Unidade 3

UMA INTRODUÇÃO A BANCO DE DADOS RELACIONAL

Nessa unidade você conhecerá os fundamentos dos Gerenciadores de Bancos de Dados Relacionais, e dentre eles a Álgebra Relacional que dá suporte ao processo de manipulação dos dados. Conhecerá o modelo estrutural proposto para o tratamento de dados e os tipos de integridade a serem garantidas. Conhecerá o Catálogo do Banco de Dados seus benefícios e que tipo de informação podemos encontrar nele. Conhecerá o mecanismo das Visões, suas potencialidades, limitações e principais características. Terá também uma breve visão sobre a linguagem SQL e por fim conhecerá o processo de indexação, suas vantagens e desvantagens.

UNIDADE 3 – UMA INTRODUÇÃO A BANCO DE DADOS RELACIONAL

3.1 CONCEITOS FUNDAMENTAIS DE BANCO DE DADOS RELACIONAL

Um breve histórico

O primeiro protótipo de SGBD Relacional a ser desenvolvido foi o *System/R* da IBM, por volta de 1976. Ao final do projeto a IBM já possuía o protótipo do Banco de Dados Relacional e a Linguagem de consulta estruturada de fácil acesso denominada ainda *SEQUEL*. No entanto, o que se sabe é que esse produto nunca foi lançado. Por outro lado, um grupo de engenheiros que havia participado do projeto do *System/R* fundou uma nova empresa denominada *Relational Software* e lançou o primeiro produto comercial, baseado também na linguagem de consulta estruturada, denominado *ORACLE*. Na década de 80 a IBM lançou o seu primeiro produto relacional comercial *DB2* (*Database2*) e o SQL/DS (*SQL Data*) (MANZANO, 2002, p.17).

O modelo de dados relacional surgiu na década de 70 e até hoje é exaustivamente utilizado. Foi fruto do trabalho de E. F. Codd, matemático, pesquisador da IBM que percebeu certa ligação entre teorias da matemática e o armazenamento de dados. Foi justamente a partir dos seus conhecimentos da teoria de conjuntos e do conceito de relação que ele projetou o modelo para banco de dados relacional. Nesse modelo, o principal elemento é a tabela (relação) que possui uma estrutura bidimensional com uma dimensão vertical por colunas (atributos) e uma horizontal por linhas (tuplas) (HARRINGTON, 2002, p.66)

Em meados de 1985, Codd publicou um artigo onde definiu **12 regras** que os produtos de SGBD deveriam atender para se enquadrarem dentro da categoria de Relacional. Essas 12 regras estão descritas abaixo (HARRINGTON, 2002, p.115 até p. 126):

Regra 1: As regras para informações → "Todas as informações em um banco de dados relacional são representadas explicitamente no nível lógico exatamente da mesma maneira – por valores em tabelas". Tal regra garante que a únicas estruturas de dados utilizadas em banco de dados relacionais sejam **tabelas bidimensionais**. Tais estruturas servirão para armazenar todo tipo de objeto (entidades, relacionamentos) necessário.

Regra 2: A regra do acesso garantido → "Cada e todos os dados (valor atômico) em um banco de dados relacional tem a garantia de serem logicamente acessíveis pela reclassificação de uma combinação de nomes de tabelas, valor de chave primária e nome de coluna". Essa regra garante que conhecendo somente o nome da tabela, o nome da coluna e a chave primaria da linha você conseguirá encontrar um dado específico. Não há nenhuma regra que garanta que uma tabela deva ter uma chave primária, tanto é que podemos criar tabelas e populá-las sem definir uma chave primária. No entanto, tabelas com essa característica estarão em desacordo com a regra do acesso garantido.

Regra 3: Tratamento sistemático de valores nulos → "Valores nulos (distintos da string de caractere vazia ou de uma string de caracteres em branco ou de qualquer outro numero) são suportados completamente em um SGBD relacional para representar de uma maneira sistemática as informações ausentes, independentemente do tipo de dados". Valores nulos podem trazer problemas especialmente nos processos de recuperação de dados (Leitura). Se você realiza uma consulta, colocando como clausula de restrição uma coluna que possua alguma linha com valor nulo, essa linha poderá não aparecer no resultado a menos que você especifique um critério adicional de restrição para valor nulo. Imagine uma tabela de PRODUTOS que possua a coluna PREÇO como uma coluna passível de ser nula e exista varias linhas na tabela cujo preço esteja nulo. Se você fizer uma consulta selecionando todos os produtos que custem menos que 10,00, em muitos SGBDs, as linhas de valores nulos não aparecerão. Se você mudar a consulta para então listar os produtos que custem mais que 10,00, novamente os produtos com preço nulo também não aparecerão. Observe que as linhas cuja coluna está com valor nulo ficarão em um "limbo", acessíveis somente quando tratado a presença do valor nulo.

Regra 4: Catálogo on-line dinâmico baseado no modelo relacional → "A descrição de banco de dados é representada no nível lógico da mesma maneira que dados simples, de modo que usuários autorizados possam aplicar a mesma linguagem relacional que aplicam a dados regulares a sua interrogação". A

estrutura de dados utilizada para implementar o Dicionário de Dados (Catálogo) de um Banco Relacional deve ser a mesma utilizada para implementar a tabela de dados. Significa que a forma (estrutura) como nós implementamos nossos dados dentro de um Banco Relacional é a mesma que o fabricante do SGBD usou para implementar as estruturas de dados utilizadas pelo SGBD para realizar o seu gerenciamento. Os tipos de tabelas utilizadas por cada fabricante de Banco Relacional para implementar o Dicionário de Dados variam bastante, mas é garantido que todos os fabricantes implementam o seu Dicionário de Dados estruturando-os em forma de tabelas.

Regra 5: A regra da sublinguagem de dados abrangentes → Um sistema relacional pode suportar varias linguagens e vários modos de utilização de terminal (por exemplo, modo de preenchimento de espaços em branco). Entretanto, há pelo menos uma linguagem cujas instruções podem ser descritas por alguma sintaxe bem definida, como *strings* de caracteres, e que seja abrangente para suportar todos os seguintes itens:

. Definição de Dados
. Definição de Visualização
. Manipulação de Dados (interativa e pelo programa)
. Restrições de Integridade
. Limites de Transação (começo, realização e reversão)

A partir da versão 92 da linguagem SQL todas estas regras passaram a ser suportadas. Atualmente a maioria dos SGBDs Relacionais utilizam SQL como a principal linguagem de manipulação de dados. Tem-se dessa forma, garantia de cumprimento as restrições dessa regra.

Regra 6: A regra da visualização da atualização → "Todas as visualizações que teoricamente são atualizáveis são também atualizáveis pelo sistema". Essa regra busca garantir basicamente que, toda visão atualizável, ou seja, que cumpra as regras de atualização (ver conceito sobre visões no capitulo 3, seção 3.3) quando atualizada deve propagar tais atualizações também para as tabelas de origem dos dados.

Regra 7: Inserção de alto nível, atualização e exclusão → "A capacidade de tratar uma relação básica ou uma relação derivada como um único operando é aplicada não apenas à recuperação de dados, mas também a inserção, atualização e exclusão de dados". Através dessa regra Codd quis assegurar que a linguagem de manipulação de dados de dados possa inserir, atualizar e excluir mais de uma linha com um único comando. Tal potencialidade é fornecida pela linguagem SQL para os SGBDs Relacionais simplifica a lógica de programação, pois não é necessário ler linha a linha para localizar as linhas que serão modificadas. Você poderá especificar em um único comando os critérios de restrição sobre os dados lidos para sim aplicar a modificação sobre os dados que atenderem a tais critérios.

Regra 8: Independência de dados física → "Atividades de programas de aplicações e atividades de terminais permanecem logicamente intactas sempre que alguma alteração ocorrer na representação de armazenamento ou nos métodos de acesso". Com isso, você poderá mover o banco de dados de um dispositivo físico para outro, alterar o layout físico sem causar nenhum impacto na maneira como os programas aplicativos ou os usuários finais interagem com o banco de dados.

Regra 9: Independência de dados lógica → "Programas de aplicações e atividades de terminais permanecem logicamente intactos quando são feitas alterações de qualquer tipo às tabelas básicas que preservam informações que teoricamente permitem a estabilidade dos dados". Modificações aplicadas no nível conceitual da arquitetura do Sistema de Banco de Dados, como por exemplo, incluindo uma nova tabela, incluindo uma nova coluna em uma tabela, ou até mesmo excluindo uma tabela do esquema não deve ter conseqüências para as outras partes do esquema. É claro que, quando excluímos uma tabela do esquema, consequentemente estamos excluindo todos os dados atrelados a esta estrutura. Certamente nesse caso, a preservação da informação não está sendo mantida como recomenda a regra.

Regra 10: Independência de Integridade → "Restrições de integridade específicas para um banco de dados relacional em particular devem ser definíveis na sublinguagem de dados relacional e passíveis de serem armazenadas no catálogo, não nos programas de aplicação. No mínimo duas das seguintes restrições de integridade devem ser suportadas: 1. integridade de entidade: nenhum componente de uma chave primária tem permissão para ter um valor nulo. 2. integridade referencial: para cada valor de chave estrangeira não-nulo e distinto em um banco de dados relacional, deve existir um valor de chave primária que corresponda ao mesmo domínio".

Regra 11: Independência de Distribuição → "Um SGBD relacional tem independência de distribuição". Em um banco de dados distribuído os dados são armazenados em mais que um computador. No entanto, para o usuário final, a distribuição dos dados não deve afetar a visão do usuário que deve trabalhar como os dados como se estes estivessem centralizados. O usuário não precisa conhecer o local de armazenamento dos dados.

