Artigo

MYSQL – Trabalhando com Views

Neste artigo serão apresentados os principais conceitos de armazenamento de procedimentos no servidor de bancos de dados MySQL.

Neste artigo serão apresentados os principais conceitos de armazenamento de procedimentos no servidor de bancos de dados MySQL, além de mostrar como trabalhar essencialmente com objetos chamados Views. Além dos conceitos que serão vistos, aprenderemos também, como montar nossas próprias visões dos dados de tabelas em um banco de dados.

Para os exemplos que criaremos durante o artigo, utilizaremos o banco de dados

Introdução ao MySQL

Para quem trabalha com desenvolvimento de sistemas e administração de dados diretos em bases de dados concentradas em um SGBD como o MySQL, Oracle ou SQL Server, sabe o quanto é chateante ter que escrever e reescrever determinadas consultas todos os dias ou mesmo mais de uma vez no mesmo dia. Muitas destas consultas são derivadas de várias tabelas o que nos dá um re-trabalho ao montar todas aquelas JOIN's, utilizar esse ou aquele índice setado para esta ou aquela tabela para que também a performance de tal consulta tenha um tempo razoavelmente atraente.

Sabemos também que a cada momento em que reescrevemos uma mesma consulta, conseguir os mesmos resultados de antes é uma tarefa séria, já que existem várias formas para se escrever uma mesma consulta, levando em conta a ordem em que as tabelas aparecem, a ordem dos campos e tudo mais. Tudo isso implica em ganho ou perda de performance. Estamos falando tanto em performance que parece que nosso artigo tomou outros rumos, mas não. De fato, quando temos um grande trabalho de ajuste de performance em uma consulta, podemos rapidamente transformar esta em uma View, que a partir disso, permanecerá armazenada no servidor de bancos de dados em forma de tabela, para que possamos consultá-la todas as vezes que precisarmos, sem ter que reescrever a mesma, aproveitando todo o trabalho de refino de performance supracitado.

tables" ou ainda outras Views. E alguns casos, as Views são atualizáveis e podem ser alvos de declaração `INSERT`, `UPDATE` e `DELETE`, que na verdade modificam sua "based tables".

Os benefícios da utilização de Views, além dos já salientados, são:

- Uma View pode ser utilizada, por exemplo, para retornar um valor baseado em um identificador de registro;
- Pode ser utilizada para promover restrições em dados para aumentar a segurança dos mesmos e definir políticas de acesso em nível de tabela e coluna. Podem ser configurados para mostrar colunas diferentes para diferentes usuários do banco de dados;
- Pode ser utilizada com um conjunto de tabelas que podem ser unidas a outros conjuntos de tabelas com a utilização de `JOIN's` ou `UNION`.

Criando Views

Para definir Views em um banco de dados, utilize a declaração `CREATE VIEW`, a qual tem a seguinte sintaxe:

```
1 CREATE [OR REPLACE] [ALGORITHM = algorithm_type] VIEW  VIEW view_nam
2 AS select_statement
3 [WITH [CASCADED | LOCAL] CHECK OPTION]
```

- `column_list` : recurso para que possamos sobrescrever os nomes das colunas que serão recuperadas pela declaração SELECT;
- `select_statement` : é a sua declaração `SELECT` e indica a forma na qual você deseja que os dados sejam retornados. Tal declaração deverá indicar a forma a qual você deseja retornar os dados, podendo ser utilizado funções, `JOIN's` e `UNION` . Podem ser utilizadas quaisquer tabelas ou views contidas no servidor de bancos de dados `MySQL` , observando a questão de nomes totalmente qualificados ou não.
- `OR REPLACE` : pode ser utilizada para substituir uma View de mesmo nome existente no banco de dados ao qual ela pertence. Pode-se utilizar `ALTER TABLE` para o mesmo efeito;
- `ALGORITHM` : essa cláusula define qual algoritmo interno utilizar para processar a View quando a mesma for invocada. Estes podem ser `UNDEFINED` (cláusula em branco), `MERGE` ou `TEMPTABLE` .
- `WITH CHECK OPTION` : todas as declarações que tentarem modificar dados de uma view definida com essa cláusula serão revisadas para checar se as modificações respeitarão a condição `WHERE` , definida no `SELECT` da View.

Ao criar uma View, um arquivo com extensão “ `.frm` ”, com o mesmo nome desta também é criado no diretório do banco de dados, sob o diretório de dados do MySQL (`datadir`).

Definindo Views

Para iniciarmos com a mão na massa, definiremos uma View simples para listar

```
7 FROM vw_viewCity
8 LIMIT 3;
```

Código 2. Criando uma view simples e executando a mesma com um SELECT

Podemos explicar que, a partir desse momento, se dermos um `SHOW TABLES` no banco de dados `World`, veremos que uma tabela adicional foi criada, que é a View que criamos. Para uma conceituação mais ampla, uma View é um mapeamento lógico de várias tabelas contidas em um ou mais bancos de dados que por sua vez estão em um servidor MySQL. No caso da View criada na **Figura 2**, temos uma tabela virtual (`vw_viewCity`) baseada em uma tabela chamada de base (**City**). Um bom exemplo para utilizarmos a tal lista de colunas é criar a mesma View, mas agora sobrescrevendo o nome da coluna “Name” para “Cidade”, como segue no **Código 3**.

```
1 CREATE VIEW vw_viewCity AS
2 SELECT Name
3 FROM City;
```

Código 3. Erro! O que houve?

