75.15 / 75.28 / 95.05 - Base de Datos

Data Warehousing

Mariano Beiró

Dpto. de Computación - Facultad de Ingeniería (UBA)

5 de julio de 2022

Topics

OLTP (On-line transaction processing)

Contexto. Capacidad transaccional.

- En el ámbito empresarial, las bases de datos almacenan:
  - Datos estáticos sobre las entidades involucradas en el negocio (ej.: clientes, productos, contactos, direcciones, ...)
  - Datos que se generan dinámicamente a través de la operatoria habitual (compras, pedidos, devoluciones, liquidaciones, ...).
- El volumen de datos dinámicos se relaciona con el tamaño de la organización, y debe ser previsto para dimensionar la base de datos: requisitos de hardware y red, qué SGBD utilizaremos, etc...
- Esta capacidad de un SGBD para procesar el volumen de datos que la actividad de la empresa genera es denomina capacidad transaccional.
- Con el surgimiento de la Web, los SGBD's comenzaron a procesar transacciones a través de Internet.

OLTP (On-line transaction processing)

Arquitectura de 3 capas. OLTP.

- Se desarrolló así una arquitectura de tres capas (three layer architecture), que en el caso de la Web está formada por:
  - Capa de presentación: Es la interfaz Web en que el usuario carga los parámetros de su consulta y ve el resultado.
  - Capa lógica: Es la capa intermedia que hace de servidor Web. Recibe la consulta y la ejecuta conectándose a nodos de almacenamiento.
  - Capa de datos: Constituída por los nodos de almacenamiento de datos y sus procesos.
- La capa lógica, si bien no almacena información, es de fundamental importancia para la escalabilidad del sistema y sus capacidades de concurrencia.
- Los SGBD's, que hasta entonces sólo representaban la capa de datos de la arquitectura, comenzaron a solaparse cada vez más con la capa lógica.

OLAP vs. OLTP

- A la capacidad de procesar transacciones en línea en forma concurrente entre una gran cantidad de usuarios, y de manera escalable, se la conoce como OLTP (On-line transaction processing).
- Pero la inteligencia de negocios requiere extraer información de los datos almacenados que sirva de soporte para la toma de decisiones.
- Para ello, hacen falta dos premisas distintas:
  1. Reducir la cantidad de datos.
  2. Poder expresar consultas más complejas.
- Como guía para incluir estas capacidades dentro de los SGBDs, E. Codd propuso el concepto de capacidad analítica.
- Codd utilizó la sigla OLAP (On-line analitical processing) para diferenciar a estas cualidades de las capacidades transaccionales clásicas de los SGBD's.

Reglas OLAP

Según Codd, una herramienta de procesamiento analítico (OLAP) debería brindar una serie de servicios, que denominó reglas OLAP[1]:

- Vista conceptual multidimensional. Mantener los datos en una matriz en la que cada dimensión representa un *atributo*.
- Manipulación intuitiva de los datos. Poder diseñar la vista conceptual a través de una interfaz amigable.
- Accesibilidad. Es habitual que las fuentes de datos con que trabajamos sean muy heterogéneas (distintos SGBD's, formatos). OLAP debería mediar entre las fuentes y la interfaz de usuario.
- Extracción batch e interpretativa. Poder almacenar en una base de datos propia el resultado del procesamiento en *batch*, y también actualizar ese resultado "en vivo" si el cliente lo requiere.
- Modelos de análisis. Poder responder consultas de tipo estadístico o predictivo que ayuden a la toma de decisiones.
- Arquitectura cliente-servidor.

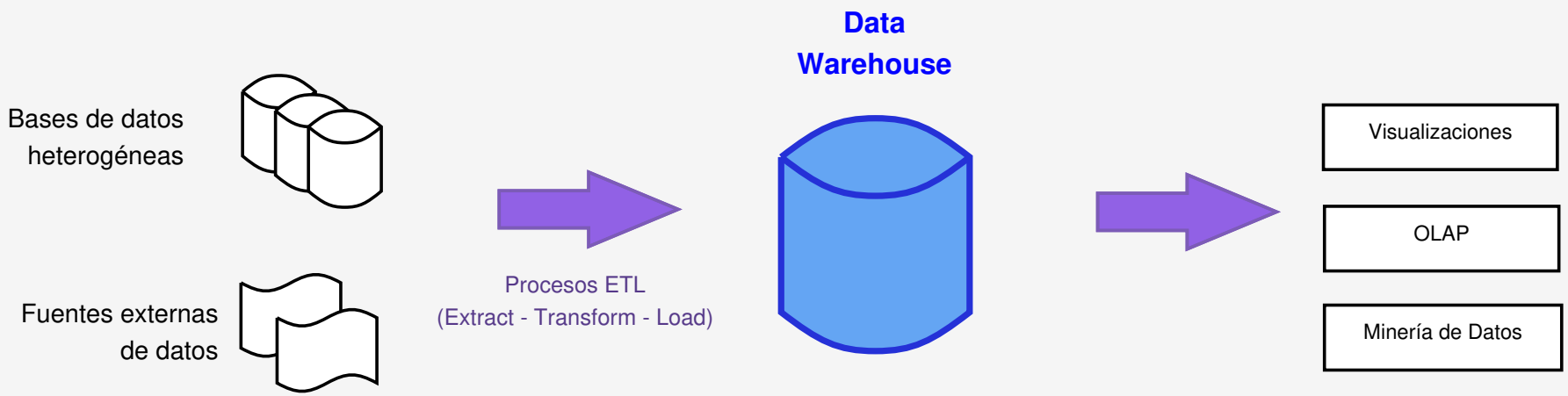
[1] Sólo incluímos aquí 6 de las 12 reglas.

Data Warehouse

Concepto

[ELM16 29.2]

- Las aplicaciones OLAP se ejecutan generalmente en un copia paralela de la/s base/s de datos principal/es de una organización, conocido como data warehouse.
- Así, los *data warehouses* integran datos provienientes de fuentes de datos heterogéneas.



Modelo dimensional

Modelado conceptual

[ELM16 29.3]

- Para construir una estructura OLAP, el SGBD toma una instantánea de la base de datos en un momento determinado, y *agrega* los datos para construir un modelo multidimensional de los mismos.
- Para definir este modelo en términos conceptuales se pueden utilizar[2]:
  1. Diagramas de estrella *(star)*
  2. Diagramas copo de nieve *(snowflake)*

[2] Aquí introduciremos únicamente los diagramas de estrella. En los mismos, las tablas de dimensiones pueden estar desnormalizadas. En los diagramas copo de nieve, en cambio, todas las tablas se encuentran normalizadas.

Modelado conceptual

Hechos y dimensiones

- En el modelo dimensional nuestro objetivo es definir una serie de medidas numéricas sobre un conjunto de atributos a los que denominaremos dimensiones.
  - Ejemplo: Si nos interesa saber cómo varió la opinión de las personas sobre nuestros distintos productos en los últimos meses, y dependiendo de su edad y ciudad de residencia, nuestras dimensiones serán: *(producto, mes, edad, ciudad)*.
  - En general las dimensiones tienen dominios discretos o bien sus valores estarán agrupados en un conjunto finito de rangos.
- Durante el modelado conceptual de un *data warehouse* debemos definir cuáles serán dichas dimensiones.
- A la medida numérica asociada a un valor concreto de cada una de las dimensiones la denominamos hecho.
  - *"En noviembre de 2016 en Rosario las personas de entre 40 y 60 años valoraban nuestro shampoo con 8.6 puntos"* es un hecho.

Modelado conceptual

Jerarquías de dimensiones

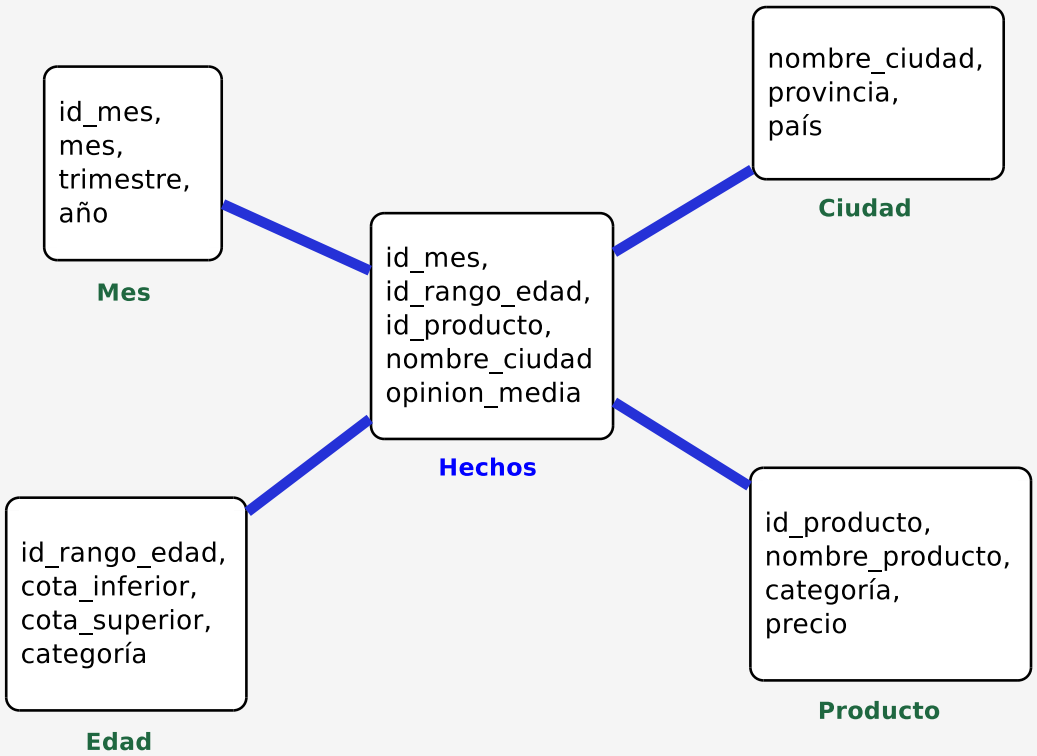
- En general cada dimensión tiene una serie de atributos asociados, que podemos representar a través de una relación:
  - Mes(id_mes, mes, trimestre, año)
    (75, 11, 4, 2016)
  - Edad(id_rango_edad, cota_inferior, cota_superior, categoría)
    (4, 40, 59, 'Adulto')
  - Producto(id_producto, nombre_producto, categoría, precio)
    (715, 'Shampoo', 'Higiene personal', 77.30)
  - Ciudad(nombre_ciudad, provincia, país)
    ('Rosario', 'Santa Fe', 'Argentina')
- A las tablas que describen las dimensiones en un *data warehouse* se las denomina tablas de dimensiones.
- A partir de estos atributos puede ser interesante agrupar los atributos de cada dimensión por jerarquías.
  - Ejemplos: *"Una ciudad pertenece a una provincia"*, *"Un producto es de una determinada una categoría"*.

Modelado conceptual

Diagama de estrella

[CONN15 32.4]

- El diagrama de estrella permite comunicar la estructura de hechos y dimensiones de un *data warehouse*.
- Representamos al conjunto de dimensiones y sus medidas agregadas en una tabla central, conocida como tabla de hechos, y la conectamos con cada una de las tablas de dimensiones.



Modelado lógico

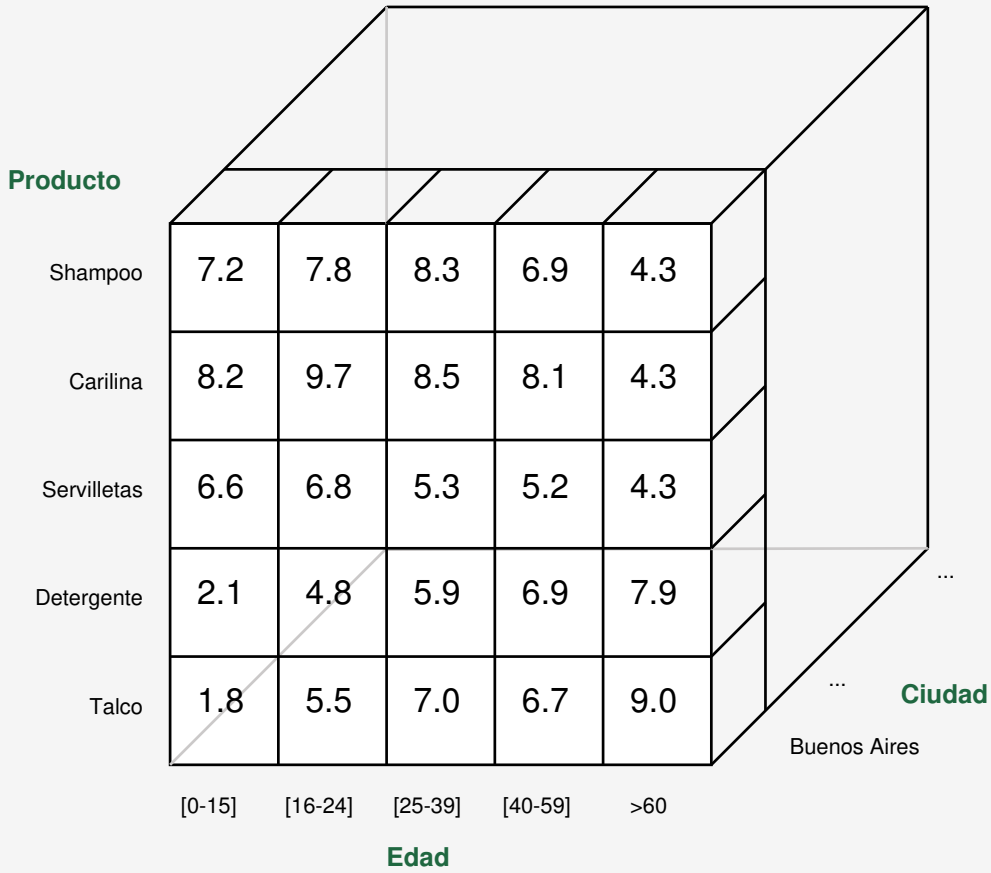
Implementaciones de OLAP

- Si bien en OLTP cada transacción se registra en filas distintas dentro de ciertas tablas, en OLAP la tabla de hechos guardará información sumarizada, de acuerdo a las dimensiones que nos interesará explorar.
- La forma de almacenamiento de las tablas de hechos sumarizadas depende de las distintas implementaciones de OLAP. Entre ellas:
  - MOLAP (Multidimensional OLAP): Los datos agregados se almacenan en una base de datos multidimensional ó cubo de datos. En general los mismos se precalculan en base al uso esperado.
  - ROLAP (Relational OLAP): La tabla de hechos agregada se almacena como una relación más. Es la implementación más habitual en los SGBD's relacionales.
  - HOLAP (Hybrid OLAP): Combina el uso de un SGBD relacional con un servidor MOLAP.

MOLAP

Cubos OLAP

- Un cubo de datos OLAP es una matriz multidimensional en que se almacena una medida agregada por una serie de dimensiones de un conjunto de datos.



ROLAP

Materialización de vistas

- En ROLAP, la tabla de hechos agregada (sumarizada) se guarda como una relación en un SGBD relacional.
- En vez de almacenar el resultado de la agregación en un arreglo multidimensional, sólo guardamos la *"SQL view"* que la define.
- Cada cierto tiempo la vista se ejecuta y se almacena su resultado (se "materializa").
- Existen distintas políticas sobre en qué momento materializar una vista para hacerla consistente con los datos almacenados:
  - Mantenimiento inmediato: La vista se actualiza cada vez que se ejecuta una nueva transacción sobre los datos.
  - Mantenimiento diferido:
    - Mantenimiento lazy: Cuando el usuario requiere los resultados.
    - Mantenimiento periódico: La vista se actualiza cada un cierto tiempo.
    - Mantenimiento forzado: Después de una cantidad fija de cambios.
- En cualquier caso, el SGBD puede realizar optimizaciones para que la materialización de la vista sea lo más eficiente posible.

Operaciones OLAP

Roll-up y drill-down

- La operación de roll-up consiste en agregar los datos de una dimensión, subiendo un nivel en su jerarquía. Por ejemplo, si tenemos el total de ventas por ciudad y producto, podríamos hacer un *roll-up* de la ciudad para obtener un total por provincia y producto.
- La operación contraria al roll-up es el drill-down. En este caso bajamos un nivel en la jerarquía de una de las dimensiones.

Operaciones OLAP

Pivoteo y slicing

- La operación de pivoteo consiste en producir una tabla agregada por un subconjunto del conjunto de dimensiones en cierto orden deseado. En nuestro ejemplo con las dimensiones *(producto, mes, edad, ciudad)* podríamos pivotear en (edad, producto), para obtener una medida agregada de la opinión que cada franja etárea tiene de cada uno de nuestros productos, *independientemente del período y de la ciudad*.
- La operación de slicing y dicing permiten realizar una selección en una dimensión (feta, o *slice*) o en más de una dimensión (dado, o *dice*). Por ejemplo, podríamos obtener la opinión agregada por producto, mes y edad sólo en la ciudad de Córdoba.

Soporte OLAP en SQL

- El estándar SQL-99 definió nuevas extensiones relacionadas con el procesamiento analítico en línea (OLAP).
- A continuación introduciremos las cláusulas `GROUPING SETS`, `ROLLUP` y `CUBE`.
- Luego, en SQL:2003 se agregaron cláusulas como `RANK` (índices de ranqueo para los resultados) y `OVER` (promedios con ventanas deslizantes), también útiles para el procesamiento analítico.
- Si bien no es SQL estándar, algunos SGBD's como Oracle, Sybase y PostgreSQL brindan el comando `CREATE MATERIALIZED VIEW` para materializar una vista y especificar la frecuencia con que la misma debe refrescarse.

La cláusula `GROUPING SETS`

- Supongamos que disponemos de datos sobre las ventas de una empresa estructurados de la siguiente forma:

factura	dia	mes	año	sucursal	producto	cantidad	monto
0035-0130	22	05	2017	Recoleta	Detergente	1	33.50
0035-0131	23	05	2017	Palermo	Gaseosa	5	100.00
0035-0131	23	05	2017	Palermo	Verdura	2	42.00
0035-0132	23	05	2017	Recoleta	Cerveza	8	215.00
0035-0132	23	05	2017	Recoleta	Pañales	2	156.00
0035-0132	23	05	2017	Recoleta	Verdura	1	21.00
0035-0132	23	05	2017	Recoleta	Shampoo	1	68.00
0035-0133	23	05	2017	Caballito	Shampoo	1	68.00
....							

- ¿Cómo podríamos construir una tabla que muestre el monto total vendido en cada sucursal y el monto total vendido por cada producto?

La cláusula `GROUPING SETS`

```
SELECT sucursal, NULL, SUM(monto) AS monto
FROM Ventas
GROUP BY (sucursal)
UNION
SELECT NULL, producto, SUM(monto) AS monto
FROM Ventas
GROUP BY (producto);
```

- La cláusula `GROUPING SETS` nos permite simplificar la sintaxis de este tipo de operatoria:

```
SELECT sucursal, producto, SUM(monto) AS monto
FROM Ventas
GROUP BY GROUPING SETS (sucursal, producto);
```

La cláusula `GROUPING SETS`

- El resultado será:

sucursal	producto	monto
Caballito		816
Palermo		197
Recoleta		729
	Cerveza	435
	Detergente	134
	Gaseosa	100
	Leche	50
	Manteca	42
	Pañales	698
	Shampoo	136
	Verdura	147

La cláusula `GROUPING SETS`

- Cada uno de los *grouping sets* nos define un conjunto de atributos de agrupamiento distinto. Por ejemplo, si quisieramos tener los montos vendidos por cada par (sucursal, producto), los subtotales por sucursal y el monto vendido total, podríamos ejecutar:

```
SELECT sucursal, producto, SUM(monto) AS monto
FROM Ventas
GROUP BY GROUPING SETS ((sucursal, producto), sucursal, ());
```

La cláusula `GROUPING SETS`

- Y el nuevo resultado será:

sucursal	producto	monto
Caballito	Cerveza	55
Caballito	Detergente	67
Caballito	Pañales	542
Caballito	Shampoo	68
Caballito	Verdura	84
Caballito		816
Palermo	Cerveza	55
Palermo	Gaseosa	100
Palermo	Verdura	42
Palermo		197
Recoleta	Cerveza	325
Recoleta	Detergente	67
Recoleta	Leche	50
Recoleta	Manteca	42
Recoleta	Pañales	156
Recoleta	Shampoo	68
Recoleta	Verdura	21
Recoleta		729
		1742

La cláusula `ROLLUP`

- La cláusula **`ROLLUP`** es una simplificación del **`GROUPING SETS`** que, dado un conjunto de atributos (es decir, un *grouping set*), realiza el agrupamiento por dicho conjunto y adicionalmente calcula los subtotales para cada subconjunto de atributos del conjunto, de derecha a izquierda.
- Por ejemplo, hacer **`GROUP BY ROLLUP (a,b,c)`** es equivalente a hacer:
  - `GROUP BY GROUPING SETS (a,b,c), (a,b), (a), ()`

La cláusula `CUBE`

- La cláusula **`CUBE`** es también una simplificación del **`GROUPING SETS`** que, dado un conjunto de atributos (*grouping set*), realiza el agrupamiento por dicho conjunto y adicionalmente calcula los subtotales para todo subconjunto de atributos del conjunto.
- Por ejemplo, **`GROUP BY CUBE (a,b,c)`** es equivalente a:
  - `GROUP BY GROUPING SETS (a,b,c), (a,b), (a, c), (b, c), (a), (b), (c), ()`

SQL

Ejercicio

Dada la base de datos de ventas de una empresa, normalizada a través de las siguientes tablas:

- DatosSucursales(id_sucursal, dirección, barrio)
  58, Av. Entre Ríos 380, Congreso
- DatosProductos(cod_producto, nombre, categoría)
  3341, Azurra, Cervezas
- DatosFacturas(cod_factura, dia, mes, año, id_cliente, id_sucursal)
  0035-41291, 28, 05, 2017, NULL, 58
- DetallesFacturas(cod_factura, cod_producto, cantidad, monto)
  0035-41291, 3341, 4, 140.00

Escriba una consulta que calcule el monto total vendido en cada categoría para cada mes del año 2016, indicando también los subtotales por mes y el total anual.

SQL

Ejercicio

Solución

SELECT mes, categoría, SUM(monto) AS monto

FROM DetallesFacturas detf, DatosFacturas df, DatosProductos dp

WHERE detf.cod_factura=df.cod_factura

AND detf.cod_producto=dp.cod_producto

AND año=2016

GROUP BY ROLLUP (mes, categoría);

Bibliografía

[ELM16] Fundamentals of Database Systems, 7th Edition.
R. Elmasri, S. Navathe, 2016.
Capítulo 29.

[CONN15] Database Systems, a Practical Approach to Design, Implementation and Management, 6th Edition.
T. Connolly, C. Begg, 2015.
Capítulo 31, Capítulo 32, Capítulo 33

[GM09] Database Systems, The Complete Book, 2nd Edition.
H. García-Molina, J. Ullman, J. Widom, 2009.
Capítulo 10.[6-7].

Bibliografía

[SILB19] Database System Concepts, 7th Edition.
A. Silberschatz, H. Korth, S. Sudarshan, 2019.
Capítulo 11

[TDWT] The Data Warehouse Toolkit, 3rd Edition.
R. Kimball, M. Ross, 2013.