1. CONCEITOS E NOÇÕES SOBRE SISTEMAS DE BANCOS DE DADOS

Continuação da Apostila 1...

1.8. GERENCIAMENTO DE TRANSAÇÕES

Uma transação é uma unidade de execução do programa que acessa e possivelmente atualiza vários itens de dados.

Normalmente, uma transação é iniciada por um programa do usuário escrito em uma linguagem de manipulação de dados ou linguagem de programação de alto nível; como por exemplo: SQL, C++, Java; em que é delimitada pelas instruções na forma de *Begin Transaction* e *End Transaction*. A transação consiste em todas as operações executadas entre o *Begin Transaction* e *End Transaction*.

Para garantir a integridade dos dados, é necessário que o sistema de banco de dados mantenha as propriedades **ACID** das transações.

1.8.1. Propriedades ACID

As transações executadas em sistemas gerenciadores de bancos de dados convencionais são estruturadas para cumprirem todas as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

São as propriedades ACID que possibilitam a avaliação da consistência lógica dos dados e das transações, portanto elas são de suma importância para que os dados cheguem ao banco de dados corretamente, garantindo assim a execução das transações de maneira completa, conforme ilustrado a seguir.

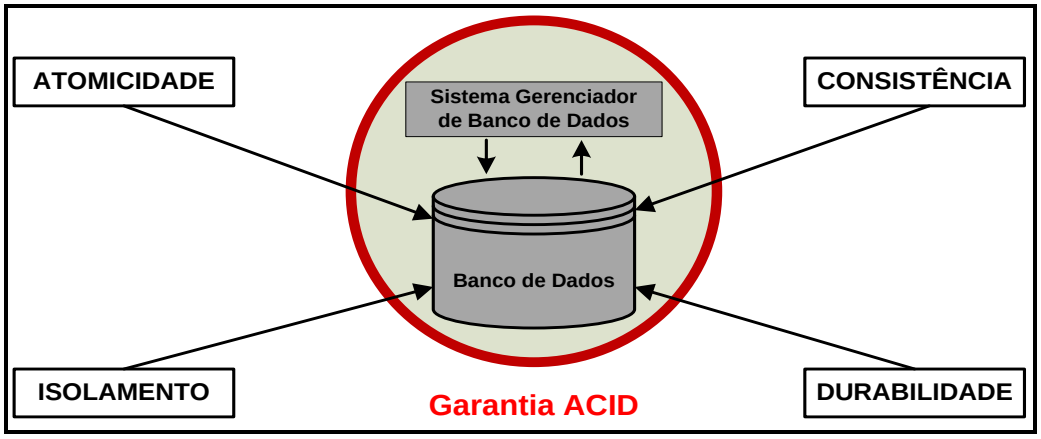

Figura 4. Garantia das Propriedades ACID (Sistemas Convencionais)

Atomicidade: Garante que uma transação é totalmente executada ou nenhum passo dela é executado, ou seja, não se pode dividir as operações que a compõe, ou todas as operações que compõe a transação são efetuadas por completo ou nenhuma será;

Consistência: garante que a execução de uma transação sempre transforma o estado consistente de um banco de dados em outro estado consistente;

Isolamento: garante que as ações de uma transação não são visíveis a nenhuma outra transação até que ela seja comprometida, implicando na não existência de interdependências na execução das transações;

Durabilidade: garante que as ações de uma transação no banco de dados são permanentes;

1.8.1. Gerenciamento de Transações Relatando as Propriedades ACID

As restrições temporais geralmente são colocadas nas regras de negócio da aplicação, em procedimentos armazenados, em gatilhos ou em métodos, condicionando as execuções das operações que envolvem uma transação.

Uma transação é composta por uma coleção de operações - que envolvem inserções (INSERT), alterações (UPDATE), exclusões (DELETE) e consultas (SELECT) - que são executadas e gerenciadas pelo sistema gerenciador de banco de dados (COMMIT, SAVEPOINT, ROLLBACK).