Nossa intenção foi boa, mas faltou um pouco mais de análise. Teremos que sobrescrever também a View, pois já existe uma outra com o mesmo nome. Portanto, usaremos o `OR REPLACE` para facilitar nossa vida, **Código 4**.

```
1 CREATE OR REPLACE VIEW vw_viewCity AS
2 SELECT Name
3 FROM City;
```

Podemos criar Views mais sofisticadas, com um comando `SELECT` mais trabalhado, podendo utilizar das cláusulas `WHERE`, `GROUP BY`, `HAVING` e `ORDER BY`. Alguns SGBD's comerciais não permitem a utilização de `ORDER BY` em meio ao `SELECT` na definição de uma View, a exemplo do SQL Server, da Microsoft. O **Código 5** mostra uma consulta mais trabalhada para a criação de uma View, envolvendo as tabelas Country e CountryLanguage do banco de dados World.

```
1  CREATE VIEW vw_countryLangCount AS
2  SELECT Name, Count(Language) AS LangCount
3  FROM Country, CountryLanguage
4  WHERE Code = CountryCode
5  Group by Name;
6
7  SELECT * FROM vw_countryLagCount LIMIT 3;
```

Código 5. Criando uma view mais trabalhada

Atualizando Views

Views podem ser constituídas facilmente, como vimos anteriormente. Mas para termos Views que podem receber declarações de atualização tais como `UPDATE` e `DELETE`, que de fato alteram as tabelas base ("based tables"), temos que ter alguns cuidados na criação do objeto de visualização. Uma View criada com funções agregadas, por exemplo, não poderá receber atualizações, pois os dados logicamente estão agregados ou agrupados e não teremos correspondências diretas para uma exclusão ou atualização.

país na tabela `CountryLanguage`. Isso não nos possibilita atualizar uma determinada linha, pois, o que retornou dessa consulta foi um conjunto agrupado por país.

Quando temos uma View com um `SELECT` simples, sem agrupamento, podemos atualizá-la, esta que na verdade, somente receberá a declaração e quem serão atualizadas serão as tabelas base que tem seus dados mapeados para esta View. Para que você não estrague seu banco de dados World, criaremos uma tabela chamada "`CountryPop`", com seguinte sintaxe mostrada no **Código 6**.

```
1 CREATE TABLE CountryPop
2
3 SELECT Name, Population, Continent
4 FROM Country;
```

Código 6. Criamos a tabela na qual utilizaremos como based table para uma View atualizável

Com a tabela base criada, vamos então definir a View para que possamos atualizar a mesma. O **Código 7**mostra a View sendo definida, faremos um `SELECT` na View para exibir a linha que atualizaremos e na seqüência emitimos um `UPDATE` para que a mesma seja atualizada.

```
1 CREATE TABLE EuroPop AS
2 SELECT Name, Population
3 FROM CountryPop
4 WHERE Continent = 'Europe';
5
6 UPDATE EuroPop
```

Código 7. Atualizando uma View

Views com WITH CHECK OPTION

Podemos ainda trabalhar algumas restrições na criação de algumas Views, como utilizar a opção `WITH CHECK OPTION`. Atribuindo esta opção na criação de uma View significa que atualização que são emitidas terão que se encaixar às condições definidas na cláusula `WHERE` da consulta `SELECT`. Peguemos um exemplo de uma View que nos retorne todos os países da tabela CountryPop que acabamos de criar, que tenha populações maior ou igual a 100000000 (**Código 8**).

```
1  CREATE VIEW vw_largePop AS
2  SELECT Name, Population
3  FROM CountryPop
4  WHERE Population >= 100000000
5  WITH CHECK OPTION;
6
7
8  SELECT *
9  FROM vw_largePop;
```

Código 8. View com WITH CHECK OPTION

Definimos na cláusula `WHERE` da View que somente poderia ser atualizado o número da população do país se este comando de atualização enviasse um número maior ou igual a `100000000`. No **Código 9**, enviaremos um `UPDATE` para atualizar a população da **Nigéria** para `9999999`, que é ligeiramente menor que o número definido no `WHERE` da View.

`WITH CHECK OPTION` somente será aceita em meio a uma View atualizável, caso aquela que você vier a definir não seja atualizável, um erro será enviado e a mesma não será criada.

DROP VIEW

Para excluir uma View, basta utilizar o comando `DROP VIEW view_name`.

Vimos então como trabalhar com View ou visualizações que nada mais são que mapeamentos lógicos de outras tabelas em uma nova, definido com comandos `SELECT's`. Vimos como criar Views, como alterar ou sobrescrever as mesmas e trabalhar com restrições ao atualizá-las. Num próximo artigo sobre Views, veremos como recuperar metadados, como criar Views e dar permissões nas mesmas parcialmente e os privilégios requeridos para um usuário cria-las.

Links Úteis

- MYSQL: Site Oficial do Mysql

Tecnologias:

Banco de Dados MySQL SQL

Falar com professor - Tire a sua dúvida.