Regra 12: Regra de não-subversão → "Se um sistema relacional tiver uma linguagem de baixo nível (um único registro de cada vez), essa linguagem de baixo nível não pode ser utilizada para subverter ou pular as regras de integridade ou restrições expressas na linguagem relacional de nível mais alto (múltiplos registros de cada vez) ". Essa regra busca garantir que qualquer linguagem que busque acessar os dados, deve respeitar todas as Restrições de Integridade que estão armazenadas no Dicionário de Dados.

Esquemas e Instância

Antes de avançarmos no universo do Banco de Dados Relacional cabe distinguir dois elementos importantes: as noções de **esquema** e **instância** do banco de dados (SILBERSCHATZ, KORTH, SUDARSHAN, 2006, p.27, DATE, 2003, p.57):

O **Esquema** de um Banco de Dados equivale ao projeto lógico onde se define todas as relações (tabelas) que estarão compondo a estrutura dos dados. Um Esquema de Banco de Dados é composto de um conjunto de esquemas de relações. Podemos ver o Esquema como um cabeçalho que define os elementos que comporão a estrutura da relação bem como os tipos de dados que serão suportados por cada elemento da estrutura. Em resumo, o Esquema refere-se ao layout de uma tabela, composto pelo nome da tabela, o nome de cada coluna, o tipo de dado associado a cada coluna e as restrições de integridade.

A **Instância** é o próprio banco de dados, isto é, trata-se do conjunto de valores que povoam a relação, respeitando-se é claro os requisitos estruturais definidos no esquema. É o corpo que acompanha o cabeçalho representando o esquema. A Instância refere-se aos valores das colunas e linhas de determinado esquema.

Podemos também dizer que os Esquemas das relações são mais constantes ao longo do tempo enquanto suas Instâncias estão em constante movimento, normalmente crescendo, algumas vezes, diminuindo e muitas vezes se modificando. Mudanças de Esquema podem acontecer, é claro, como o acréscimo de um novo atributo/coluna. Mas certamente não são tão freqüentes. Além disso, considerando-se que um Esquema de relação, tenha sido elaborado a partir de um processo de software, cujo levantamento de requisitos e análise do ambiente tenha sido elaborado com o rigor necessário, não soa muito bem que, após a implementação física de tal projeto, ocorram mudanças frequentes nos Esquemas das tabelas.

3.2 OS TRÊS ASPECTOS DO BANCO DE DADOS RELACIONAL

Os Sistemas Relacionais são construídos tomando por base o Modelo Relacional de Dados. Tal modelo pressupõe três aspectos básicos (DATE, 2003):

3.2.1 Aspecto Estrutural**:** A estrutura de dados do Modelo Relacional é a **Relação** ou, informalmente, a **Tabela**. Os dados no Banco de Dados Relacional são vistos como uma coleção de Relações, ou seja, uma coleção de Tabelas. Alem disso, não existem ponteiros para expressar a ligação entre as linhas, pois todos os relacionamentos também são representados por dados armazenados em tabelas, respeitando a Regra numero 1 definida por Codd. Cada tabela será identificada por um nome único. Conforme dito na introdução desse capítulo, uma tabela é uma visão bidimensional dos dados, formada por linhas e colunas. As linhas expressam um individuo dentro do conjunto mapeado pela tabela. As colunas representam as características dos objetos ou dos indivíduos pertencentes a uma tabela. Tais características são os elementos que darão forma e distinguirão os indivíduos dentro do conjunto. Quando se está modelando uma tabela é importante conhecer o tipo de dado que será armazenado em cada coluna. Em termos formais, o universo dos possíveis valores que uma coluna poderá assumir em um determinado instantâneo de tempo é denominado **Domínio** (DATE, 2003).

Domínio

Um **Domínio** é um conjunto de valores atômicos, isto é, cada valor no domínio é indivisível. Esta é uma imposição do Modelo Relacional Normalizado, no qual não se podem especificar atributos compostos ou multivalorados; esta proibição caracteriza a Primeira Forma Normal (1FN), que faz parte da própria definição do Modelo. O projeto de Banco de Dados Relacional decorrente da primeira forma normal é chamado de **Modelo Relacional Plano,** pois impede a possibilidade de uma nova dimensão imposta pelo armazenamento de atributos multivalorados e compostos.

Um **Domínio** é um conjunto de valores atômicos todos do mesmo tipo a partir dos quais os valores reais aparecem especificados como atributos. Em outras palavras, Domínio refere-se ao conjunto dos possíveis valores que um atributo pode assumir.

Os Domínios têm certa importância operacional, que é a seguinte: se dois atributos retiram seus valores do mesmo domínio, então as comparações e as junções, uniões que envolvem tais pares de atributos provavelmente farão sentido, uma vez que há comparação de igual para igual.

Embora a bibliografia utilizada não possua esta definição, mas e possível fazer uma conceituação de domínio sem com isso comprometer o entendimento e o rigor conceitual adotado anteriormente. Os domínios podem ser classificados em dois tipos: Discretos e Amplos (contínuos).

Os domínios são denominados **discretos**, quando conseguimos definir, dimensionar e até mesmo controlar a lista dos possíveis valores que uma coluna pode assumir. Ex.: Sexo (M/F), Estado Civil (Casado/Solteiro/Viúvo/Divorciado). Já os **amplos** estão presentes quando não conseguimos definir, nem conhecer previamente, dimensionar e mesmo controlar a lista dos possíveis valores que uma coluna pode assumir. Ex.: Nome do Empregado.

Atributos (colunas)

Os atributos referem-se ao nome de um papel desempenhado por algum domínio no esquema de uma relação/tabela. O grau (ou aridade, do inglês *arity*) de uma relação/tabela refere-se ao numero de atributos/colunas do esquema dessa relação/tabela. Dentro de uma mesma relação cada atributo (coluna) deverá ter nome distinto. Cada atributo em um determinado momento no tempo terá o seu valor preenchido com algum possível valor retirado do domínio especificado para ele ou terá valor nulo (*null*). Nulo é um tipo especial de valor atribuído a um atributo (coluna) para expressar a ausência de qualquer valor para aquele dado naquele instante do tempo. Uma visão interessante descreve um atributo como "diferentes interpretações para um domínio" (ELMASRI/NAVATHE, 2005,p.90,91,92).

Tuplas (linhas)

Uma tupla representa a materialização de um conjunto de atributos relativos a um objeto/individuo representado pela relação/tabela e modelado através do esquema de tal relação/tabela. Como o esquema de uma relação é um conjunto de atributos representativos das características do objeto modelado, cada tupla pode ser vista então como uma lista ordenada de valores de atributos (ELMASRI/NAVATHE, 2005, p.90,91).

Relações (Tabelas)

Vimos que no universo do Banco de Dados Relacional tudo gira em torno de tabelas. Um elemento importante sobre as tabelas é que elas tratam da representação lógica e não física dos dados. Esta estrutura lógica engloba os Níveis Conceitual e Externo da arquitetura de 3 camadas ANSI/SPARC. A teoria relacional não tem nada a ver como o Nível Interno, mas sim com a aparência dos dados para o usuário.

Geralmente uma tabela será a representação de uma entidade no Modelo Entidades Relacionamentos. Dizemos geralmente, pois algumas tabelas podem surgir na fase de Projeto do Sistema fruto de normalização. Maiores detalhes sobre Normalização encontram-se na Unidade 4. Cada tabela será formada por um conjunto de linhas ou tuplas. Uma linha dentro de uma tabela representa um individuo dentro do conjunto de indivíduos que a tabela representa. Cada linha, por sua vez, será formada por um conjunto de colunas, que nada mais são do que os atributos que caracterizam o indivíduo representado. Cada valor que poderá preencher uma coluna será retirado de um domínio de possíveis valores.

Logo abaixo temos três exemplos de tabelas. Inicialmente nos concentraremos em duas somente: M (Material) e NFC (Nota Fiscal de Compra). Na primeira (M), vemos 3 colunas (no-material#, nome, preço). Encontramos 8 linhas identificadas pelos valores da coluna no-material# como 1, 2, 3, 4, 5, 6, 7 e 8. Observe que a coluna no-material# possui um caráter especial. Isso foi utilizado para destacar essa coluna das demais. No caso, no-material# equivale à **chave primária** da tabela em questão. Falaremos de **chave primária** ainda nesse capitulo na seção **3.2.3 Aspectos de Integridade.** Por agora basta saber que esta coluna desempenha um papel essencial no gerenciamento dos dados.

A segunda tabela NFC possui as colunas no-nf#, nome, no-material$ e valor. Nesse caso, observamos um outro elemento importantíssimo no BD relacional. Refere-se ao elemento de ligação entre linhas de tabelas distintas. No caso, a **chave estrangeira**. A coluna no-material$ está inserida dentro da tabela NFC para permitir fazer a ligação entre as linhas de NFC e as linhas de M. A partir dessa coluna, temos condições de saber os demais dados relativos aos produtos constantes na Nota Fiscal de Compra. Falaremos também sobre **chaves estrangeiras** ainda nesse capítulo também na seção **3.2.3 Aspectos de Integridade.**

3.2.2 Aspecto Manipulador: Tomando por base os fundamentos da **Álgebra Relacional**, temos operadores de manipulação dos dados armazenados que geram Relações a partir das Relações origem (derivam tabelas de outras tabelas). Essa é uma importante característica dos sistemas relacionais, denominada **fechamento.** Como a saída de qualquer operação relacional é também uma Relação, significa que a saída de uma operação pode ser a entrada de outra operação. Assim podemos ter a combinação de varias operações da Álgebra Relacional, como por exemplo, fazer uma projeção sobre o resultado de uma junção. Isso nos permite escrever **operações aninhadas**, já que o resultado da execução de uma operação poderá ser a entrada para a outra.

Mas alguns podem estar se perguntando por que estudar a Álgebra Relacional. De fato, a Álgebra Relacional serve como um mecanismo de **otimização**, pois independentemente da maneira como o usuário escreve uma consulta ao banco de dados, tal consulta deverá ser reformulada em termos das operações nativas da Álgebra Relacional de modo a favorecer o **desempenho** na execução de tal consulta (DATE, 2003, p.166, p.167).

A Tabela 1 abaixo demonstra a lista das Operações Fundamentais da Álgebra Relacional.

Operações da Álgebra Relacional		
Tipo	**Operações fundamentais**	**outras operações**
Unária	selecionar projetar renomear	
Binária	produto cartesiano união diferença	interseção junção natural divisão atribuição

Tabela1 – Classificação das operações da Álgebra Relacional

Denomina-se operação unária, por operar sobre uma única relação.
Denomina-se operação binária, por operar com mais que uma relação.

Para exemplificar as operações, tomamos como ponto de partida as três tabelas descritas abaixo: NotaFiscalCompra (NFC), NotaFiscalVenda (NFV) e Materais (M).

no-nf #	nomenfc	no-material$	Valor
1	Jose	01	100
2	Geny	03	200
3	José	03	300
4	Regina	04	400
5	Regina	03	50
6	Jose	05	600

Tabela 2 – Relação de Nota Fiscal de Compra (NFC)

no-material#	nomemat	Preço
01	Blusa	2
02	Carteira	4
03	Calça	6
04	Meia	8
05	Sapato	10
06	Cinto	12
07	Bolsa	14
08	Chapéu	16

Tabela 3 – Relação de Material (M)

no-nf#	nomecli	no-material$	Valor
1	Jose	01	10
2	Geny	03	20
3	João	03	30
4	Regina	04	40
5	Regina	03	50
6	Pedro	02	60

7	Ricardo	07	70
8	Ana	01	80
9	Marcos	01	90
10	Ana	01	100

Tabela 4 – Relação de Nota Fiscal de Venda (NFV)

Para fins de exemplificação, adotaremos **NFC**, quando estivermos nos referenciando a Relação de **Nota Fiscal de Compra**, **M**, quando referenciando a Relação de **Material** e **NFV** quando referenciando a Relação de **Nota Fiscal de Venda**. Observe que nas relações NFC e NFV existe um símbolo # destacando o atributo no-nf. Tal símbolo denota que estes atributos cumprem o papel de determinante, ou seja, identificam univocamente uma tupla dentro do conjunto. Alem disso, o atributo no-material presente tanto em NFV quanto em NFC também possui um símbolo especial $ que denota que tais atributos cumprem o papel de referenciar outra relação. O atributo no-material dentro da relação NFV e NFC serve para relacionar tuplas dessas relações a cada tupla da relação M. A escolha dos símbolos # ou $ foi aleatória. Poderíamos ter adotado outro símbolo qualquer.

Operações fundamentais da Álgebra Relacional: Abaixo são descritas as operações fundamentais da Álgebra Relacional a partir das quais qualquer consulta poderá ser especificada. São elas: Seleção, Projeção, União, Diferença, Produto Cartesiano e Renomeação (SILBERSCHATZ, KORTH, SUDARSHAN, 2006, ELMASRI/NAVATHE, 2005)

- **Operação de Seleção:** Operação unária que seleciona tuplas que satisfazem a um determinado critério expresso através de uma **condição de seleção**. A operação de seleção é aplicada a cada tupla individualmente e age sobre as relações de forma a parecer que está ocorrendo um **particionamento horizontal** dos dados em dois conjuntos de tuplas: as tuplas que satisfazem a condição de seleção e são mantidas no processamento e as tuplas que não satisfazem e são descartadas. Algumas características são importantes sobre a operação de seleção:
    - O numero de atributos (**grau**) resultantes de uma operação de seleção é **sempre** igual ao grau da relação (R).
    - O numero de tuplas da relação resultante deverá ser **menor** ou **igual** ao numero de tuplas presentes na relação origem (R).
    - A fração de tuplas selecionadas por uma condição de seleção denomina-se **seletividade** da condição.

Usa-se a letra grega **sigma (σ)** para denotar a **operação seleção**. O predicado aparece no subscrito (condição de seleção) e a relação de argumento está entre parêntesis (R). O subscrito é uma expressão booleana.

$$\sigma_{\text{<condição de seleção>}}(R)$$

O predicado ou expressão booleana especificada em <condição de seleção> é composta de clausulas na forma de:

<nome do atributo> <operação de comparação> <valor constante>
Ou
<nome do atributo> <operação de comparação> <nome do atributo>

Onde:
<nome do atributo> → é o nome de um atributo em R.
<operação de comparação> → poderá ser utilizado algum dos operadores tais como: =, ≠, ≤, ≥, < e >
<valor constante > → trata-se de um possível valor constante do domínio do atributo.

Exemplo1: Listar todas as Notas Fiscais de Compra referentes ao Material de código igual a 3. Para gerar tal resultado a partir das relações demonstradas acima, teríamos a seguinte operação

$$\sigma_{\text{no\_material=3}}(NFC)$$

O resultado é uma relação que contém apenas as tuplas de NFC que satisfazem a comparação indicada, ou seja, que tenham o valor **"3"** para o domínio **no-material**, conforme tabela abaixo:

no-nf	nomenfc	no-material	valor
2	Geny	3	200
3	José	3	300
5	Regina	3	50

Tabela 5 – resultado da operação de seleção, com somente uma operação de seleção

Além da própria operação de seleção podemos combinar vários predicados em um predicado maior usando os conectivos e(Λ), ou(V) e não(¬).

Exemplo2: Listar todas as Notas Fiscais de Compra referentes ao Material de código igual a 3 e cujo valor seja maior que 100. Para gerar tal resultado a partir das relações demonstradas acima, teríamos a seguinte operação:

$$\sigma_{\text{no\_material=3 } \Lambda \text{ valor>100}} \text{ (NFC)}$$

O resultado é uma relação que contém apenas as tuplas de NFC que satisfazem a comparação indicada, ou seja, que tenham **"3"** para o domínio **no-material** e o valor seja maior que 100 para o domínio **valor** , conforme tabela abaixo:

no-nf	nomefor	no-material	valor
2	Geny	3	200
3	José	3	300

Tabela 6 – resultado da operação de seleção, com duas operações de seleção

A operação de seleção é **comutativa**, ou seja, a ordem como as operações de seleção são aplicadas não influenciam o resultado. Assim, a operação do exemplo2 poderia ser escrita da forma abaixo e o resultado seria o mesmo mostrado na tabela acima:

$$\sigma_{\text{valor>100 } \Lambda \text{ no\_material=3}} \text{ (NFC)}$$

- **Operação de Projeção:** Algumas consultas desejam obter somente alguns atributos de uma relação. Para obter tais resultados usa-se a operação de **Projeção**. A **Projeção** seleciona atributos específicos de uma relação específica, de acordo com determinado critério. Dado o esquema de uma relação, é possível gerar um resultado omitindo alguns atributos e retornando somente os atributos definidos no predicado da operação. A operação de seleção age sobre as relações de forma a parecer que está ocorrendo um **particionamento vertical** dos dados em dois conjuntos: as colunas que satisfazem a condição de seleção e são mantidas no processamento e as colunas que não satisfazem e são descartadas.

Usa-se a letra grega **pi (π)** para simbolizar a **operação projeção**. Os atributos que desejamos que sejam retornados serão listados como um subescrito de **pi** (π). Como o resultado de uma operação sempre resulta uma nova relação, **tuplas duplicadas no resultado serão eliminadas**. Exceto claro, quando na lista de atributos estiver presente um atributo chave, que por si garantirá a unicidade de cada tupla.

$$\pi_{\text{<lista de atributos>}} \text{ (R)}$$

Onde:
<lista de atributos > → lista dos atributos de R que estão sendo projetados.

Exemplo3: Listar o nome do fornecedor e o numero do material fornecido por ele.

$$\pi_{\text{nome, no-material}} \text{ (NFV)}$$

O resultado é uma relação que contém apenas os domínios indicados, no caso, nome e no-material de NFV. **As tuplas em duplicatas foram eliminadas.** Observe que para ser considerada uma duplicidade, toda a tupla deve ser analisada, ou seja, todos os atributos devem possuir o mesmo valor em mais que uma tupla. Se um atributo for diferente, então toda a tupla é considerada diferente e não será eliminada. No exemplo abaixo a tupla referente à segunda compra de Ana para o produto de código 01 não aparece novamente no resultado.

nomecli	no-material
Jose	1
Geny	3
João	3
Regina	4
Regina	3
Pedro	2
Ricardo	7
Ana	1
Marcos	1

Tabela 7 – resultado da operação de projeção

- **Operação União:** Em algumas situações necessitamos obter informações armazenadas em mais que uma relação. Precisamos então acessar mais que uma relação e buscar os elementos entre as duas relações citadas. Como resultado, a operação de **união** constrói uma relação com todas as tuplas das relações referenciadas, eliminando as duplicidades. Para se efetuar a união entre duas relações quaisquer há que garantir que as duas relações tenham a mesma quantidade de atributos e o domínio do primeiro atributo da primeira relação seja equivalente ao domínio do primeiro atributo da segunda relação e assim sucessivamente para todos os demais pares de atributos das duas relações. Essa característica denomina-se **união compatível**.

Usa-se o símbolo de **união (U)** da teoria de conjuntos para expressar a **operação união**.

R U S

Exemplo4: Listar os dados de NFC e NFV. A operação equivalente é:

(NFC) U (NFV)

O resultado é uma relação que contém todas as tuplas de NFC e de NFV, **sem repetição**. Considera-se repetição, se todos os atributos forem idênticos em mais que uma tupla.

no-nf	nomefor	no-material	valor
1	Jose	1	100
2	Geny	3	200
3	José	3	300
4	Regina	4	400
5	Regina	3	50
6	Jose	5	600
1	José	1	10
2	Geny	3	20
3	João	3	30
4	Regina	4	40

6	Pedro	2	60
7	Ricardo	7	70
8	Ana	1	80
9	Marcos	1	90
10	Ana	1	100

Tabela 8 – resultado da operação de união

Em algumas ocasiões, pode ocorrer de que as relações a serem utilizadas na operação de junção não são união compatível, ou seja, não possuem o mesmo esquema. Dessa forma, ainda é possível realizar a junção da mesma a partir da projeção dos atributos comuns de cada junção. O exemplo5 demonstra tal situação.

π <lista de atributos> (R) U π <lista de atributos> (S)

Exemplo5: Listar os dados de NFC e NFV projetando somente o nome e o no-material. A operação equivalente é:

π nome, no-material (NFC) U π nome, no-material (NFV)

O resultado é uma relação que contém todas as tuplas de NFC e de NFV, **sem repetição**. Nesse caso, algumas linhas não aparecerão nesse novo resultado. O uso da projeção cria um novo esquema para um resultado parcial e dentro desse novo esquema aparece duplicidade que no esquema do exemplo anterior não aparecia.

nomefor	no-material
Jose	1
Geny	3
José	3
Regina	4
Regina	3
Jose	5
João	3
Pedro	2
Ricardo	7
Ana	1
Marcos	1

Tabela 9 – resultado da operação de união realizada sobre uma projeção

A União é uma operação **comutativa.**

R U S = S U R

E é também **associativa.**

R U (S U T) = (R U S) U T

- **Operação Diferença**: Constrói uma relação consistindo em todas as tuplas que aparecem na primeira, mas não na segunda, do par de relações.

Usa-se o sinal de **menos (-)** para expressar a **diferença** entre as relações.

R - S

Exemplo6: Listar a diferença entre as relações NFC e NFV.

(NFC) - (NFV)

O resultado é uma relação que contém apenas as tuplas que aparecem em NFC, mas não em NFV.

no-nf	nomefor	no-material	valor
1	Jose	1	100
2	Geny	3	200
3	José	3	300
4	Regina	4	400
6	Jose	5	600

Tabela 10 – resultado da operação de diferença

A diferença **não** é uma operação **comutativa.**

NFC - NFV ≠ NFV - NFC

- **Operação Produto Cartesiano:** Constrói uma relação a partir de duas relações, com todas as combinações possíveis de pares de tuplas dessas duas relações. Quando mais de uma relação está presente em uma operação é possível ocorrer repetição do nome de um ou mais atributos nas relações citadas. Nesse caso, adota-se a regra de prefixar o nome do atributo que pode estar duplicado com o nome da relação. Os demais atributos podem ficar imunes a esta regra. O produto cartesiano pode ser feito sobre relações com esquema distinto, ou seja, ela **não** obriga que as relações sejam **união compatível**.

Usa-se um sinal de **vezes (X)** para simbolizar o **produto cartesiano.**

R X S

Exemplo7: Listar o produto cartesiano entre as relações NFC e M. A operação equivalente a esse enunciado é:

(NFC) X (M)

O resultado é uma relação que contém todas as tuplas de NFC combinadas com todas as tuplas de M. Se a primeira relação tiver 10 tuplas e a segunda 20 tuplas então o resultado terá 200 tuplas equivalente a combinação de cada tupla da primeira relação com cada tupla da segunda relação. Observe que o atributo no-material aparece prefixado com o nome da relação ao qual pertence. Os pares de tuplas que se formam do produto cartesiano não necessariamente são correlatos, ou seja, que possuam algum elemento de ligação válida. No caso de NFC e M, existe o atributo no-material em ambas as relações e que poderia ser usado para encontrar as Notas Fiscais referentes de cada Material. No entanto, tal restrição não se aplica ao produto cartesiano, permitindo com isso que a combinação das tuplas de NFC com as tuplas de M seja feita de maneira generalizada, sem nenhum critério de seleção.

no-nf	nomefor	NFC.no-material	valor	M.no-material	nomemat	preço
1	Jose	1	100	1	Blusa	2
1	Jose	1	100	2	Carteira	4
1	Jose	1	100	3	Calça	6
1	Jose	1	100	4	Meia	8
1	Jose	1	100	5	sapato	10
1	Jose	1	100	6	Cinto	12
1	Jose	1	100	7	Bolsa	14
1	Jose	1	100	8	Chapéu	16
2	Geny	3	200	1	Blusa	2
2	Geny	3	200	2	Carteira	4
2	Geny	3	200	3	Calça	6
2	Geny	3	200	4	Meia	8
2	Geny	3	200	5	sapato	10
2	Geny	3	200	6	Cinto	12
2	Geny	3	200	7	Bolsa	14

2	Geny	3	200	8	Chapéu	16
3	João	3	300	1	Blusa	2
3	João	3	300	2	Carteira	4
3	João	3	300	3	Calça	6
3	João	3	300	4	Meia	8
3	João	3	300	5	Sapato	10
3	João	3	300	6	Cinto	12
3	João	3	300	7	Bolsa	14
3	João	3	300	8	Chapéu	16
4	Regina	4	400	1	Blusa	2
4	Regina	4	400	2	Carteira	4
4	Regina	4	400	3	Calça	6
4	Regina	4	400	4	Meia	8
4	Regina	4	400	5	Sapato	10
4	Regina	4	400	6	Cinto	12
4	Regina	4	400	7	Bolsa	14
4	Regina	4	400	8	Chapéu	16
5	Regina	3	50	1	Blusa	2
5	Regina	3	50	2	Carteira	4
5	Regina	3	50	3	Calça	6
5	Regina	3	50	4	Meia	8
5	Regina	3	50	5	Sapato	10
5	Regina	3	50	6	Cinto	12
5	Regina	3	50	7	Bolsa	14
5	Regina	3	50	8	Chapéu	16
6	Jose	5	600	1	Blusa	2
6	Jose	5	600	2	Carteira	4
6	Jose	5	600	3	Calça	6
6	Jose	5	600	4	Meia	8
6	Jose	5	600	5	Sapato	10
6	Jose	5	600	6	Cinto	12
6	Jose	5	600	7	Bolsa	14
6	Jose	5	600	8	Chapéu	16

Tabela 11 – resultado da operação de produto cartesiano

O resultado demonstrado por um produto cartesiano não parece ter muita utilidade, ou não aparenta atender a nenhum propósito. Seria mais adequado que obtivéssemos somente as notas fiscais pertencentes a um determinado produto. Nesse caso, precisaríamos aplicar algum critério de seleção ao produto cartesiano gerado. A operação abaixo propõe uma solução para obter tal resultado.

$$\sigma_{\text{nfv.no-material = m.no-material}} \text{ (NFC) X (M)}$$

Com essa modificação teríamos o resultado descrito abaixo. Nesse caso, somente as tuplas que formaram par através da coluna no-material permanecem no resultado. Esse novo resultado certamente oferece alguma informação útil, ao contrário do resultado produzido pelo produto cartesiano.

no-nf	nomefor	NFC.no-material	valor	M.no-material	nomemat	preço
1	Jose	1	100	1	Blusa	2
2	Geny	3	200	3	Calça	6
3	João	3	300	3	Calça	6
4	Regina	4	400	4	Meia	8
5	Regina	3	50	3	Calça	6
6	José	5	600	5	Sapato	10

Tabela 12 – resultado da operação de produto cartesiano com operação de seleção

- **Operação de Renomeação:** É útil atribuir um nome para a relação resultante de uma operação, já que, diferente das relações no banco de dados, os resultados das expressões da Álgebra Relacional não possuem um nome para referenciá-las. É possível aplicar a operação renomeação sobre uma mesma relação para obter a mesma relação com um novo nome.

Usaremos a letra grega **ro (ρ)** para simbolizar a **operação Renomeação**.

$$\rho_{<\text{novo nome}>} (R)$$

Exemplo8: Listar os dados da relação Nota Fiscal Venda renomeando o resultado.

$$\rho_{\text{nova-nfc}} (NFC)$$

O resultado é uma relação, porém com um novo nome.

nova-nfc			
no-nf	nomefor	no-material	valor
1	Jose	1	100
2	Geny	3	200
3	José	3	300
4	Regina	4	400
5	Regina	3	50
6	Jose	5	600

Tabela 13 – resultado da operação de renomeação

Operações Adicionais da Álgebra Relacional: Tais operações não acrescentam qualquer capacidade a Álgebra Relacional, mas simplificam as consultas. São elas: Interseção, Junção natural, Divisão e Atribuição.

- **Operação Interseção:** Constrói uma relação com todas as tuplas que aparecem em ambos os pares de relações específicas.