Exemplificando: para realizar uma atividade de transferência de fundos bancários (uma única unidade lógica de trabalho – uma transação), várias operações são executadas:

- Quando o valor sai de uma determinada conta, uma operação de registro de movimento deve ser realizada, bem como uma operação de subtração para a atualização do saldo;

```
INSERT INTO MOVIMENTO VALUES('00011111-1','13/11/2004','14:33:01',150.00,'D');
UPDATE CONTA
SET SALDO = SALDO – 150.00
WHERE NUM_CONTA =  '00011111-1';
```

- Quando o valor entra na conta de destino, uma operação de soma de ser realizada, atualização o saldo desta conta.

```
INSERT INTO MOVIMENTO VALUES('00022222-2','13/11/2004','14:33:04',150.00,'C');
UPDATE CONTA
SET SALDO = SALDO + 150.00
WHERE NUM_CONTA =  '00022222-2';
```

Esse exemplo simples demonstra que a atividade realizada por uma transação, pode ser composta por diversas operações diferentes.

Atomicidade: neste caso, ou são realizadas a duas operações, a de subtração e a de soma, ou nenhuma operação poderá ser realizada;

Consistência: neste caso, com a conclusão das duas operações, estaria garantido que a execução de uma transação sempre transforma o estado consistente de um banco de dados em outro estado consistente;

Isolamento: ocultaria a operação de subtração até que a operação de soma estivesse pronta;

Durabilidade: garante que tanto a operação de subtração quanto à de soma serão permanentes no banco de dados;

1.9. CONTROLE DE CONCORRÊNCIA

Uma das propriedades fundamentais de uma transação é o isolamento. Contudo, quando várias transações são executadas simultaneamente no banco de dados, a propriedade de isolamento pode não ser mais preservada.
Para garantir sua preservação, o sistema precisa controlar a interação entre as transações simultâneas; esse controle é alcançado por meio de uma série de mecanismos chamados esquemas de **Controle de Concorrência** (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 427).

Um modo de garantir a seriação é exigir que os itens de dados sejam acessados de uma maneira mutuamente exclusiva; ou seja, enquanto uma transação está acessando um item de dados, nenhuma outra transação pode modificar esse item de dados, como por exemplo, enquanto estiver ocorrendo uma alteração (UPDATE) em um registro (linha), nenhuma outra operação pode ser realizada neste registro.

O método mais comum usado para implementar esse requisito é permitir que uma transação acesse um item de dados somente se estiver atualmente mantendo um bloqueio sobre esse item.

1.9.1. Bloqueios

Existem vários modos como um item de dados pode ser bloqueado. Em particular dois deles serão apresentados a seguir (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 427-430).

Compartilhado: Se uma transação T_i tiver obtido um bloqueio no modo compartilhado sobre o item Q, então T_i pode ler, mas não pode escrever Q.

Exclusivo: Se uma transação T_i tiver obtido um bloqueio no modo exclusivo sobre um item Q, então T_i pode ler e escrever Q.

É necessário que cada transação solicite um bloqueio em um modo apropriado sobre o item de dados Q, dependendo dos tipos de operações que realizará sobre Q. A transação faz a solicitação ao gerenciador de controle de concorrência. A transação só pode prosseguir com a operação depois que o gerenciamento de controle de concorrência conceder o bloqueio à transação.

Como ilustração, considere que A e B sejam duas contas que são acessadas pelas transações T_1 e T_2. A transação T_1 transfere \$50 da conta B para a conta A. A transação T_2 apresenta a quantia total de dinheiro nas contas A e B, ou seja, a soma $A + B$.

T_1 : Lock(B); : read(B); : $B := B - 50$; : write(B); : unlock(B); : lock(A); : read(A); : $A := A + 50$; : write(A); : unlock(A).	T_2 : Lock(A); : read(A); : unlock(A); : lock(B); : read(B); : unlock(B); : display($A + B$).

Suponha que os valores das contas A e B sejam \$100 e \$200, respectivamente. Se essas duas transações forem executadas em série, ou na ordem T_1, T_2, então a T_2 mostrará o valor \$300. Agora se as transações forem executadas simultaneamente, então o valor mostrado pela transação T_2 poderá ser \$250. Ou seja, subtrai \$50 da conta B e mostra a soma antes de ocorrer o crédito na conta A.

