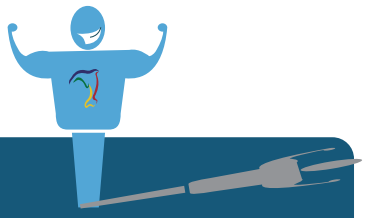
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

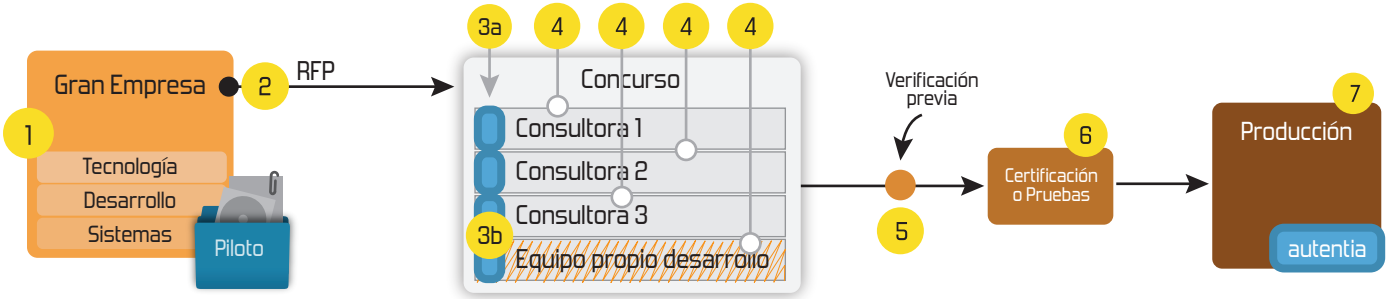
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

Ver cursos que ofrece Autentia

Descargar comics en PDF y alta resolución

[¡NUEVO!] 2008-12-01

2008-11-17

2008-09-01

2008-07-31

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la seccion comic.

Tutorial desarrollado por

Alfonso Blanco Criado

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero Informático *

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (17 MB)

AdictosAlTrabajo.com es el Web de difusión de conocimiento de Autentia.

autentia real business solutions

Catálogo de cursos

Descargar este documento en formato PDF: vistasMaterializadas.pdf

Fecha de creación del tutorial: 2009-02-26

Vistas materializadas

En este tutorial vamos a ver lo que son las **vistas materializadas** (*materialized views*) y vamos a hacer un ejemplo de creación de una vista materializada y su posterior uso. Lo primero de todo es recordar lo que es una vista en un modelo de base de datos relacional.

Una vista es una consulta almacenada que representa un conjunto de tablas (posiblemente de diferentes esquemas) a la que le vamos a poner un nombre y vamos a tratarla como si fuese una tabla más de nuestro esquema, pero sin llegar a ser realmente una tabla. Algo que tiene que quedar claro es que una vista NO guarda datos, sino que solo almacena la consulta que nos va a ayudar a acceder a los datos. Pero, ¿porqué usar vistas? es muy sencillo, principalmente hay dos motivos. El primer motivo es de seguridad, a lo mejor no necesitamos que determinados usuarios acceden a toda la información de la base de datos y simplemente queremos formar un conjunto de datos específicos para dichos usuarios. El segundo motivo tiene que ver con la estructura de nuestro modelo de datos, ya que si es bastante complejo o con muchas tablas nos puede ser muy útil crear este tipo de vistas para organizar una cierta información de modo que nos sea mucho más cómodo acceder a ella mediante consultas mucho más sencillas.

Definición

A diferencia de las vistas "normales" una vista materializada **almacena físicamente** los datos resultantes de ejecutar la consulta definida en la vista. Este tipo de vistas materializadas realizan una carga inicial de los datos cuando se definen y posteriormente con una frecuencia establecida se actualizan los datos de la misma.Con la utilización de vistas materializadas logramos aumentar el rendimiento de las consultas *SQL* además de ser un método de optimización a nivel físico en modelos de datos muy complejos y/o con muchos datos.

Una vez definida una vista materializada uno de los problemas que nos encontramos es el de la **actualización de los datos**. Como se ha comentado antes, estas vistas contienen fisicamente los datos de las "tablas base", por lo que si cambian los datos de estas tablas no se reflejarán en la vista materializada. Para ello necesitamos establecer un mecanismo de resfresco automático en el que tendremos que definir el **tipo** y la **forma** de refresco.

La sentencia *SQL* que nos permite definir una vista materializada es esta:

view plain print ?

```
01. CREATE MATERIALIZED VIEW nombre_vista
02.   [TABLESPACE nombre_ts]
03.   [PARALELL (DEGREE n)]
04.   [BUILD {INMEDIATE|DEFERRED}]
05.   [REFRESH {FAST|COMPLETE|FORCE|NEVER}|{ON COMMIT|ON DEMAND|[START WITH fecha_inicio] NEXT intervalo}]
06.   [{ENABLE|DISABLE} QUERY REWRITE]
07.   AS SELECT ... FROM ... WHERE ...
```


Últimos tutoriales

Últimas ofertas de empleo

Con la palabra *BUILD* establecemos la forma de carga de datos en la vista. Con la opción *INMEDIATE* (opción por defecto) se cargarán los datos justo después de crear la vista, mientras que con la opción *DEFERRED* se definirá la vista cuando se ejecute la sentencia *SQL* sin cargar ningún dato, que se cargarán cuando se realize el primer refresco de la vista.