A **interseção** é representada pelo símbolo de **interseção (∩)**.

$$R \cap S$$

Exemplo9: Listar tuplas em Notas Fiscais de Compra que sejam semelhantes a uma tupla em Notas Fiscais de Venda.

$$(NFC) \cap (NFV)$$

O resultado é uma relação que contém apenas as tuplas idênticas em NFC e NFV.

no-nf	nomefor	no-material	valor
5	Regina	3	50

Tabela 14 – resultado da operação de interseção

- **Operação Junção Natural:** Com a operação de junção, partimos de um Produto Cartesiano e construímos uma nova relação, somente com os pares que satisfizerem a uma condição específica. Permite combinar uma seleção e um produto cartesiano em uma única operação. “A operação junção natural forma um produto cartesiano dos seus dois argumentos, realiza uma seleção forçando igualdade nos atributos que

aparecem nos dois esquemas de relação e, finalmente, remove atributos duplicados" (SILBERSCHATZ, 2006, p.38). Observe que **para se realizar a junção natural**, **deve haver nas duas relações um mesmo atributo que possa ser utilizado como elemento de comparação**, ou seja, que possua o mesmo nome e o mesmo **domínio**. No processo de junção, os pares de tuplas existentes em cada uma das relações são combinados, de acordo com o atributo comum (observe que todos os atributos que possuam o mesmo nome nas duas relações serão utilizados para a cláusula de junção) e formam um único esquema. Este novo esquema, em alguns casos, será a concatenação de todos os atributos das duas relações, **eliminando-se um dos atributos utilizados na clausula de seleção**.

A junção natural é representada pelo símbolo de **junção (⋈).**

$$R \bowtie S$$

Exemplo10: Buscar os dados de todas as Notas Fiscais de Venda com os dados dos respectivos Materiais.

$$(NFC \bowtie M)$$

O resultado é uma relação que contém os valores de NFC combinados aos valores de M, onde o conteúdo do atributo no-material for equivalente nas duas relações.

no-nf	Nomefor	no-material	valor	nomemat	preço
1	Jose	1	100	Blusa	2
2	Geny	3	200	Calça	6
3	João	3	300	Calça	6
4	Regina	4	400	Meia	8
5	Regina	3	500	Calça	6
6	Jose	5	600	Sapato	10

Tabela 15 – resultado da operação de junção natural

Se não for especificada nenhuma condição de junção, todas as tuplas serão qualificadas e a junção degenera em um Produto Cartesiano, também chamado de Cross Product ou Cross Join.

A **Junção Natural** é **associativa.**

$$(NFC \bowtie M) = (M \bowtie NFC)$$

Uma variação da junção denominada **junção teta** é utilizada quando na operação de junção as relações **não** possuem um atributo em comum. As características da junção teta são semelhantes as da junção natural, exceto pelo fato de que nos esquemas das relações envolvidas não existe um atributo com o mesmo que possa ser utilizado para fazer a seleção. Dessa forma, deve-se declarar explicitamente os atributos que cumprirão esta tarefa.

$$R \bowtie_{\text{<condição de junção>}} S$$

Onde:
<condição de junção > →atributos a serem utilizados como clausula de seleção, mais o operador de comparação.

Quanto na condição de junção usa-se um operando de igualdade, tal junção é denominada **equijoin** ou **equijunção** (junção de igualdade).

Se o atributo utilizado na condição de junção possuir valor nulo, a tupla equivalente aquele atributo é eliminada.

- **Operação Divisão:** A operação de **divisão** é indicada para consultas que envolvam o critério "para todo". Suponha que desejássemos saber qual cliente já comprou todos os materiais cadastrados. Observe que deveríamos utilizar uma operação que verificasse para cada tupla de cliente se a mesmo está atrelada a todos os materiais.

Ela é referenciada pelo símbolo de **divisão (÷).**

$$\pi_{\text{<lista de atributos>}} (R) \div \pi_{\text{<atributo>}} (S)$$

Exemplo11: Buscar o nome de cliente que já comprou todos os produtos.

$$\pi_{\text{nomecli, no-material}} (NFV) \div \pi_{\text{no-material}} (M)$$

Esta operação mostrará o nome dos clientes que já compraram todos os materiais. A base de dados proposta no exemplo aqui não forneceu resultado a tal consulta, pois não temos clientes que tenham comprado todos os produtos cadastrados.

- **Operação Atribuição:** Esta operação possibilita atribuir parte do resultado de uma operação para uma variável temporária, semelhante ao que ocorre com as linguagens de programação.

Ela é referenciada pelo símbolo de **uma seta direcionada para a esquerda (←).**

<nome da variável> ← <atributo> (R)

Exemplo12: Atribuir a uma variável denominada nome-cliente os nomes dos clientes.

nome-cliente ← nomecli (NFV)

Esta operação irá selecionar o nome dos clientes e atribuirá o resultado a uma variável denominada nome-cliente.

Operações Estendidas da Álgebra Relacional: Propõe incrementos nas operações fundamentais, como por exemplo, permitir operações aritméticas como parte da projeção, ou mesmo a junção externa que possibilita lidar com valores nulos. São elas: Projeção Generalizada, funções agregadas e junção externa.

- **Operação Projeção Generalizada:** Esta operação estende a operação de projeção original permitindo que funções aritméticas sejam usadas na lista de projeção. Usa-se o mesmo **símbolo** da **projeção**. A modificação se dá no poder que a projeção passou a ter e não necessariamente na modificação do tipo de operação utilizada.

$$\pi_{\text{<lista de atributos>}} (R)$$

Exemplo13: Listar o nome e o código de cada material, mais o preco de cada material com desconto de 50%.

$$\pi_{\text{no-material, nomemat, preço*0.5}} (M)$$

Esta operação irá selecionar tanto os atributos do esquema M, assim como o resultado da função aritmética presente no exemplo.

no-material	Nomemat	preço * 0,5
1	Blusa	1
2	Carteira	2
3	Calça	3

4	Meia	4
5	Sapato	5
6	Cinto	6
7	Bolsa	7
8	Chapéu	8

Tabela 16 – resultado da operação de projeção generalizada

- **Operação de Funções Agregadas:** A partir de um conjunto de valores, aplicando-se uma **Função Agregada** podemos retornar um único valor como resultado. O símbolo utilizado para denotar uma função agregada é a letra **G** em fonte caligráfica. **Como infelizmente não consegui encontrar a fonte, tal qual ela é proposta na bibliografia, utilizei uma variação de G (Ģ) da lista de símbolos do Word.** Creio que esta pequena diferença gráfica não deverá comprometer em nada o entendimento da operação. A operação de **Agregação** (Ģ) significa que uma agregação deve ser aplicada e no seu subscrito é descrita a **Função Agregada** a ser aplicada sobre a agregação. As principais funções agregadas são:
  - **Sum** → obtém a soma de um conjunto de valores
  - **Avg** → obtém a média de um conjunto de valores
  - **Count** → obtém o numero de elementos presentes em um conjunto
  - **Min** → obtém o menor valor de um conjunto de valores
  - **max** → obtém o maior valor de um conjunto de valores

$$Ģ_{\text{<função agregada>}}(R)$$

Exemplo14: Obter o somatório dos valores das Notas Fiscais de Compra

$$Ģ_{\text{sum(valor)}}(NFC)$$

Esta operação irá selecionar os atributos valor da relação NFC e aplicar a função de soma sobre ela. O resultado é:

sum(valor)
1650

Tabela 17 – resultado da operação de função agregada (sum → soma)

Exemplo15: Obter a quantidade de elementos, a média dos valores, o menor e o maior valor das Notas Fiscais de Compra

$$Ģ_{\text{count(no-nf)}\ ,\ \text{avg(valor)}\ ,\ \text{min(valor)}\ ,\ \text{max(valor)}}(NFC)$$

Esta operação irá selecionar o atributo valor da relação NFC e aplicar as funções descritas. O resultado é:

count(no-nf)	avg(valor)	min(valor)	max(valor)
6	275	50	600

Tabela 18 – resultado da operação de diversas funções agregadas

Em muitos casos, deseja um resultado que expresse um subconjunto dentro do conjunto da relação. Ao invés de obter o valor total comprado, busca-se o valor total comprado de cada material. Nesse caso, a relação NFC deveria ser particionada em grupos de materiais distintos e sobre cada grupo aplicação a função. A operação abaixo demonstra tal situação.

$$_{\text{no-material}}Ģ_{\text{sum(valor)}}(NFC)$$

Esta operação irá demonstrar o numero de cada material que aparece na relação NFC e o valor total comprado do mesmo. A presença do atributo no-material do lado esquerdo da operação Ģ indica que a relação descrita no predicado (NFC) precisa ser particionada em grupos (no caso, em grupos distintos pelo numero do material)

para então ser aplicação a Função de soma (sum) sobre cada grupo. Com base nessa nova conceituação, pode-se formular a **operação de agregação** completa, conforme abaixo.

$$_{G1, G2,...,Gn}\mathcal{G}_{F1(A1), F2(A2), ...,Fm(Am)}\ (E)$$

onde:
G → refere-se à lista de atributos pelo qual se pode agrupar a relação
F → refere-se às funções agregadas
A → nome do atributo sobre o qual irá operação a função
E → qualquer expressão da Álgebra Relacional (ou relação)

As tuplas no resultado da expressão E são particionadas em grupos de tal maneira que:

- todas as tuplas que possuam o mesmo valor de G pertencem a um único grupo, distinto dos demais grupos.

Assim como numa projeção generalizada, o resultado de uma operação de agregação não possui um nome. Pode-se então usar uma operação de renomeação para fornecer um nome para tal resultado. O exemplo abaixo demonstra esta situação. Nesse exemplo, o resultado da função é renomeado para "somatório_compras". Além disso, será calculado o valor total comprado separado por material.

$$_{\text{no-material}}\mathcal{G}_{\text{sum(valor) as somatório\_compras}}(NFC)$$

a operação descrita acima obterá o seguinte resultado:

no-material	somatório-compras
1	100
3	550
4	400
5	600

Tabela 19 – resultado da operação de funções agregadas com operação de agregação

- **Operação de Junção Externa:** A **junção externa** é uma extensão da junção natural para lidar com situações onde existe ligação somente em um subconjunto das tuplas presente nas relações referenciadas. Assim, supondo a relação R e outra relação S, pode haver tuplas de R que não se relaciona a nenhuma tupla de S, pode haver tuplas de S que não se relaciona a nenhuma tupla de R e pode haver tuplas de R e S relacionadas. A **junção natural** somente consegue encontrar tuplas que formem pares, ou seja, tuplas que possuam relacionamento entre as relações citadas. A **junção externa** age como o complemento da junção natural permitindo que se encontrem as tuplas que originalmente seriam eliminadas pela ausência de correspondência entre as relações.

Podemos ter três combinações diferentes de **junção externa**: **junção externa a esquerda, junção externa a direita e junção externa completa**.

Junção externa a esquerda: seleciona todas as tuplas da relação citada a esquerda da operação de junção que não corresponde a nenhuma tupla da relação da direita. O símbolo para expressar **junção externa a esquerda** é ($\supset\!\bowtie$). Se for projetado algum resultado proveniente da relação da direita, os valores referentes a esses atributos (que não existem) são preenchidos com nulos. O resultado então é acrescentado ao resultado da **junção natural**. Dessa maneira, a relação resultante terá todas as tuplas que formaram pares nas relações citadas, mais todas as tuplas da relação citada a esquerda e que não formaram nenhum par com as tuplas da relação da direita.

$$(R \ \supset\!\bowtie_{\text{<condição de junção>}} \ S)$$

Onde:

<condição de junção > →atributos a serem utilizados como clausula de seleção, mais o operador de comparação.

Exemplo16: Listar os dados de todos os produtos e caso ele já tenha sido comprado, listar também os dados da Nota Fiscal de Compra do mesmo.

$$(M \overset{\supset}{\bowtie}_{<m.no\text{-}material = nfc.no\text{-}material>} NFC)$$

O resultado é:

M.no-material	nomemat	Preço	no-nf	Nomefor	NFC.no-material	valor
1	Blusa	2	1	Jose	1	100
3	Calça	6	2	Jeny	3	200
3	Calça	6	3	Jose	3	300
3	Calça	6	5	Regina	3	50
4	Meia	8	4	Regina	4	400
5	Sapato	10	6	Jose	5	600
2	Carteira	4	Nulo	Nulo	nulo	nulo
6	Cinto	12	Nulo	Nulo	nulo	nulo
7	Bolsa	14	Nulo	Nulo	nulo	nulo
8	Chapéu	16	Nulo	Nulo	nulo	nulo

Tabela 20 – resultado da operação de junção externa a esquerda

Junção externa a direita: simétrica a junção externa a esquerda. Seleciona todas as tuplas da relação citada a direita da operação de junção que não corresponde a nenhuma tupla da relação da esquerda. O símbolo para expressar **junção externa a esquerda** é ($\bowtie\!\subset$). Os valores referentes aos atributos da relação da esquerda (que não existe) são preenchidos com nulos. O resultado então é acrescentado ao resultado da **junção natural**. Dessa maneira, a relação resultante terá todas as tuplas que formaram pares nas relações citadas, mais todas as tuplas da relação citada à direita e que não formaram nenhum par com as tuplas da relação da esquerda.

$$(R \bowtie\!\subset_{<condição\ de\ junção>} S)$$

Onde:
<condição de junção > →atributos a serem utilizados como clausula de seleção, mais o operador de comparação.

Exemplo17: Obter os dados dos materiais que já tenham sido comprados ou não.

$$(NFC \bowtie\!\subset_{<nfc.no\text{-}material = m.no\text{-}material>} M)$$

Esta operação irá selecionar os dados dos materiais, todos os materiais, e aqueles que apareçam em alguma NFC, os dados da NFC também serão listados. O resultado é:

no-nf	nomefor	NFC.no-material	valor	M.no-material	Nomemat	preço
1	Jose	1	100	1	Blusa	2
2	Jeny	3	200	3	Calça	6
3	Jose	3	300	3	Calça	6
5	Regina	3	500	3	Calça	6
4	Regina	4	400	4	Meia	8

6	Jose	5	600	5	Sapato	10
Nulo	Nulo	Nulo	Nulo	2	Carteira	4
Nulo	Nulo	Nulo	Nulo	6	Cinto	12
Nulo	Nulo	Nulo	Nulo	7	Bolsa	14
Nulo	Nulo	Nulo	Nulo	8	Chapéu	16

Tabela 21 – resultado da operação de junção externa a direita

Junção externa completa: Une as características das duas junções em uma única operação. Preenche com nulo os dados da relação tanto a direita quanto a esquerda das relações que não satisfizerem ao critério da junção. Acrescenta esse novo resultado ao resultado da junção natural. O símbolo pra expressar **junção externa completa** é (⟗)

$$(R \text{ ⟗}_{\text{<condição de junção>}} S)$$

Onde:
<condição de junção > →atributos a serem utilizados como clausula de seleção, mais o operador de comparação.

Exemplo18: Obter os dados das notas fiscais de compra e das notas fiscais de venda.

$$(NFC \text{ ⟗}_{\text{<nfc.no-material = nfv.no-material>}} NFV)$$

Primeiro será executada uma junção natural buscando os pares de tuplas das relações citadas. Depois será feira uma junção externa a esquerdo e acrescentado ao resultado anterior e por fim uma junção externa a direita e também será acrescentado ao resultado. O resultado final é:

NFC. no-nf	Nomefor	NFC. no-material	NFC. valor	NFV. no-nf	Nomecli	NFV. no-material	NFV. valor
1	Jose	1	100	1	Jose	1	10
1	Jose	1	100	8	Ana	1	80
1	Jose	1	100	9	Marcos	1	90
1	Jose	1	100	10	Ana	1	100
2	Jeny	3	200	2	Geny	3	20
2	Jeny	3	200	3	João	3	30
2	Jeny	3	200	5	Regina	3	50
3	Jose	3	300	2	Geny	3	20
3	Jose	3	300	3	João	3	30
3	Jose	3	300	5	Regina	3	50
4	Regina	4	400	4	Regina	4	40
5	Regina	3	50	2	Geny	3	20
5	Regina	3	50	3	João	3	30
5	Regina	3	50	5	Regina	3	50
6	Jose	5	600	Nulo	Nulo	Nulo	Nulo
Nulo	Nulo	Nulo	Nulo	6	Pedro	02	60
Nulo	Nulo	Nulo	Nulo	7	Ricardo	07	70

Tabela 22 – resultado da operação de junção externa completa

3.2.3 Aspectos de Integridade: Restrições de Integridade são restrições sobre os valores que uma coluna poderá receber. Um Sistema não poder garantir a verdade, apenas a consistência. Embora pareça semelhante, há uma diferença fundamental entre a verdade (os dados no banco de dados refletem o estado de coisas verdadeiras no mundo real) e consistência (os dados no banco de dados não ferem a nenhuma restrição de integridade definida). Um banco que expressa a verdade é certamente um banco de dados integro, mas o inverso não é verdade, pois mesmo integro um banco de dados pode não estar de acordo com a verdade do mundo real.

A integridade de um Banco de Dados Relacional pode ser garantida a partir de três conjuntos distintos de restrições: **Restrições Baseadas em Modelo, Restrições Baseadas em Aplicação e Restrições Baseadas em Esquema** (ELMASRI/NAVATHE, 2005)

Restrições Baseadas em Modelo → Referem-se ao conjunto de restrições inerentes ao Modelo de Dados. As características das relações citadas abaixo formam o conjunto de restrições baseadas em Modelo. Tais características buscam distinguir uma relação de um arquivo e de uma tabela. São elas: A Ordenação de tuplas em uma relação, Ordenação de valores de uma tupla e uma definição alternativa de uma relação, Valores e Nulls nas tuplas e Interpretação de uma relação. Falaremos brevemente sobre cada uma destas restrições abaixo:

- Ordenação de tuplas em uma relação → Se baseia nos fundamentos matemáticos que propõem que dentro de uma relação as tuplas não precisam seguir nenhuma ordenação. Diferente dos arquivos e das tabelas que precisam ter uma seqüência de armazenamento ou de exibição.

- Ordenação de valores dentro de uma tupla e uma definição alternativa de uma Relação → Mantendo o rigor conceitual matemático, os atributos que compõem o esquema de uma relação devem estar dispostos em uma ordenação pré-definida e não deve ser alterada. No entanto uma definição alternativa preconiza que a partir do par formado pelo nome do atributo e pelo seu correspondente valor, não faria diferença a ordem que um atributo aparece na relação, pois como a junção dos elementos (esquema do atributo e instância do mesmo) teríamos acesso a qualquer elemento de dado dentro de uma tupla. Nessa nova abordagem, uma tupla pode ser considerada como um conjunto de pares formados por o cabeçalho de um atributo e o seu corpo (seu valor).

- Valores e Nulls na tuplas → cada valor em uma tupla é um valor atômico, não divisível em outros componentes. Atributos compostos e multivalorados não tem suporte para serem implementados no Banco de Dados Relacional. Quanto ao conceito de null, tanto pode representar valores desconhecidos para um determinado atributo de uma tupla ou simplesmente que para aquela tupla o valor daquele atributo não se aplica.

- Interpretação de uma relação → o esquema de uma relação expressa uma declaração de um tipo de asserção ou uma declaração. Um esquema expressa que elementos do objeto estudado serão armazenados e que a relação entre os elementos refletem características verdadeira sobre o objeto. Já a instância reflete um fato daquela asserção. Por exemplo, podemos dizer que a entidade NFC possui um no-nfc, um nome, um no-material e um valor. Uma instância de NFC poderia ser uma Nota Fiscal de Compra de numero 01, cujo fornecedor chama-se José, que comprou o material de código 01 e o valor total da nota é de 100.

Restrições Baseadas em Aplicação → Referem-se ao conjunto de restrições que não podem ser expressas nos modelos de dados e devem ser mantidas pelos programas de aplicação. Não trataremos dessas restrições nesse material por entendermos que se trata de elemento externo ao estudo de Banco de Dados Relacional.

Restrições Baseadas em Esquema → Referem-se ao conjunto de restrições que podem ser expressas nos esquemas dos modelos de dados e podem ser implementadas a partir de construções DDL. Tais construções agem no nível conceitual da arquitetura de três camadas ANSI/SPARC. São elas: Restrições de Domínio, Restrições de Chaves, Restrições de Valores Nulos, Restrições de Integridade da Entidade e Restrições de Integridade Referencial.

- Restrições de Domínio → Tais restrições garantem que dentro de cada tupla, o valor de um determinado atributo deva pertencer ao conjunto dos possíveis valores para aquele atributo, ou seja, que o valor do atributo seja retirado do Domínio desse atributo.

- Restrições de Chave → Tais restrições buscam garantir que em uma relação não existam duas tuplas iguais. Essa garantia poder ser obtida por um atributo ou um conjunto de atributos que devem cumprir a dois princípios básicos: o **principio da unicidade** e o **principio da minimalidade.** O **principio da unicidade** garante que o conteúdo de um atributo não se repita em outra tupla de uma relação, em outras palavras, não pode haver duas linhas em uma tabela com o mesmo valor na coluna que esteja no papel de chave. Já o **principio da minimalidade** busca garantir que se retirarmos quaisquer atributos da composição de uma chave ela não mais poderá garantir a unicidade, ou seja, se retirarmos alguma coluna que componha a chave, a garantia de unicidade não mais é verdadeira. É possível que uma relação possua mais que uma coluna que cumpra as características descritas para uma chave. Nesse caso, cada uma das colunas poderá ser vista como uma **chave candidata**. Do elenco de chaves candidatas, uma deverá cumprir o papel de unicidade das tuplas, e essa chave então será denominada **chave primária** ou (PK) do inglês Primary Key**.** As demais colunas serão chamadas de **Chaves Alternativas**. Alem de garantir a unicidade das linhas em uma tabela, a chave primária servirá para estabelecer a **ligação** entre tabelas, quando da tradução dos Relacionamentos identificados no Modelo Conceitual.

  Um conceito bem interessante sobre chave primaria é o seguinte: Uma Chave Primaria é como um substantivo, porque nomeia o objeto de cada linha. As outras colunas são como adjetivos, uma vez que oferecem informações adicionais sobre o objeto (PATRICK, 2002, p.13)

- Restrições de Valores Nulos → Trata-se de mais um possibilidade na especificação de um atributo. Valores nulos são aplicados a atributos cujo conteúdo não se aplique para determinada tupla ou que é desconhecido. Atributos obrigatórios são representados pela palavra **NOT NULL**, enquanto os atributos opcionais, ou seja, atributos que poderão aceitar o nulo são representados com a palavra chave **NULL**.

- Restrições de Integridade da Entidade → Uma importante motivação para existência de chaves em uma relação vem de um aspecto da teoria de conjuntos, que reconhece as relações básicas como correspondentes às entidades no mundo real. Por definição, as entidades do mundo real são passíveis de distinção, isto é, tem uma identificação única do mesmo tipo. É nesse ponto que as chaves cumprem seu papel, pois desempenham a função de identificar **univocamente** uma tupla dentro da relação. A chave dá identidade a uma entidade. Assim, um valor de chave primária que fosse totalmente nulo seria uma contradição quanto à maneira de expressar a questão; com efeito, dever-se-ia dizer que determinada entidade não teve *identidade,* isto é, não existir. Daí o nome "**Identidade da Entidade**".

- Restrições de Integridade Referencial → Este tipo de restrição age sobre duas tabelas. Surgem ainda no Modelo Conceitual como conseqüência da tradução do relacionamento entre as entidades. O objetivo é garantir a consistência de dados que representam a ligação entre tuplas das relações envolvidas. Em termos informais, uma tupla que faça referência a outra relação deve encontrar tal tupla na outra relação. O conceito de Integridade Referencial está diretamente relacionado ao conceito de **Chave Estrangeira** ou (FK) do inglês **Foreign Key**. Uma chave estrangeira é uma coluna em uma tabela cujo conteúdo deverá existir dentro de outra coluna (essa outra coluna **obrigatoriamente** será **Chave Primária**) de alguma tabela existente no esquema do banco de dados para a qual a tabela que possua a FK se referencia. Uma coluna de uma tabela poderá ser uma chave estrangeira se atender aos seguintes requisitos:

    - **Domínio** → O Domínio da Chave Estrangeira deverá ter o mesmo domínio da chave primária que ela referencia.

    - **Existência** → O valor atribuído a uma Chave Estrangeira deverá existir como um valor na coluna Chave Primária da tabela que ela referencia ou será nulo.

  Observe que a tabela de referência de uma chave estrangeira pode ser ela mesma, expressando nesse caso um **auto-relacionamento** entre os elementos de uma entidade.

Uma análise interessante sobre a Chave Estrangeira é: **O esquema de uma relação/tabela poderá ter um conjunto variado de atributos e dentre eles, poderá ter uma coluna que cumpre o papel de Chave Primária do esquema de outra relação** (SILBERSCHATZ, KORTH, SUDARSHAN, 2006, p.30)

Ações Referenciais (DATE, 2003, p.238): Quando definimos uma restrição de Integridade Referencial estamos na verdade construindo um vínculo entre duas relações ou duas tabelas que passarão a cumprir regras nos dois sentidos. No sentido da tabela que terá a coluna estrangeira serão aplicadas as restrições definidas acima (domínio e existência). Porem a tabela detentora da chave primária para a qual a chave estrangeira se referencia também passa a sofrer restrições. Tais restrições são denominadas **ações referenciais**. Tais ações estão descritas abaixo:

1) Quando se tenta EXCLUIR o valor de uma chave primária que possua chave estrangeira relacionada podemos ter três formas de implementar tal situação:

   a) CASCADE: A operação de anulação em “Cascatas” exclui as linhas equivalentes a FK para a PK excluída.

   b) RESTRICT: A operação de exclusão “restringe-se” ao caso onde não existam FKs para a PK que está sendo excluída (de outra forma é rejeitada).

   c) Em caso de omissão da ação referencial, ela será tratada como NO ACTION: Terá um funcionamento semelhante ao Restrict. Permitirá a execução da ação de exclusão somente para as chaves primárias que não possuam chaves estrangeiras atreladas. Caso possua, será tratada como restrita.

2) Quando se tenta ALTERAR o valor de uma chave primaria que possua chave estrangeira relacionada podemos ter também três formas de implementar tal situação:

   a) CASCADE: A operação de alteração em “Cascatas” altera as linhas equivalentes a FK para a PK excluída.

   b) RESTRICT: A operação de alteração “restringe-se” ao caso onde não existam FKs para a PK que está sendo alterada (de outra forma é rejeitada).

   c) Em caso de omissão da ação referencial, ela será tratada como NO ACTION: Terá um funcionamento semelhante ao Restrict. Permitirá a ação de alteração somente para as chaves primárias que não possuam chaves estrangeiras atreladas. Caso possua, será tratada como restrita.

3.2 CATÁLOGO

O **Catálogo** do Banco de Dados que também pode ser denominado o **Dicionário de Dados** tem a responsabilidade de armazenar:

- os vários esquemas que são gerados pelo Banco, seja ele Interno, Conceitual e Externo;

- todos os mapeamentos necessários para se trafegar dentro dos esquemas;

- todas as Restrições de Integridade;

- todos os critérios de Segurança.

Dizemos que o catalogo é um **metadado** ou **descritor, pois** ele armazena a descrição sobre outros dados. Através da análise do Catálogo, podemos conhecer os diversos objetos armazenados pelo SGBD, tais como:

- Tabelas existentes no Banco de Dados;

- Colunas pertencentes a cada Tabela;

- Índices de cada Tabela;

- Usuários que podem acessar o Banco de Dados;

- Restrições de Integridade (chaves, tipos de dados, obrigatoriedade).

Além disso, o próprio catálogo é elaborado em forma de Tabelas, ou seja, o Catálogo de um Banco de Dados é um Banco de Dados. Dessa forma, podemos invocar as tabelas do catálogo da mesma maneira que invocamos as tabelas que nós mesmos criamos. É um modelo de dados, que permitirá armazenar as diversas informações que você, como usuário desenvolvedor, irá construir ao longo do seu trabalho utilizando o SGBD. Olhando para o modelo de dados de um catálogo, temos a oportunidade de conhecer um modelo, elaborado pelo próprio fabricando do SGBD e dessa forma ter alguma idéia de como o fabricante concebe o próprio processo de Modelagem de Dados. Parece razoável pensar que os fabricantes de SGBD tenham um bom conhecimento de Modelagem de Dados.