Uma forma para resolver esse problema, é adiar o desbloqueio para o final da transação. A transação T_3 corresponde a T_1 com desbloqueio adiado. A transação T_4 corresponde a T_2 com desbloqueio adiado.

T_3 : Lock(B); : read(B); : $B := B - 50$; : write(B); : lock(A); : read(A); : $A := A + 50$;	T_4 : Lock(A); : read(A); : lock(B); : read(B); : display($A + B$). : **unlock(A);** : unlock(B).

: write(*A*); : unlock(*B*); : **unlock(*A*).**	

As operações de exclusão (DELETE), de inclusão (INSERT) e de alteração (UPDATE) devem ser realizadas com o uso de Bloqueios Exclusivos sobre o item de dados (tupla; linha; registro) a ser excluído, incluído ou alterado.

1.10. SISTEMA DE RECUPERAÇÃO

Um sistema de computador, como qualquer outro dispositivo, está sujeito a falhas por uma série de causas: falha de disco, falta de energia, erro de software, um incêndio na sala e até mesmo sabotagem. Em qualquer falha informações podem ser perdidas. Portanto, o Sistema de Banco de Dados precisa tomar ações de antemão para garantir que as propriedades de atomicidade e durabilidade das transações sejam preservadas. Uma parte integral de um Sistema de Banco de Dados é um **esquema de recuperação** que pode restaurar o Banco de Dados ao estado coerente que existia antes da falha, bem como oferecer **alta disponibilidade**, ou seja, precisa reduzir o tempo durante o qual o Banco de Dados não pode ser usado após uma falha. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 459).

1.10.1. Classificação das Falhas

Existem vários tipos de falha que podem ocorrer em um sistema, cada um deles precisa ser tratado de uma maneira diferente. O tipo mais simples de falha é aquele que não resulta na perda de informações no sistema. As falhas que são mais difíceis de tratar são aquelas que resultam em perda de informações.

Falha de transação: existem dois tipos de erros que podem causar a falha de uma transação:

> **Erro lógico**: transação não pode mais continuar com sua execução normal devido a alguma condição interna, como entrada defeituosa, dados não encontrados, estouro ou limite de recursos ultrapassado.
>
> **Erro do sistema**: o sistema entrou em um estado indesejável (por exemplo, impasse), fazendo com que uma transação não possa continuar com sua execução normal. Contudo, a transação não pode ser reexecutada em outro momento.

Falha do sistema: existe um defeito do hardware, ou um *bug* no software de Banco de Dados ou no Sistema Operacional, causando a perda do conteúdo

do armazenamento volátil e encerrando o processamento da transação. O conteúdo do armazenamento não volátil permanece intacto e não é adulterado.

Falha do disco: um bloco de disco perde seu conteúdo como resultado de uma falha da cabeça ou uma falha durante uma operação de transferência de dados. As cópias dos dados em outros discos, ou backups de arquivamento em mídia terciária, como fitas, são usadas para a recuperação de falhas.

1.10.2. Meios de Armazenamento Físico

Armazenamento Volátil (principal): caracteriza-se por acesso rápido, mas limitada capacidade de armazenamento. Esta categoria inclui:

Memória Cache: é um meio de armazenamento de alta performance. É a forma de armazenamento mais rápida e mais cara. A memória Cache é pequena e o seu uso é gerenciado pelo Sistema Operacional para aumentar a velocidade de processamento.

Memória Principal: é o meio de armazenamento usado para os dados que estão disponíveis para serem diretamente processados. Geralmente os Bancos Dados são muito grandes para serem armazenados inteiramente na memória principal. O conteúdo da memória principal é geralmente perdido se ocorrer uma falha de energia elétrica ou uma queda do sistema.

Armazenamento Não Volátil (secundário): esse tipo de armazenamento, em geral apresenta maior capacidade, menor custo e maior tempo de acesso quando comparado com o armazenamento nas memórias. Não permite o processamento direto dos dados; os dados devem ser copiados para a memória antes de serem processados e depois regravados. Nessa categoria estão incluídos:

Fita Magnética (armazenamento de acesso seqüencial): em geral é utilizado para *backup* de grandes massas de dados. A desvantagem em relação ao disco é a velocidade de acesso, a vantagem é o custo muito baixo. Para se localizar um item de dados na fita, ela deve ser lida seqüencialmente desde o início.

