Distribuição e Fragmentação de Bases de Dados

por Pablo Vieira Florentino

Projeto de Distribuição de Bases de Dados

Devido à expansão de sistemas em rede e da Internet, sistemas distribuídos de natureza comercial, principalmente Sistemas Gerenciadores de Banco de Dados (SGBD), começaram a preencher as necessidades de grandes organizações. Sistemas desta natureza correspondem melhor às necessidades da estrutura organizacional de instituições geograficamente e amplamente distribuídas, uma vez que os mesmos provêem melhor desempenho e são mais confiáveis e disponíveis. Mesmo assim, instituições geograficamente centralizadas podem se beneficiar da distribuição de seus sistemas, balanceando o processamento entre diversos sites internos à sua estrutura. É cada vez mais simples e fácil a implantação de um sistema de banco de dados distribuído em face do constante declínio no custo de elementos de processamento e memória, popularização dos computadores pessoais e aumento da escalabilidade dos sistemas através de maior facilidade para expansões e integração de recursos diferenciados, permitindo a montagem de diferentes sites. Aplicações para bancos e grandes lojas de departamentos são exemplos clássicos beneficiados com a tecnologia de sistemas distribuídos, em especial SGBD distribuídos (SGBDD). O Projeto de Distribuição de Bases de Dados (PDBD) visa definir esquemas de fragmentação e alocação de bases de dados de acordo com os acessos feitos pelas aplicações, de forma a otimizar tais acessos.

Existem duas estratégias para a distribuição de bases de dados: ascendente e descendente. A abordagem ascendente é comumente aplicada sobre bases de dados legados, em que a tarefa de modelagem objetiva integrar as bases existentes. Este tipo de ambiente é geralmente caracterizado por sistemas heterogêneos.

No caso da estratégia descendente, o processo envolve atividades básicas para desenvolvimento de sistemas como análise de requisitos e captura de informações, gerando um esquema global, até a implementação do projeto físico de distribuição em projetos locais. Esta abordagem é aconselhável para sistemas de bases de dados que estejam sendo projetados no início da modelagem da aplicação. No entanto, a distribuição de bases de dados é um processo dinâmico e evolutivo, devendo ser sempre revisado a partir de histogramas coletados no SGBD.

Neste artigo detalharemos a estratégia descendente, na qual o projeto de distribuição possui duas etapas principais: a etapa de fragmentação e a etapa de alocação. A etapa de fragmentação define o esquema de fragmentação de uma base de dados, onde para cada tabela da base é definida a técnica de fragmentação a ser aplicada, e são determinados os fragmentos da tabela. Para esta etapa, podem ser utilizadas técnicas básicas de fragmentação, como a fragmentação horizontal (FH) e a fragmentação vertical (FV). Estas duas técnicas podem ser combinadas e utilizadas de forma conjunta na mesma tabela (fragmentação híbrida (FHib)), ou em tabelas distintas (fragmentação mista). Escolher qual a técnica mais apropriada para fragmentar cada tabela da base de dados é um dos grandes problemas que atualmente apresenta apenas algumas propostas de soluções. Para realizar a fragmentação das tabelas da base é necessário analisar diversas informações relativas à aplicação como: as consultas mais solicitadas e suas respectivas freqüências; características das tabelas que formam a base de dados, seus relacionamentos com outras tabelas e suas cardinalidades (quantidade de registros existentes em uma tabela).

A etapa de alocação indica como os fragmentos gerados devem ser alocados nos diferentes sites do sistema baseando-se em informações como localização das aplicações e tamanho das tabelas, freqüências e padrões de acesso.

Etapa de Fragmentação

Na estratégia descendente, a fragmentação é a primeira etapa no PDBD. Seu objetivo é definir fragmentos ou partições de dados que servirão de subsídio para a etapa posterior de alocação. Na etapa de fragmentação, para cada tabela da base de dados é definida a técnica de fragmentação mais adequada, onde as partições da tabela são especificadas de acordo com a técnica escolhida.