Con la palabra *REFRESH* definimos el método y la frecuencia de refresco de los datos.

La palabra *QUERY REWRITE* establece si queremos que el **optimizador** de nuestra base de datos pueda reescribir las consultas. El optimizador, sabiendo que ya existe una determinada vista materializada, puede modificar internamente nuestra consulta sobre una determinada tabla, de tal forma que se mejore el rendimiento de la consulta devolviendo los mismos datos que la consulta original.

Refresco

Como es entendible la política de refresco de cada vista repende altamente de nuestras necesidades y requerimientos sobre la frecuencia de actualización de los datos de las "tablas base".

Tipos de refresco

- **COMPLETE** : se borrarán todos los datos de la vista y se volverá a ejecutar la consulta definida en la vista por lo que se recargarán fisicamente los datos de las "tablas base".
- **FAST** : podemos decir que este tipo de refresco es una actualización incremental, es decir, solo se refrescarán aquellos datos que se hayan modificado desde el último refresco. Evidentemente este tipo de refresco es mucho más *fast* ;-) que el *complete*. Pero, ¿cómo sabe la base de datos que datos se han modificado desde el último refresco? lo sabe gracias a que previamente hemos tenido que crear unos determinados ***log*** de la vista (*VIEW LOG*) sobre cada una de las "tablas base" de la vista materializada.

```
view plain print ?
01. CREATE MATERIALIZED VIEW LOG ON tabla_base
02.   WITH PRIMARY KEY
03.   INCLUDING NEW VALUES;
```

  Hay que decir que si usamos funciones *sum, avg, max, min*, etcétera, no vamos a poder usar este tipo de refresco.

- **FORCE** : si se puede realizar el refresco tipo *FAST* se ejecuta, y sino se realiza el refresco *COMPLETE*. Es el valor por defecto del tipo de refresco.
- **NEVER** : nunca se realizará un refresco de la vista.

Formas de refresco

- **Refresco manual :** mediante el paquete de *PL/SQL DBMS_MVIEW* podemos forzar a realizar un refresco usando para ello la función *REFRESH*.

```
view plain print ?
01. DBMS_MVIEW.REFRESH ('nombre_vista');
02.
```

  Con la función *REFRESH_DEPENDENT* se refrescarán todas las vistas materializadas que tengan algunas de sus "tablas base" en la lista de tablas pasada como parámetro de entrada.

```
view plain print ?
01. DBMS_MVIEW.REFRESH_DEPENDENT ('tabla1, tabla2, tabla3, ... , tablaN');
02.
```

  Con la función *REFRESH_ALL_MVIEWS* se refrescarán todas las vistas materializadas de nuestra base de datos.

- **Refresco automático :** este refresco automático podemos hacerlo usando la palabra *ON COMMIT*, con la que se fuerza al refresco de la vista en el momento en el que se haga un *commit* sobre una de las "tablas base" de dicha vista. Otro tipo de refresco automático es el llamado refresco programado, en el cual podemos definir el momento exacto en el que queremos que se refresque nuestra vista. Para ello tenemos que definir la fecha del refresco en formate *datetime* y el intervalo de este.

Ejemplo práctico

Una vez visto todos los detalles teóricos vamos a hacer un pequeño ejemplo práctico para ver como definir vistas materializadas y analizar sus comportamientos.

Lo primero que vamos a hacer es crear una tabla (*SQL*) en nuestra base de datos (*Oracle 9*) y insertar algunos datos.

```
view plain print ?
01. CREATE TABLE tabla
02. (campo1 int PRIMARY KEY,
03. campo2 int);
04.
05. insert into tabla values (1,2);
06. insert into tabla values (2,298);
07. insert into tabla values (3,223);
08. insert into tabla values (4,121);
09. insert into tabla values (5,34);
10. insert into tabla values (6,34);
11. insert into tabla values (7,78);
12. insert into tabla values (8,44);
13. insert into tabla values (9,34);
14. insert into tabla values (10,12);
```

A continuación creamos un *VIEW LOG* para la anterior tabla.

```
view plain print ?
01. CREATE MATERIALIZED VIEW LOG ON tabla
02. WITH PRIMARY KEY
03. INCLUDING NEW VALUES;
```

Despúes creamos nuestra vista materializada.

```
view plain  print  ?
01. CREATE MATERIALIZED VIEW tabla_vm
02. BUILD IMMEDIATE
03. REFRESH FAST ON COMMIT
04. AS SELECT * FROM tabla;
```

Como se puede ver, la definición de la vista es absurda, ya que es la propia tabla, pero para nuestro ejemplo nos servirá ya que lo que se pretende es ver el comportamiento de la vista materializada creada.