Disco Magnético (armazenamento de acesso direto): são de importância vital para um Banco de Dados, já que normalmente são utilizados para armazenar grandes quantidades de dados. Para serem processados, os dados devem ser transferidos do disco para a memória principal, e as alterações precisam ser gravadas novamente. Esse tipo de armazenamento é dito de **acesso direto** porque permite obter dados em qualquer posição do disco. O armazenamento em disco geralmente sobrevive a quedas do sistema, mas não garante a integridade das

informações em falhas de cabeça. As informações são armazenadas na superfície do disco em círculos concêntricos, cada um com um diâmetro diferente. Esses círculos são chamados de trilhas. Cada trilha armazena a mesma quantidade de dados, portanto, os bits estarão mais compactados nas trilhas de menor diâmetro. Os dados são transferidos entre o disco e a memória principal em blocos. Uma trilha é dividida em um conjunto de blocos iguais pelo sistema operacional durante a formatação. O endereço de um bloco é uma combinação do número da face do disco (1 ou 2), número da trilha, e número do bloco dentro da trilha.

1.10.3. Acesso aos Dados

O Sistema de Banco de Dados reside permanentemente no armazenamento não volátil (normalmente, discos) e é particionado em unidades de armazenamento de tamanho fixo, chamadas **blocos**.

Os blocos são as unidades de transferência de dados de e para o disco, e podem conter vários itens de dados (tuplas, registros).

As transações incluem informações do disco na memória principal e depois enviam a informação para o disco. As operações de entrada e saída são feitas em unidades de blocos. Os blocos residindo no disco são chamados de **blocos físicos**, os blocos residindo temporariamente na memória principal são conhecidos como **blocos de buffer**. A área da memória em que os blocos residem temporariamente é chamada de **buffer de disco**.

Os movimentos de bloco entre o disco e a memória principal são iniciados por meio das duas operações a seguir:

input(*B*) transfere o bloco físico *B* para a memória principal.
output(*B*) transfere o bloco de buffer *B* para o disco e substitui o bloco físico apropriado.

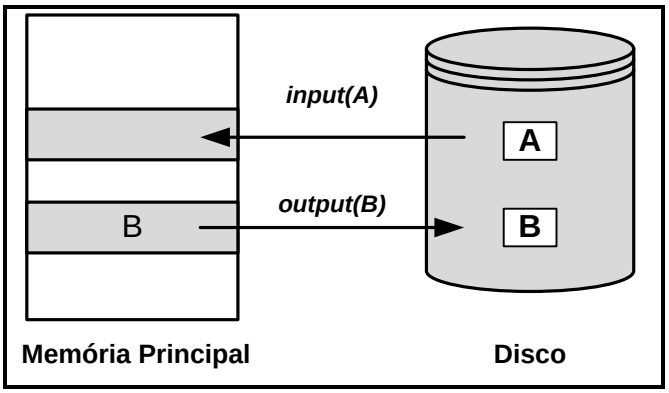

Figura 6. Output e Input de Itens de Dados do Disco para Memória

A transação T_i possui uma área de trabalho privada em que são mantidas as cópias de todos os itens de dados acessados e atualizados por T_i. O sistema cria essa área de trabalho quando a transação é iniciada; o sistema a remove quando a transação é confirmada ou abortada. Cada item de dados *X* mantido na área de trabalho da transação T_i é indicado por *xi*. A transação T_i interage com o Sistema de Banco de Dados transferindo dados de e para sua área de trabalho no buffer do sistema. As transferências são realizadas por essas duas operações:

read(*X*) atribui o valor do item de dados *X* à variável local x_i. Ela executa essa operação da seguinte maneira:

a) Se o bloco B_x em que *X* reside não estiver na memória principal, emite *input*(*B*).
b) Atribui a x_i o valor de *X* vindo do bloco de buffer.

write(*X*) atribui o valor da variável local xi ao item de dados *X* no bloco de buffer. Ela executa essa operação da seguinte maneira:

a) Se o bloco B_x em que *X* reside não estiver na memória principal, ela emite *input*(*B*).
b) Atribui o valor x_i a *X* no buffer B_x.

Observe que as duas operações podem exigir a transferência de um bloco do disco para a memória principal. Porém, elas não exigem especificamente a transferência de um bloco da memória principal para o disco.

Um bloco de buffer, por fim, é gravado em disco porque o gerenciador de buffer precisa do espaço da memória para outras finalidades ou porque o Sistema de Banco de Dados deseja refletir a mudança feita em *B* no disco.

1.10.4. Recuperação Baseada em *LOG*

A estrutura mais utilizada para registrar as modificações do Banco de Dados é o *log*. O *log* é uma seqüência de registros de *log*, registrando todas as atividades de atualização no Banco de Dados. Existem vários tipos de registros de *log*. Um registro de *log* de atualização descreve uma única escrita no Banco de Dados. Ele tem estes campos:

O **identificador de transação** é o identificador exclusivo da transação que realizou a operação *write*.
O **identificador de item de dados** é o identificador exclusivo do item de dados escrito. Normalmente, esse é o local no disco do item de dados.
O **valor antigo** é o valor do item de dados antes da escrita. Exemplo: o parâmetro ":OLD" na linguagem PL/SQL em Triggers.

O **novo valor** é o valor que o item de dados terá após a escrita. Exemplo: o parâmetro ":NEW" na linguagem PL/SQL em Triggers.

Sempre que uma transação realiza uma escrita, é essencial que o registro de *log* para essa escrita seja criado antes que o Banco de Dados seja modificado. Quando existe um registro de *log*, pode-se emitir a modificação no Banco de Dados se isso for desejado. Além disso, tem-se a capacidade de desfazer uma modificação que já foi enviada ao Banco de Dados. Isso é feito usando o campo do valor antigo nos registros de *log*.

Para os registros de *log* serem úteis à recuperação do sistema e a falha de disco, o *log* precisa residir no armazenamento estável. O *log* deve conter um registro completo de toda a atividade do Banco de Dados. Como resultado, o valor de dados armazenados no *log* pode se tornar incrivelmente grande. A varredura do *log* é realizada de trás para frente, ou seja, do mais recente ao menos recente.

Ilustração: Suponha que essas transações são executadas em série, na ordem T_0 seguida por T_1, e que os valores das contas A, B e C antes que a execução ocorra fossem \$1.000, \$2.000 e \$700, respectivamente. Antes porém, é necessário indicar os vários tipos de registros de *log*, tais como:

- $<T_i$ start>. A transação foi iniciada.
- $<T_i, X_j, V_1, V_2>$. A transação realizou uma escrita sobre o item de dados X_j. X_j tinha valores V_1 antes da escrita e terá o valor V_2 após a escrita.
- $<T_i$ commit>. A transação foi confirmada.
- $<T_i$ abort>. A transação foi abortada (*rollback*).

TRANSAÇÕES	*LOG*	BANCO DE DADOS
T_0 : read(A); : $A := A - 50$; : write(A); : read(B); : $B := B + 50$; : write(B); T_1 : read(C); : $C := C - 100$; : write(C);	$<T_0$ start> $<T_0, A, 1000, 950>$ $<T_0, B, 2000, 2050>$ $<T_0$, commit> $<T_1$, start> $<T_1, C, 700, 600>$ $<T_1$, commit>	 A = 950 B = 2050 C= 600

1.11. PROCESSAMENTO DA CONSULTA

O **processamento da consulta** refere-se ao conjunto de atividades envolvidas na extração de dados de um Banco de Dados. As atividades incluem tradução de consultas em linguagem de banco de dados de alto nível para expressões que podem ser usadas no nível físico do sistema de arquivos, uma série de transformações de otimização da consulta e a avaliação real das consultas (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 357).

1.11.1. Visão Geral

As etapas envolvidas no processamento de uma consulta são: (a) Análise e tradução; (b) Otimização; (c) Avaliação, conforme ilustrado na próxima figura.

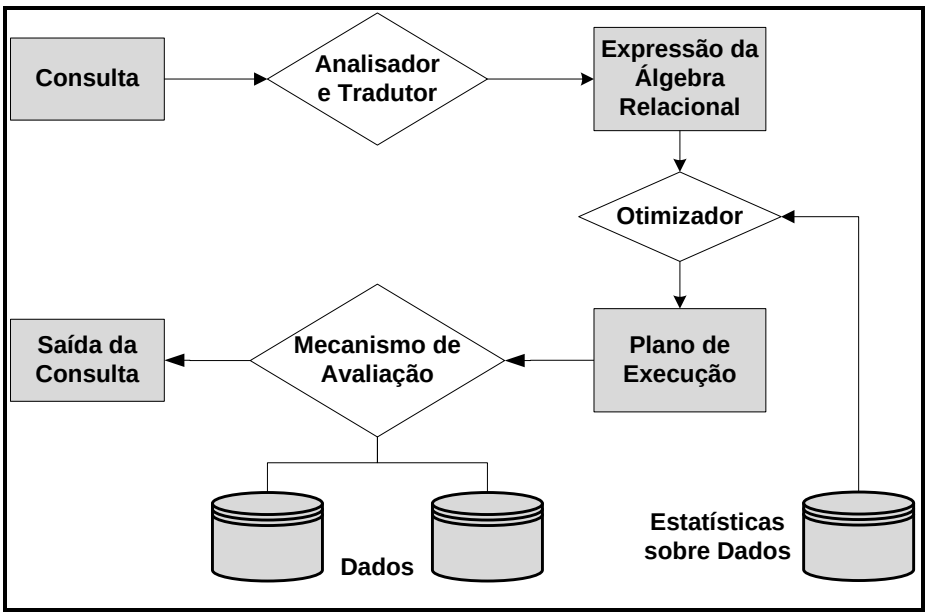

Figura 7. Etapas do Processamento de uma Consulta

Antes que qualquer processamento de consulta possa começar, o sistema precisa traduzir a consulta para uma forma utilizável. Uma linguagem como SQL é adequada para o uso humano, mas não para a representação interna do sistema de uma consulta. Uma representação interna mais útil é aquela baseada na álgebra relacional estendida.

Assim, a primeira ação que o sistema precisa tomar no processamento da consulta é traduzir determinada consulta para a sua forma interna. Esse processo de tradução é semelhante ao trabalho realizado pelo analisador (*parser*) de um compilador. Ao gerar a forma interna da consulta, o analisador verifica a sintaxe da consulta do usuário e se os nomes de relação (tabela) que aparecem na consulta são nomes de relações no Banco de Dados. O sistema

constrói uma representação de árvore de análise da consulta, que pode traduzir para uma expressão da álgebra relacional. Se a consulta foi expressa em termos de uma *view* (visão), a fase de tradução também substitui todos os usos da *view* pela expressão da álgebra relacional que define a *view*. A maioria dos textos sobre compilador explica a análise.

Dada uma consulta, geralmente existem vários métodos para calcular a resposta, conforme ilustrado a seguir:

LINGUAGEM SQL	ÁLGEBRA RELACIONAL
SELECT *saldo* FROM *conta* WHERE *saldo* < *3000*	$\sigma_{saldo<3000}(\Pi_{saldo}\ (conta))$ ou $\Pi_{saldo}(\sigma_{saldo<3000}\ (conta))$

Além do mais, é possível executar cada operação da álgebra relacional por um de vários algoritmos diferentes. Por exemplo, para implementar a seleção anterior, pode ser feita uma pesquisa em cada tupla em *conta* para encontrar tuplas com *saldo* menor que 3000. Se um índice de árvore estiver disponível sobre o atributo *saldo*, é possível usá-lo em vez de localizar as tuplas.

1.11.2. Álgebra Relacional

A **álgebra relacional** consiste numa linguagem de consulta “pura”, sem “maquiagem sintática”, e serve para ilustrar as técnicas fundamentais de extração de dados do banco de dados.