Fragmentação Horizontal

Os fragmentos da tabela a ser fragmentada podem ser determinados através de funções de fragmentação definidas de acordo com as características da aplicação. Por exemplo, uma tabela de Empregados a ser fragmentada em duas partições: aqueles que possuem salário acima de 5000 e aqueles com salário menor que 5000 ou exatamente este valor. Teria-se então a definição das duas partições de Empregado definidas por duas funções de fragmentação distintas:

$$Empregado_1 = F_{[salário > 5000]}(Empregado)$$

$$Empregado_2 = F_{[salário <= 5000]}(Empregado)$$

A fragmentação horizontal pode ser classificada em dois tipos: horizontal primária (FHP) e horizontal derivada (FHD). A primeira aplica-se criando agrupamentos de acordo com as características dos registros da própria tabela analisada. O exemplo de fragmentação da tabela Empregado ilustra a aplicação da FHP. No segundo caso, os fragmentos são definidos de acordo com características da FHP de uma outra tabela relacionada. Por exemplo, uma tabela Dependente, relacionada com a tabela Empregado, seria fragmentada de forma derivada em relação a Empregado originando os seguintes fragmentos:

- $Depend_1$ = Dependente $\ltimes$ $Empregado_1$
- $Depend_2$ = Dependente $\ltimes$ $Empregado_2$

A fragmentação horizontal primária pode ser definida como a divisão de uma tabela de dados em subconjuntos de seus registros. A tabela é seccionada horizontalmente, criando partições formadas por grupos de instâncias que obedecem a uma determinada função de fragmentação.

A FHP pode ser aplicada de três diferentes formas (

Figura **1**):

- **Fragmentação por faixas de valores** - as funções de fragmentação neste caso se baseiam nas cláusulas condicionais (predicados lógicos) que podem ser utilizadas em uma consulta para determinar as partições. Por exemplo, a consulta "select nome from Professor where idade < 70" define o predicado lógico "idade < 70" sobre a tabela Professor e indica a aplicação de uma fragmentação horizontal por faixa de valor sobre esta tabela, sugerindo a criação de duas partições: uma com professores com idade abaixo de 70 e outra com professores com idade igual ou maior que 70. Assim, as funções de fragmentação horizontal FH1 e FH2 sobre a tabela Professor seriam definidas por:

  $FH_1 = (idade < 70)$ e $FH_2 = (idade >= 70)$.

- **Funções de Dispersão (Funções Hash),** que acontece mediante a aplicação de uma determinada função *hash* sobre o valor de algum atributo da tabela a ser fragmentada. Neste caso, o resultado da aplicação da função *hash* vai então indicar a que fragmento da tabela o registro será direcionado. A própria função *hash* representa a função de FH.
- **Fragmentação Circular (Round-Robin),** na qual distribuem-se os registros da tabela entre os fragmentos de uma forma seqüencial e circular à medida que são inseridos, sendo esta técnica indicada para acessos que utilizam leituras seqüenciais dos dados. Esta forma de fragmentação é bastante utilizada em sistemas que exploram processamento paralelo.