En la anterior sentencia *SQL* creamos una vista materializada de nombre **tabla_vm** que se cargará inicialmente justo cuando se ejecute (*BUILD IMMEDIATE*), se refrescarán solo aquellas entradas que se hallan modificado en la tabla base (*FAST*) y lo harán cuando se ejecute la acción de *commit* (*ON COMMIT*).

Vamos a ver si realmente la teoría se corresponde con la práctica. Inicialmente tanto la tabla base como la vista materializada contienen estos valores:

CAMPO1	CAMPO2
1	2
2	298
3	223
4	121
5	34
6	34
7	78
8	44

Cuando ejecutamos una sentencia sobre la tabla base que actualize un determinado dato de ella (*update tabla set campo2 = 123 where campo1 = 1*) podemos ver que se ha actualizado en la tabla base pero no en la vista materializada ya que aun no se ha lanzado el refresco.

tabla base - tabla

CAMPO1	CAMPO2
1	123
2	298
3	223
4	121
5	34
6	34
7	78
8	44

vista materializada - tabla_vm

CAMPO1	CAMPO2
1	2
2	298
3	223
4	121
5	34
6	34
7	78
8	44

Para que la vista se refresque y se actualize solo con el dato modificado en la tabla base tenemos que hacer *COMMIT*.

CAMPO1	CAMPO2
1	123
2	298
3	223
4	121
5	34
6	34
7	78
8	44

Como se puede ver en la imagen de arriba ya tenemos los datos totalmente actualizados en nuestra vista materializada.

Este pequeño ejemplo no tiene valor para poder analizar la mejora en el rendimiento al usar este tipo de vistas pero nos ha servido (...o eso espero...) para que se vea como se puede crear una vista materializada y como es su funcionamiento.

Para borrar la vista materializada creada tenemos que ejecutar la siguiente sentencia *SQL*.

```
view plain  print  ?
01. drop materialized view tabla_vm;
```

Aquí dejo los scripts *SQL* utilizados.

Conclusión

Como ya se ha comentado anteriormente la definición de este tipo de vista solo tiene sentido para modelos de datos muy complejos y con muchos datos, ya que si no es así no se percibiría apenas la mejora en el rendimiento y lo que provocaríamos es tener una serie de datos físicamente en nuestra base de datos sin mucho sentido.

Ahora viene lo mejor....además estás vistas materializadas no están disponibles para todas las bases de datos del mercado, solo se pueden usar con base de datos Oracle, DB2, Informix, Microsoft SQL Server (ellos las llaman vistas indexadas y son muy similares), Adaptative Server Enterprisa y ANTs Data Server.

Aquí os dejo el enlace a la documentación oficial de **Oracle** en relación a las vistas materializadas (*materialized views*)

- Puedes opinar sobre este tutorial haciendo clic aquí.
- Puedes firmar en nuestro libro de visitas haciendo clic aquí.
- Puedes asociarte al grupo AdictosAlTrabajo en XING haciendo clic aquí.
- Añadir a favoritos Technorati. ADD THIS BLOG TO MY Technorati FAVORITES

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subcripción a novedades:

E-mail [ ] Aceptar

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
MySql en Windows	MySql es una de las principales bases de datos "gratuitas" que podemos encontrar en Internet. En este tutorial aprendereis a instalarlo en Windows	2003-06-23	59033	Muy bueno	2	
CMP Entity Beans y MySql	Os mostramos como crear un Entity Bean con persistencia controlada por el servidor, configurado para usar MySql	2003-10-03	22688	Bueno	1	
Introducción a HSQL	Este tutorial presenta una sencilla introducción al motor de bases de datos HSQL	2006-05-09	11420	Bueno	3	
Modelado de MySQL con herramientas gratuitas	Os mostramos otra alternativa de modelado gráfico de MySql.	2004-01-25	26404	Bueno	2	
PostgreSQL 8.0 en Windows	Os mostramos como instalar la versión 8 de PostgreSQL en entorno Windows	2005-01-25	24665	Regular	3	
Administración Web de MySQL	En este tutorial se mostrará como administrar MySQL de forma rápida y muy sencilla a través de páginas webs implementadas con tecnología PHP, para ello se utilizará la herramienta PHPmyAdmin	2007-04-03	5295	-	-	
Modelado Gráfico de MySQL	Os mostramos como instalar y utilizar DeZing e Importer de Datanamic para crear el modelo lógico y físico de bustra base de datos MySQL	2004-01-07	22167	-	-	
Administracion Web de MySQL	Os mostramos como instalar en vuestro PC phpMySQL, un potente gestor Web de MySQL utilizado en la mayoría de los Hostings.	2003-12-27	17595	-	-	
Realización de DTS, Backups y Restores utilizando MS SQL SERVER	En el siguiente tutorial pretendemos mostrar, de forma general, varias de las herramientas incluidas en el servidor de bases de datos de Microsoft: MS SQL Server, para este caso concreto veremos la herramienta para la realización de transformaciones de da	2007-01-02	6030	-	-	
Apache, MySQL y PHP	Os mostramos como configurar Apache, MySQL y PHP en vuestra máquina	2003-12-27	41576	-	-	

Nota: