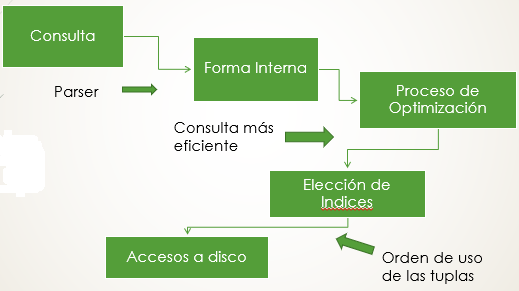
**Componentes que conforman el “costo” de la ejecución de una consulta**

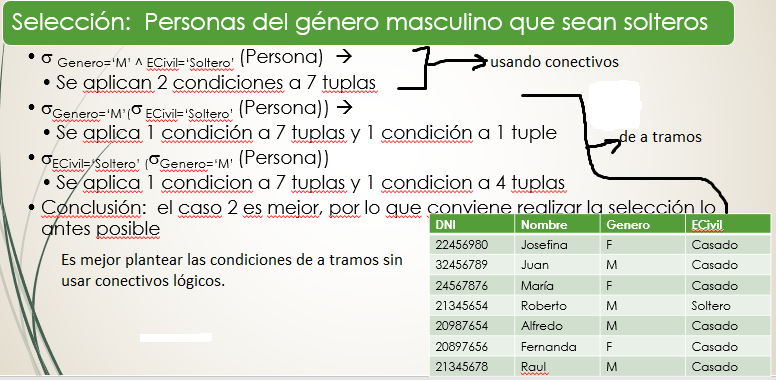
* El costo de acceso a almacenamiento secundario 🡪 acceder al bloque de datos correspondiente en el disco. Medido en milisegundos
* Costo de cómputo 🡪 el costo de realizar operaciones sobre RAM (filtrado de tuplas sobre memoria RAM). Medido en nanosegundos
* Costo de comunicación 🡪 el costo que implica enviar la consulta (resuelta sobre un servidor) y los resultados correspondientes (si es un sistema distribuido: una base de datos dispersa en varios servidores, se debe procesar la consulta en varios servidores).



* Proceso de recepción de consulta:
  + La consulta recibida se transforma en un formato interno usando el parser (reescritura de consulta que implica una consulta más fácil de leer para el DBMS).
  + Sobre esa consulta reescrita se le aplica el proceso de optimización. El DBMS optimiza las consultas. De forma que la consulta sea expresada de la forma más eficiente posible.
  + Una vez conseguida dicha forma eficiente, se eligen los índices para hacer el uso más optimo de las tuplas (esto implica reducir los accesos a discos).

**Optimización lógica**

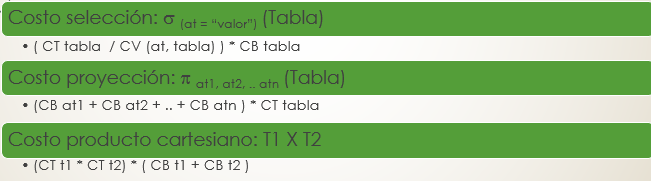
* Las consultas que hacemos en AR/SQL pueden recibir una optimización lógica, que consiste en que nuestra consulta puede realizarse de otra forma más diferente pero que implique una mayor optimización.
* Se puede hallar una expresión más eficiente que otra.

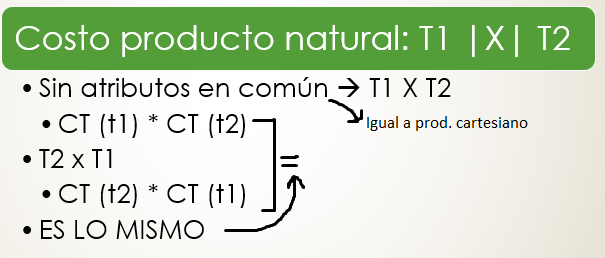


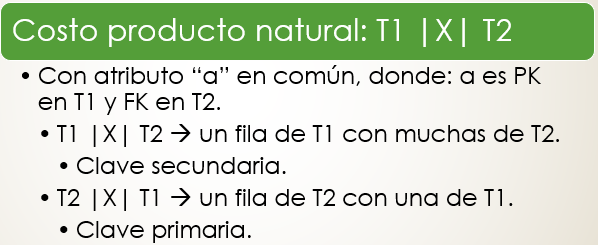
* Vemos que la consulta 2 y la 3 son semejantes xq se usa el planteo de a tramos (usando proposiciones atómicas), pero una es más eficiente que otra, entonces como sabemos cuál elegir cuando planteemos una consulta parecida en una DBMS.
  + No podremos saberlo, en cambio el optimizador de consultas de la DBMS sabe cuál es el estado de la misma, por lo que al optimizarse según un criterio estadístico elije la opción más eficiente.
  + ¿Cuál es la opción más eficiente? 🡪 aquella que tenga menos valores.
    - Como hay menos solteros que masculinos, es mejor la consulta 2.
    - Si fuese al revés, la consulta 3 sería mejor.
* Priorizamos legibilidad antes que eficiencia 🡪 el DBMS optimiza automáticamente.
* La proyección disminuye la cantidad de información que se almacena en buffers de memoria.
* Cuando una operación binaria implica reducción de cantidad, significa que provoca consultas más eficientes.

**Algunos valores:**

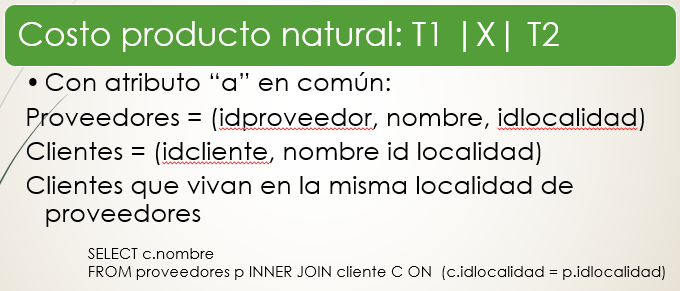
* CT tabla 🡪 cantidad de tuplas de una tabla.
* CB tabla 🡪 cantidad de bytes que ocupa cada tupla.
* CV (a, tabla) 🡪 cuantos valores diferentes hay para el atributo A en la tabla determinada.
  + Por ej, si idSocio en la tabla Socios (con un CT Socios = 1000).
  + Cual va a ser mi CV (idSocio, Socios) 🡪 1000 valores diferentes por que idSocio es clave primaria.



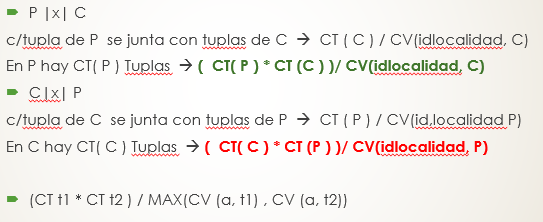




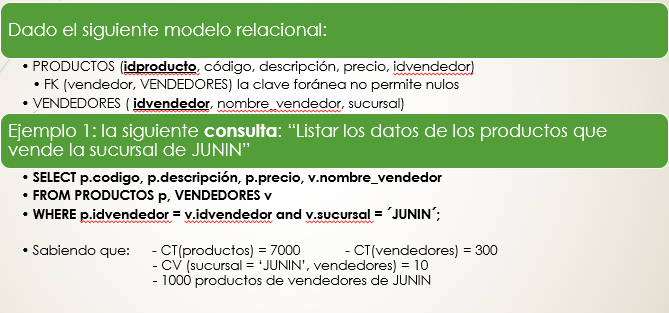
* ¿Qué es más eficiente? La clave primaria xq se usa Hashing para acceso
  + La generación de clave secundaria es accedido mediante arboles, más ineficiente.
* Por qué generan claves?
  + Un T1 con muchos T2 genera tuplas con claves secundaria.
  + Un T2 con muchos T1 genera tuplas con claves primaria (ya que alinea 1 con 1).



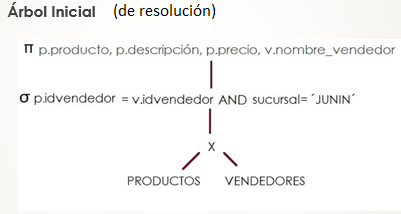
* Idlocalidad es clave foránea para tanto proveedores como para clientes.
* La consulta usa dos claves foráneas.
* ¿Qué es más eficiente? Proveedores inner join clientes o Clientes inner join proveedores?

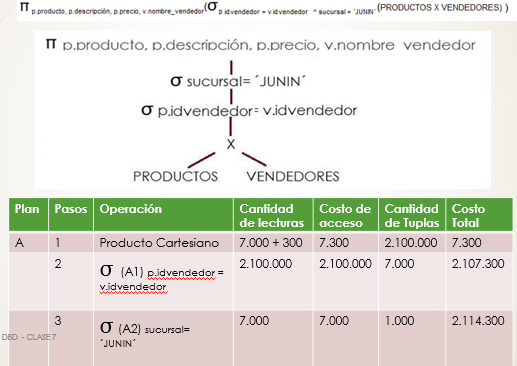


* En P |X| C y C |X| P solo cambia el denominador.
  + ¿Cuál es más eficiente? Aquella donde el denominador sea MÁS GRANDE 🡪 me dá valores más chicos.
  + La multiplicación es conmutativa, no nos interesa.

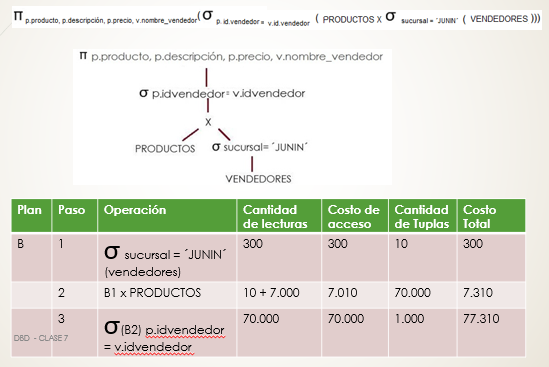


* La consulta en negro devuelve 1000 tuplas.

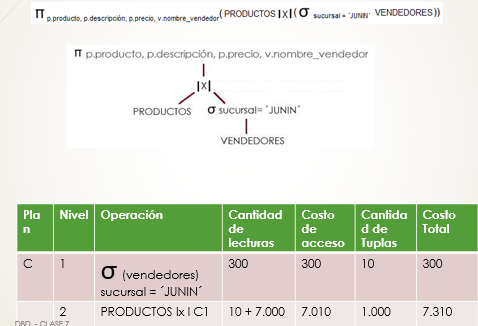




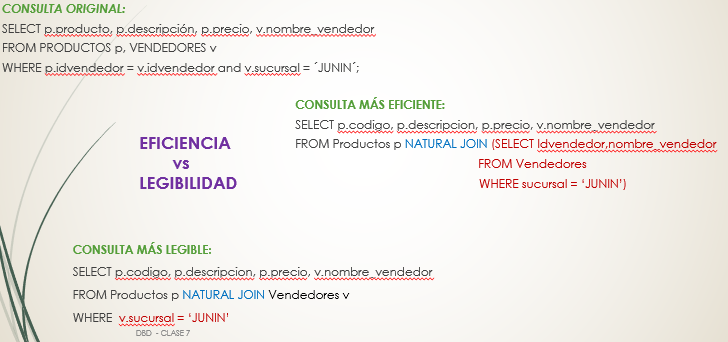
* El producto cartesiano en el primer paso genera 2.100.100 donde muchas son espurias.
* El siguiente paso es filtrar esas 2.100.100 tuplas, ósea 2.100.100 cantidad de lecturas con un mismo costo de acceso, eso me genera 7000 tuplas. El costo total final es 2.100.100 + 7.300
* El paso final es seleccionar aquellas tuplas de las 7000 donde la sucursal sea Junin, leo 7000 tuplas, accedo 7000 veces, genero 1000 tuplas. El costo total es 2.107.300 + 7000 accesos del paso final. Osea 2.114.300 lecturas para resolver la consulta.



* El primer paso es elegir los vendedores de junin, me cuesta 300 accesos para tomar todas las tuplas, el resultado son 10 tuplas. El costo total es 300.
* Al paso anterior que generó 10 tuplas se multiplica cartesianamente por la cantidad de productos que son 700, genera un costo de acceso de 7.010 y genera 70.000 tuplas. Con un costo total de 300 (paso anterior) + 7.010 = 7.310
* El paso anterior generó 70.000 tuplas, al aplicar el filtro genero 1.000 tuplas. El costo total y final es de 7.310 + 70.000 = 77.310 como costo total.
* Es mejor que el plan A. xq selecciona antes de aplicar productos cartesianos.



* Primera diferencia, usa producto natural.
  + Selecciona primero los vendedores de junin, genera 10 tuplas y lee 300 tuplas.
  + El siguiente paso lee los 7.000 productos y los 10 vendedores (7.010 lecturas), generando 1.000 tuplas en el resultado final.
  + El costo total es 7.310
  + Resulta ser el plan más eficiente, pasamos de 2 millones de operaciones de lectura a 7000 operaciones.
* Entendemos entonces que el proceso de optimización es vital.
  + Es un problema del programador dicho proceso?
    - No, se puede deslindar y dejar la responsabilidad a DBMS ya que éste SABE cómo optimizar mejor una consulta.



* No importa que consulta le demos al DBMS, siempre va a convertir a la consulta más eficiente.
* El proceso de optimización SIEMPRE queda a cargo del DBMS.
* Por lo que nos interesa hacer consultar más legibles.
  + Nuestra prioridad.