar los itemsets frecuentes acabados en $ae$, $\{a, e\}$.




$$F_e = \{\{e\}, \{d, e\}, \{a, d, e\}, \{c, e\}, \{a, e\}\}$$

Siguiendo con el proceso para los itemsets acabados en d,c,b y a, otendríamos los siguientes itemsets frecuentes:

| Sufijo | Itemsets frecuentes |
|---|---|
| e | $\{e\}, \{d, e\}, \{a, d, e\}, \{c, e\}, \{a, e\}$ |
| d | $\{d\}, \{c, d\}, \{b, c, d\}, \{a, c, d\}, \{b, d\}, \{a, b, d\}, \{a, d\}$ |
| c | $\{c\}, \{b, c\}, \{a, b, c\}, \{a, c\}$ |
| b | $\{b\}, \{a, b\}$ |
| a | $\{a\}$ |

# FP-Growth

**Ventajas:**

- Solo se requieren dos pasadas a la base de datos.
- Comprime los datos.
- No se generan *itemsets* frecuentes duplicados al ser los subproblemas disjuntos.
- Más rápido que Apriori ya que no requiere la generación de candidatos.

**Desventajas:**

- El árbol-FP puede que no quepa en memoria.
- La construcción de una árbol-FP es muy costosa. Sin embargo, una vez construido encontrar los *itemsets* frecuentes es muy fácil.
- Sólo se podan *items* individuales y no *itemsets*.
- El soporte sólo se puede calcular una vez construido el árbol-FP.

# Generación de reglas

Una vez obtenidos los *itemsets* frecuentes debemos generar las reglas de asociación.

- Cada *itemset* frecuente, $Y$ , puede generar $2^k - 2$ reglas de asociación (las reglas $Y \rightarrow \emptyset$ y $\emptyset \rightarrow Y$ no se tienen en cuenta).
- Para crear una regla de asociación a partir del *itemset* frecuente $Y$ es necesario:
  1. Dividir $Y$ en dos conjuntos disjuntos $X$ y $Y \backslash X$.
  2. Generar la regla $X \rightarrow Y \backslash X$ si supera la confianza mínima.

Ejemplo: Sea el *itemset* $\{a, b, c\}$. A partir de él se pueden generar las siguientes reglas de asociación:

$$\{a, b\} \rightarrow \{c\} \qquad \{a\} \rightarrow \{b, c\}$$
$$\{a, c\} \rightarrow \{b\} \qquad \{b\} \rightarrow \{a, c\}$$
$$\{b, c\} \rightarrow \{a\} \qquad \{c\} \rightarrow \{a, b\}$$

El soporte de cada regla supera el MinSup al ser generada a partir de un *itemset* frecuente

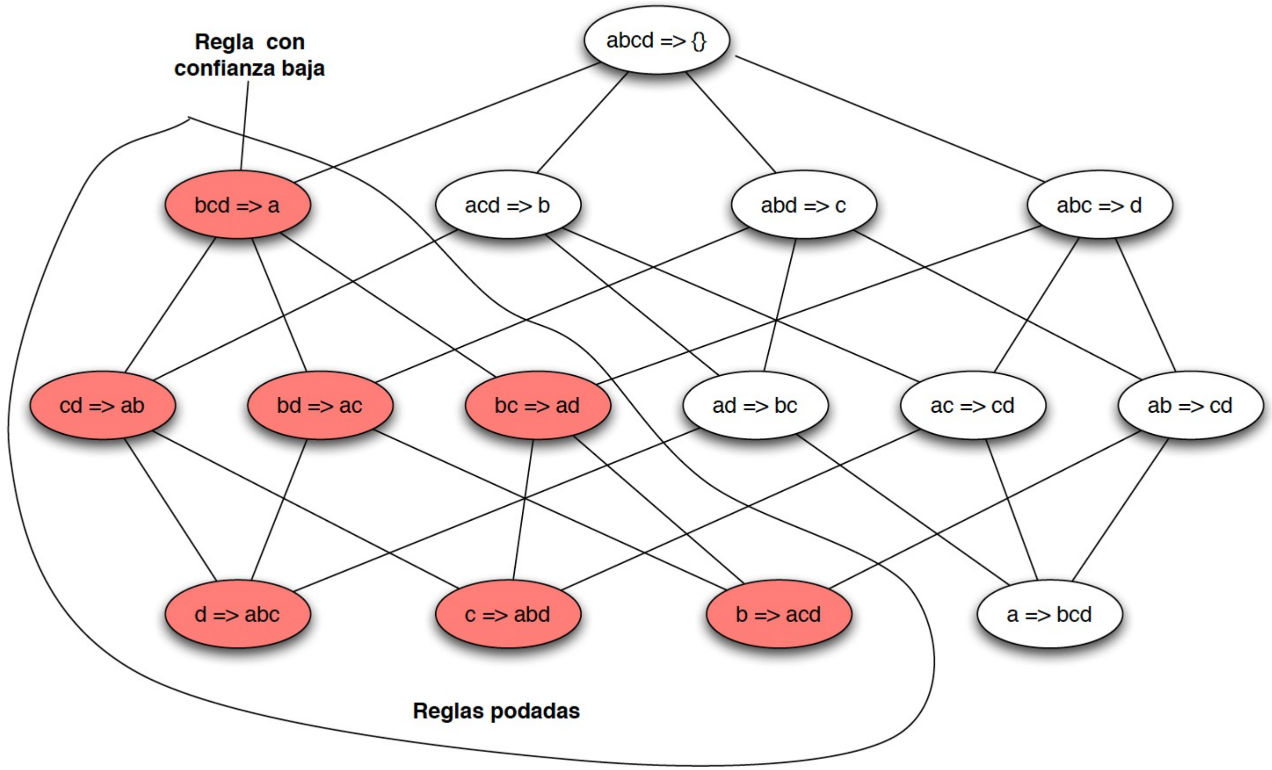
# Generación de reglas

El cálculo de la confianza no requiere volver a recorrer la base de datos dado que:

$$conf(X \to Y) = \frac{supp(X \cup Y)}{supp(X)}$$

**Propiedad:**

Sea la regla $X \to Y \backslash X$ cuya confianza es menor que el umbral, entonces cualquier regla $X' \to Y \backslash X'$, con $X' \subseteq X$, tendrá una confianza menor que el umbral.




Esta propiedad nos permite podar un subconjunto de reglas

# Generación de reglas

**Algoritmo** Apriori: Generación de reglas en Apriori

**para cada** $k-$itemset frecuente $f_k$, $k \geq 2$ **hacer**<br>
    $H_1 = \{i | i \in f_k\}$ ▷ consecuentes de un item<br>
    **ejecutar** ap-genrules($f_k, H_1$)<br>
**fin para**

**Algoritmo** ap-genrules($f_k, H_m$)

$k = |f_k|$; ▷ Tamaño del itemset frecuente<br>
$m = |H_m|$ ▷ Tamaño del consecuente<br>
**si** $k > m + 1$ **entonces**<br>
    $H_{m+1} =$apriori-gen($H_m$)<br>
    **para cada** $h_{m+1} \in H_{m+1}$ **hacer**<br>
        $conf = \sigma(f_k)/\sigma(f_k \setminus h_{m+1})$<br>
        **si** $conf \geq minconf$ **entonces**<br>
            **generar la regla** $(f_k \setminus h_{m+1}) \to h_{m+1}$<br>
        **sino**<br>
            $H_{m+1} = H_{m+1} \setminus h_{m+1}$<br>
        **fin si**<br>
    **fin para**<br>
    **ejecutar** ap-genrules($f_k, H_{m+1}$)<br>
**fin si**

El algoritmo Apriori sigue un proceso por niveles:

- Cada nivel se corresponde con el número de *items* en el consecuente.
- Se comienza con las reglas que tienen un *item* en el consecuente y nos quedamos con las que tienen una confianza alta.
- Estas reglas se utilizan para generar las reglas del siguiente nivel.

Ejemplo:

$$\left.\begin{array}{l} \{a,b,d\} \to \{c\} \\ \\ \{a,c,d\} \to \{b\} \end{array}\right\} \Longrightarrow \{a,d\} \to \{bc\}$$

# Evaluación de reglas de asociación

Los métodos que hemos analizado tienden a generar una gran cantidad de reglas de asociación.

- En aplicaciones reales podemos estar hablando de miles o millones de reglas.

Un soporte demasiado bajo complica el descubrimiento de reglas:

- Aumentan los requisitos de memoria y computación.
- Se obtienen muchas reglas.
- Se pueden obtener reglas espurias, que relacionen *items* muy frecuentes con otros muy poco frecuentes.

Reglas de una confianza alta se pueden perder debido a que la confianza no tiene en cuenta el soporte del consecuente.

Muchas de las medidas se calculan a través de la tabla de contingencia:

| | $q$ | $\bar{q}$ | |
|---|---|---|---|
| $p$ | $f_{11}$ | $f_{10}$ | $f_{1+}$ |
| $\bar{p}$ | $f_{01}$ | $f_{00}$ | $f_{0+}$ |
| | $f_{+1}$ | $f_{+0}$ | $N$ |

- $\bar{p}$ $(\bar{q})$ indica que el *item* $p$ $(q)$ no está en la transacción.
- $f_{11}$ indica el número de transacciones en las que $p$ y $q$ están presentes.
- $f_{1+}$ representa el soporte de $p$ y $f_{+1}$ el soporte de $q$.

# Evaluación de reglas de asociación

**Definición (Lift):**
El *lift* compara la frecuencia de la regla con la frecuencia base, asumida la independencia lineal entre los *items*. Para variables binarias se conoce como factor de interés y se puede calcular como sigue:

$$l(p \to \mathrm{q}) = l(p,q) = \frac{supp(p \cap q)}{supp(p) \cdot supp(q)} = \frac{N \cdot f_{11}}{f_{1+} \cdot f_{+1}}$$

Se puede interpretar de la siguiente manera:

$$l(p,q) = \begin{cases} < 1 & p \text{ y } q \text{ están correlacionadas negativamente} \\ = 1 & p \text{ y } q \text{ son independientes} \\ > 1 & p \text{ y } q \text{ están correlacionadas positivamente} \end{cases}$$

- Puede darse el caso de que asociaciones muy frecuentes tengan un factor de interés cercano a 1 debido a las distribuciones de las ocurrencias.
  - Sobre todo, si el peso de la matriz de contingencia se concentra en el término $f_{11}$

# Evaluación de reglas de asociación

**Definición (Análisis de correlación):**
Esta medida está basada en técnicas estadísticas para medir la relación entre dos variables. Para variables binarias (coeficiente *phi*) se usa la siguiente fórmula:

$$\phi(X \rightarrow Y) = \frac{f_{11}f_{00} - f_{10}f_{01}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

Se puede interpretar de la siguiente manera:

- Este valor varía entre -1 (correlación negativa perfecta) y +1 (correlación positiva perfecta).
- Las variables son estadísticamente independientes si su valor es 0.

El problema de esta medida es que da la misma importancia a la coocurrencia que a la coausencia.

- Por lo tanto, es muy útil cuando se están analizando variables binarias simétricas.