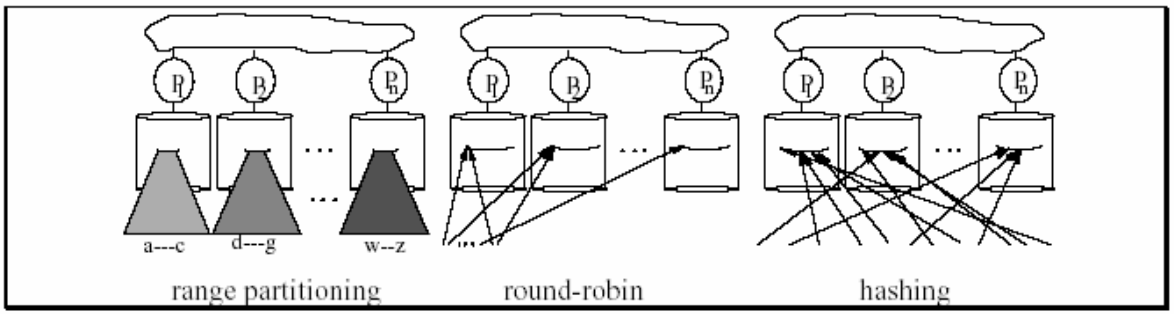

Figura 1 - Os três esquemas básicos para fragmentação horizontal de dados

Para aplicar a fragmentação horizontal derivada (FHD) é necessário ter conhecimento sobre os relacionamentos existentes entre as tabelas, muitas vezes acessados pelas aplicações através de junções (joins) existentes nas consultas. Estes relacionamentos podem ser classificados como relacionamentos Owner-Member, que caracterizam especificamente os relacionamentos 1 x n Neste caso, suponha duas tabelas A e B, e um relacionamento 1 x n entre elas, onde cada registro da tabela A é relacionado com muitos registros da tabela B, e cada registro da tabela B é relacionado com somente um registro da tabela A. Neste caso, a tabela A é denominada owner, enquanto a tabela B é denominada member.

Com a identificação deste tipo de relacionamento e das tabelas que o compõem, é possível aplicar a FHD. Com esta abordagem, as coleções owner são fragmentadas de forma horizontal primária, enquanto que as coleções member são fragmentadas de forma horizontal derivada de acordo com a FHP especificada sobre sua tabela owner.

Dado um relacionamento no qual A é a tabela owner e B a tabela member, os fragmentos horizontais derivados de B são definidos de acordo com os fragmentos primários de A. Como exemplo, dadas duas coleções Departamento e Professor, sabendo-se que as mesmas possuem um relacionamento 1 x n, no qual 1 departamento possui n professores, e que a tabela Departamento foi indicada para FHP em três partições (Dep1, Dep2, Dep3), poderia-se aplicar a FHD na tabela Professor, definindo os seus fragmentos horizontais derivados da seguinte forma:

- $Prof_1$ = Professor $\ltimes$ Dep_1
- $Prof_2$ = Professor $\ltimes$ Dep_2
- $Prof_3$ = Professor $\ltimes$ Dep_3

Com os fragmentos derivados criados, uma prática comum é, na etapa de alocação, armazenar cada fragmento horizontal derivado junto ao fragmento horizontal primário correspondente, como ilustrado na figura abaixo.

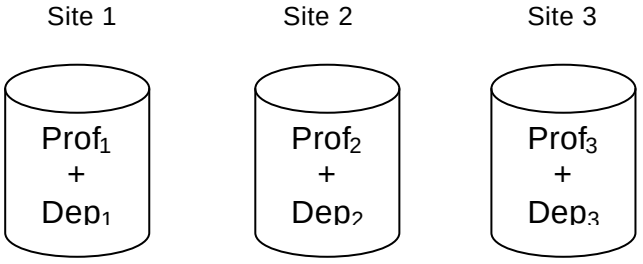

Figura 2 – Alocação sugerida para fragmentação horizontal derivada

Fragmentação Horizontal

A fragmentação vertical (FV) tem como objetivo definir partições formadas por subconjuntos de atributos de uma tabela, bem como pelo seu identificador (chave primária), dividindo-a verticalmente, de tal modo que os aplicativos do usuário possam atuar apenas sobre determinados fragmentos de atributos, isolando os acessos mais freqüentes. Assim, é possível diminuir a quantidade de dados acessada e, por conseguinte, a quantidade de memória necessária para manipular os dados. Como exemplos de áreas de aplicação que se beneficiam desta técnica de fragmentação podemos citar as áreas de sistemas de informações geográficas e sistemas multimídias, permitindo-se que os acessos a dados textuais, e de menor tamanho, fossem separados dos acessos aos tipos de dados mais pesados, características pertinentes a estes tipos de sistemas.

Na FV as estruturas lógicas das tabelas são quebradas gerando partições disjuntas (não sobrepostas). Para que seja possível a reconstrução da tabela original, a chave primária (ou o identificador) é repetida em todas as partições verticais. Ainda utilizando como exemplo a tabela hipotética Empregado com o seguinte conjunto de atributos {Cpf, Nome, Endereço, Telefone, Celular, Cargo, Idade, Escolaridade, Salário}, a

mesma pode ser fragmentada verticalmente em duas partições: Emp1(Cpf, Nome, Endereço) e Emp2(Cpf, Telefone, Celular, Cargo, Idade, Escolaridade, Salário). Assim, a função de fragmentação vertical pode ser representada por:

$FV_1 = p$ *(Cpf, Nome, Endereço)*

$FV_2 = p$ *(Cpf, Telefone, Celular, Cargo, Idade, Escolaridade, Salário)*

Fragmentação Mista e Híbrida

As técnicas de FH e FV podem ser combinadas na fragmentação de uma base de dados de duas maneiras: via fragmentação mista e/ou via fragmentação híbrida. No caso da fragmentação mista, aplicam-se técnicas de fragmentação distintas a cada uma das coleções da base de dados. A fragmentação híbrida, por sua vez, é caracterizada pela aplicação, sobre uma mesma tabela, tanto de FH como de FV, de forma a garantir que a tabela seja fragmentada atendendo a todas as características dos padrões de acesso feitos à mesma.

Utilizando ambos os exemplos apresentados para FH e FV da tabela Empregado, suponhamos que seja aplicada fragmentação híbrida seguindo as mesmas funções de fragmentação, tanto para a fragmentação horizontal como para a vertical. Neste ponto, poderíamos ter duas fragmentações resultantes para a tabela Empregado, a depender de qual fragmentação fosse aplicada em primeiro lugar:

- No caso da FH ser aplicada primeiramente, teremos a geração de quatro fragmentos:

  $FHíb_1 = p$ *(Cpf, Nome, Endereço)* $(Empregado_1 = F_{[salário > 5000]}(Empregado))$

  $FHíb_2 = p$ *(Cpf, Telefone, Celular, Cargo, Idade, Escolaridade, Salário)* $(Empregado_1 = F_{[salário > 5000]}(Empregado))$

  $FHíb_3 = p$ *(Cpf, Nome, Endereço)* $(Empregado_2 = F_{[salário <= 5000]}(Empregado))$

  $FHíb_4 = p$ *(Cpf, Telefone, Celular, Cargo, Idade, Escolaridade, Salário)* $(Empregado_2 = F_{[salário <= 5000]}(Empregado))$

- No caso da FV ser aplicada primeiramente, teremos a geração de três fragmentos:

  $FHíb_1 = p$ *(Cpf, Nome, Endereço)(Empregado)*

  $FHíb_2 = F_{[salário > 5000]}(p$ *(Cpf, Telefone, Celular, Cargo, Idade, Escolaridade, Salário) Empregado)*

  $FHíb_3 = F_{[salário <= 5000]}(p$ *(Cpf, Telefone, Celular, Cargo, Idade, Escolaridade, Salário) Empregado)*

Como pode ser observado, a segunda opção é mais vantajosa do ponto de vista da gerência dos fragmentos criados, pois cria uma menor quantidade de fragmentos, sem, no entanto, deixar de atender às características dos acessos feitos através das consultas .

Etapa de Alocação

Nesta segunda etapa do PDBD, o objetivo é definir os sites dentro da rede em que cada um dos fragmentos gerados na etapa anterior deva ser alocado. A alocação dos fragmentos pode se dar de duas maneiras:

- **Alocação Sem Replicação**, na qual cada fragmento é alocado em somente um site;
- **Alocação Com Replicação**, na qual cada fragmento é alocado em um ou mais sites. Neste caso, o projetista deve decidir o grau de replicação de cada fragmento.

Para que o projetista de banco de dados decida sobre o melhor esquema de alocação, é necessário que sejam analisados aspectos como: objetivos de desempenho, disponibilidade e confiabilidade do sistema; tipos e freqüências das transações disparadas por cada site; restrições de cada sítio quanto à capacidade de armazenamento e de processamento.