A álgebra relacional consiste numa coleção de operações usadas para manipular relações inteiras e são usadas para selecionar tuplas de uma relação individual e combinar com tuplas relacionadas de outras relações, com o propósito de especificar uma consulta (query/SQL). São seis essas operações fundamentais:

Operação SELEÇÃO

É uma operação unária (opera sobre uma relação apenas). Essa seleciona tuplas que satisfazem um dado predicado. Utiliza-se a letra grega *sigma* para denotar a operação de seleção. A expressão de seleção é dada por:

σPREDICADO(RELAÇÃO ARGUMENTO)

Exemplo:
Considerando a existência da seguinte relação no banco de dados da Indústria de Sorvete:

Relação (Tabela) PEDIDO

COD-CLI	COD-PROD	QTDE	DATA
8392	981	10	02/01/1992
8392	324	70	02/01/1992
4132	324	57	10/02/1992
1728	981	69	15/03/1992
1728	430	110	20/01/1992
8392	430	32	20/01/1992
1728	981	10	20/01/1992

Assim, para selecionar tuplas da relação PEDIDO onde o cliente é 1728 escreve-se:

$$\sigma_{COD\text{-}CLI=1728}\ (PEDIDO)$$

Então a relação resultante da consulta será:

COD-CLI	COD-PROD	QTDE	DATA
1728	981	69	15/03/1992
1728	430	110	20/01/1992
1728	981	10	20/01/1992

No predicado de seleção são permitidas comparações usando =, >, <, <=, =>, <>. Além disso, vários predicados podem ser combinados num predicado maior usando-se os conectivos “e” e “ou”.

Exemplo:
Para encontrar as tuplas pertencentes aos pedidos do cliente 1728, com quantidades maiores que 50, escreve-se:

$$\sigma_{COD\text{-}CLI=1728\ e\ QTDE>50}\ (PEDIDO)$$

Então a relação resultante da consulta será:

COD-CLI	COD-PROD	QTDE	DATA
1728	981	69	15/03/1992
1728	430	110	20/01/1992

Operação PROJEÇÃO

A operação projeção é uma relação unária que copia sua relação argumento, deixando algumas colunas (atributos) de lado. A projeção é denotada pela letra grega *pi* e é denotada pela expressão:

$$\pi \text{ LISTA-DE-ATRIBUTOS (RELAÇÃO ARGUMENTO)}$$

Exemplos:
a) Listar o código e o nome de todos os clientes

$$\pi \text{ COD-CLI,NOME-CLI (CLIENTE)}$$

b) Listar o código e o nome de todos os clientes que têm limite de crédito maior que 2000.

b.1) uma operação de seleção dos clientes com limite de crédito>2000

$$\sigma \text{ LIM\_CRED} > 2000 \text{ (CLIENTE)}$$

b.2) uma projeção sobre a relação resultante para restringir o resultado às colunas desejadas.

$$\pi \text{ COD-CLI, NOME-CLI}(\sigma \square \text{LIMITE-CRED} > 2000 \text{ (CLIENTE)})$$

Operação PRODUTO CARTESIANO

É uma operação binária. Permite combinar informações de relações diferentes. A relação resultante do produto cartesiano de várias relações possuirá todos os atributos das relações participantes. Utiliza-se um *x* para denotar o produto cartesiano.

Exemplo:
Listar todos os viajantes e o nome de todos os seus clientes. Para isso é preciso a seguinte expressão:

$$R = \sigma \text{ VIAJANTE.COD\_VIAJ=CLIENTE.COD\_VIAJ (VIAJANTExCLIENTE)}$$

Operação RENOMEAR

É uma operação usada para possibilitar o uso de uma mesma relação dentro de uma única consulta (para eliminar ambigüidade). Exemplo de empregado e gerente. Letra ρ.

Operação UNIÃO

É uma operação usada para unir tuplas de mais de uma relação, que de alguma certa forma se relacionem. Exemplo de empréstimo e depósito que um cliente realizou em uma agência bancária. Letra $\cup$. Exemplo $\pi \cup \pi$.

Operação DIFERENÇA DE CONJUNTO

É uma operação usada para encontrar tuplas que estão em uma relação e não em outra. Exemplo de depósito e empréstimo que um cliente possui em uma agência bancária. Letra $-$. Exemplo $\pi - \pi$.