Aplicação e Suporte à Fragmentação

Existem hoje diversos SGBDs com suporte a distribuição de bases de dados, seja para alocação, seja para fragmentação de tabelas ou componentes de uma determinada base de dados. Entre estes SGBDs, podemos citar: Oracle, Non Stop SQL/Compaq Informix, SqlServer, entre outros.

Para exemplificar como a fragmentação de uma tabela pode ser implementada, definida previamente a sua função de fragmentação, foi escolhido o SGBD Oracle. Nesta seção ilustraremos como, na prática, uma tabela implementada em um SGBD Oracle pode ser fragmentada horizontalmente.

Fragmentação no Oracle

O SGBD Oracle oferece suporte à fragmentação horizontal de tabelas por três formas:

- ***List***: Controla explicitamente como acontecerá a fragmentação, determinando como cada registro é alocado a uma partição. Para isto, é necessário especificar uma lista prévia de valores que servirão como chave de fragmentação na descrição de cada partição. Um exemplo desta técnica suportada pelo Oracle é apresentada a baixo:

```
CREATE TABLE sales_by_region (
deptno           NUMBER(10),
deptname         VARCHAR2(20),
quarterly_sales  NUMBER(10,2),
state                  VARCHAR2(2))
PARTITION BY LIST (state) (
PARTITION q1_northwest VALUES ('OR', 'WA'),
PARTITION q1_southwest VALUES ('AZ', 'CA', 'NM'),
PARTITION q1_northeast VALUES ('NY', 'VT', 'NJ'),
PARTITION q1_southeast VALUES ('FL', 'GA'),
PARTITION q1_northcent VALUES ('MN', 'WI'),
PARTITION q1_southcent VALUES ('OK', 'TX'));
```

- ***Range:*** neste caso os registros são mapeados para as partições baseando-se nos intervalos de valores de cada partição da(s) chave(s) definida(s) para fragmentação. Um primeiro exemplo apresenta uso desta abordagem com chave simples, indicando quem tablespace a partição será alocada:

```
CREATE TABLE professional_history (
prof_history_id  NUMBER(10),
person_id        NUMBER(10) NOT NULL,
organization_id  NUMBER(10) NOT NULL,
record_date      DATE NOT NULL,
ph_comments      VARCHAR2(2000))
PARTITION BY RANGE (record_date) (
PARTITION yr0
VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY'))
TABLESPACE <tablespace_name>,
PARTITION yr1
VALUES LESS THAN (TO_DATE('01-JAN-2001','DD-MON-YYYY'))
TABLESPACE <tablespace_name>,
PARTITION yr2
VALUES LESS THAN (TO_DATE('01-JAN-2002','DD-MON-YYYY'))
TABLESPACE <tablespace_name>,
PARTITION yr9
VALUES LESS THAN (MAXVALUE)
TABLESPACE <tablespace_name>);
```

No entanto, a chave de fragmentação pode envolver mais de um campo da tabela:

```
CREATE TABLE sales    (
invoice_no NUMBER,
      sale_year  INT NOT NULL,
      sale_month INT NOT NULL,
      sale_day   INT NOT NULL )
PARTITION BY RANGE (sale_year, sale_month, sale_day)
    ( PARTITION sales_q1 VALUES LESS THAN (1997, 04, 01)
        TABLESPACE <tablespace_name>,
      PARTITION sales_q2 VALUES LESS THAN (1997, 07, 01)
        TABLESPACE <tablespace_name>,
      PARTITION sales_q3 VALUES LESS THAN (1997, 10, 01)
        TABLESPACE <tablespace_name>,
      PARTITION sales_q4 VALUES LESS THAN (1998, 01, 01)
        TABLESPACE <tablespace_name> );
```

- ***Hash:*** Esta última técnica suportada pelo SGBD Oracle é em geral utilizada quando não se encontram características de acesso que indiquem fragmentação por Range (faixa de valores) ou por List (lista de valores). Neste caso é utilizada uma função hash que faz o split dos registros entre as partições definidas no momento da fragmentação:

```
CREATE TABLE professional_history (
prof_history_id  NUMBER(10),
person_id        NUMBER(10) NOT NULL,
organization_id  NUMBER(10) NOT NULL,
record_date      DATE NOT NULL,
prof_hist_comments VARCHAR2(2000))
PARTITION BY HASH (prof_history_id)
PARTITIONS 3
STORE IN (tablespace_name1, tablespace_name2, tablespace_name3);
```

O Oracle ainda permite que estas técnicas sejam combinadas e aplicadas conjuntamente sobre uma tabela através de *Composite Partitioning*. O Oracle suporte dois tipos de combinações para fragmentação: Range-List e Range-Hash.

Conclusão

Como pode ser observado, distribuição de bases de dados é uma das técnicas mais importantes para otimização de desempenho em sistemas gerenciadores de bases de dados. Através desta técnica, é possível balancear as demandas de acesso aos dados, dividindo o processamento e armazenamento dos dados entre diferentes sites. Assim, as transações de acesso podem ser organizadas e direcionadas para os sites detentores das informações necessárias às consultas, carregando em memória somente os dados realmente prioritários. Numa próxima edição, iremos continuar com o assunto de distribuição de dados, abordando a área de fragmentação de índices e alocação de dados.

Referências

Baião, F., Mattoso, M., Zaverucha, G., 2003, "A Distribution Design Methodology for Object DBMS". Journal of Distributed and Parallel Databases, Kluwer Academic Publishers, Julho de 2003.

BROWNING, D., 2002, "SqlServer 2000 Resource Guide - RDBMS Performance TunningGuide".In:http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/sql/reskit/sql2000/part5/c2061.asp, Acessado em 09/2003.

CERI, S.,PELAGATTI, G., 1984, Distributed Databases - Principles and Systems, McGraw-Hill, Nova Iorque, EUA.

COMPAQ, 1999, "NonStop SQL/MP - Reliable, Parallel, Scalable Database Services". In: http://h71033.www7.hp.com/object/nssqlpd.html, Acessado em 10/2003.

DEWITT, D.,GRAY, J., 1992, "Parallel Database Systems: The Future of High Performance Database Processing", Communications of the ACM, v. 36, n. 6, pp. 85-98.

DOE, C., 2003, "IBM Informix Dynamic Server 9.4: Unequalled performance, scalability and availability". In: http://www-3.ibm.com/software/data/informix/pubs/whitepapers/ids94.pdf, Acessado em 10/2003.

GARCIA-MOLINA, H.,HSU, M., 1995, "Distributed Databases". In Kim, W., Modern Database Systems, ACM Press.

GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., 1999, Implementação de Sistemas de Bancos de Dados. Prentice-Hall.

IBM CORP., 2003, "Distributed Database Programming". In: http://as400bks.rochester.ibm.com/iseries/v5r2/ic2924/info/ddp/rbal1mst02.htm, Acessado em 10/2003.

KARLAPALEM, K., NAVATHE, S., MORSI, M., 1994, "Issues in Distribution design of Object-Oriented databases". In Özsu, M., Valduriez, P., and Dayal, P., Distributed Object Management, San Francisco, EUA, Morgan-Kaufmann.

NAVATHE, S.,ELMASRI, R., 2001, Fundamentals of Database Systems. 3ª Edição, Addison-Wesley, EUA.

ORACLE CORP., 2002, "Oracle9i Partitioning Enterprise Edition". In: http://otn.oracle.com/products/oracle9i/datasheets/partitioning.html, Acessado em 02/2003.

ÖZSU, M.,VALDURIEZ, P., 1999, Principles of Distributed Database Systems. 2ª Edição, New Jersey, EUA, Prentice-Hall.

Pablo Vieira Florentino é bacharel em Ciência da Computação pela UFBA (www.ufba.br) e Mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ (www.cos.ufrj.br), na área de banco de dados. Além disso, é Professor da UniGranrio e analista de sistemas, prestando serviços para organizações como a Embratel e a Marinha do Brasil na área de desenvolvimento de sistemas de telefonia.