Os dados armazenados no catálogo dão suporte a muitas das ações realizadas pelo Gerenciador de Banco de Dados no desempenho de suas funções, como por exemplo (DATE, 2003, p.60):

- A partir dos usuários cadastrados o gerenciador poderá validar quais usuários terão acesso a quais porções ao banco de dados, bem como, que tipo de ação cada usuário poderá fazer na sua porção;

- A partir dos esquemas armazenados nos catálogos o gerenciador poderá realizar as operações vindas do mundo externo, como inclusões, alterações, exclusões e consultas e avaliar se os requisitos definidos em cada operação estão adequados aos esquemas definidos;

- A partir de informações sobre índices, o gerenciador poderá otimizar um processo de busca ao dado requisitado pelo usuário.

3.3 VISÕES

Visões, como o próprio nome diz, são formas diferentes de se visualizar um determinado objeto, uma determinada coisa, um determinado dado.

Criamos Visões, para dar pontos de vista diferentes sobre o mesmo conjunto de dados. Dessa forma, usuários distintos poderão enxergar o banco de dados, de maneira "personalizada".

A Modelagem de Dados, seguindo os critérios de Normalização, pode, muitas vezes, gerar estruturas de armazenamento extremamente fragmentadas, bem diferentes da imagem conceitual que os usuários tenham sobre estes dados. Porém, para facilitar o entendimento dos usuários, sem com isso adulterar a estrutura formal normalizada, é comum que se gere visões sobre estes dados fragmentados, juntando-os novamente, dando a impressão que fisicamente, eles também estejam unidos.

Criamos visões também para proteger os dados da visualização indevida por pessoas não autorizadas.

Outras vezes, visões são criadas para impedir que os usuários tenham acesso à estrutura original da tabela, manipulando uma nova estrutura que aponta para a mesma área de dados da tabela que a visão referencia.

Criamos Visões então para (HARRINGTON, 2002, p.78):

- fornecer um mecanismo de segurança para impedir que determinados usuários vejam partes de um esquema de banco de dados sobre os quais ele não tenha direito;

- simplificar o projeto de banco de dados para usuários não familiarizados com aspectos tecnológicos sofisticados;

- pré-definir e armazenar consultas complexas e que tenham um significativo volume de uso. Dessa forma, ao invés de ter que se elaborar a consulta, armazena-se uma view representando tal consulta.

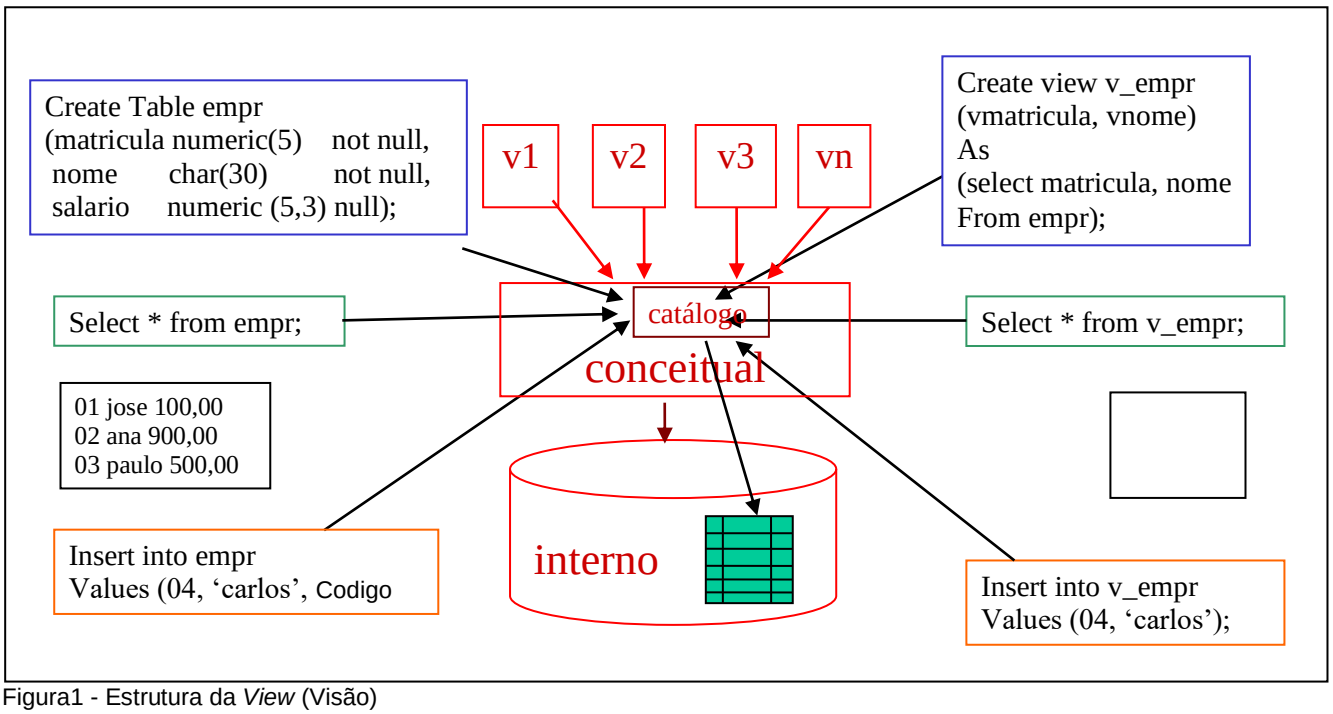

Figura1 - Estrutura da *View* (Visão)

A figura1 busca demonstrar que a Visão (View) nada mais é do que uma nova estrutura para o mesmo conjunto físico de dados. Significa dizer então, que uma view, é uma nova estrutura de dados, residente no Nível Conceitual, apontando para a mesma área de dados residente no Nível Interno/Físico. Em geral, as view não

possuem área de dados específica. Dizemos em geral, pois alguns SGBD permitem a materialização da view, mas não trataremos desse assunto nesse material. Para todos os efeitos, em qualquer ponto desse material que estivermos nos referindo a View, estaremos sempre falando View não materializada.

“Uma view não é armazenada com dados. Ao invés disso, ela é armazenada com um nome no Dicionário de Dados junto com uma consulta de banco de dados que recuperará seus dados” (HARRINGTON, 2002, p.77).

Sempre que alguém invoca um acesso a view (executar um comando SELECT sobre uma view) o SGBD executa a consulta que está armazenada no Dicionário de Dados equivalente a estrutura definida na view e colocará em memória o conteúdo que representa a view e disponibilizará esses dados para serem lidos pelo seu comando SELECT. Por isso dizemos que um SELECT em uma view significa um SELECT sobre SELECT. Os dados da view que foram gerados na memória principal estarão disponíveis somente durante a execução da instrução que invocou a view.

Podemos criar visões somente de consulta ou visões de atualização. Quando são criadas somente para consultas, as visões podem ser bem mais sofisticadas, podendo acessar diversas tabelas ao mesmo tempo. Porém, visões que são utilizadas também para atualizações devem seguir um conjunto de regras descritas abaixo, já que as atualizações realizadas apontando para a visão deverão ser propagadas na tabela que a visão referencia.

As regras para visões de atualização são as seguintes (HARRINGTON, 2002, p. 178):

1. Uma visão deve ser criada a partir de somente uma tabela ou visão.

2. Se a origem da visão for outra visão, a visão de origem também deve obedecer às regras de atualização.

3. A visão deve ser criada a partir de uma única consulta. Não se pode criar uma visão de atualização através das consultas que envolvam, por exemplo, uma união.

4. A visão deve incluir as colunas da Chave Primária da tabela.

5. A visão deve incluir todas as colunas especificadas como não nulas.

6. A visão não deve incluir nenhum grupo de dados.

7. A visão não deve remover linhas duplicadas.

3.4 A LINGUAGEM SQL

Faremos aqui um resumo sobre alguns aspectos da linguagem SQL. Informações mais detalhadas poderão ser obtidas na bibliografia recomendada ou na apostila de SQL existente na página da disciplina.

Segundo Patrick (2002, p.1) SQL (*Structured Query Language*) é uma linguagem de computador usada para obter informações de dados armazenados em uma base de dados relacional. Diferente de outras linguagens de programação, com SQL você descreve o tipo de informação que deseja obter e o computador determina o melhor procedimento para realizar essa tarefa. Linguagens com esse comportamento são denominadas **declarativas** diferente da maioria das demais linguagens de computador denominadas **procedural**. Nessas últimas, você descreve o procedimento que será aplicado ao dado e não o resultado desejado.

A linguagem SQL surgiu na década de 70 (1974), dentro dos laboratórios de pesquisa da IBM, através do trabalho do pesquisador Donald Chamberlin e outros profissionais da IBM. A linguagem tomou como modelo o ensaio sobre Bancos de Dados Relacionais desenvolvidos pelo Dr. E. F. Codd ("*A Relational Model of Data for Large Shared Data Banks*"). Inicialmente era denominada de SEQUEL (*Structured English Query Language*) e foi somente em 1979 que surgiu a primeira versão comercial, já denominada SQL (*Structured Query Language*), construída pela *Relational Software Inc*., hoje *Oracle Corporation*, por engenheiros que haviam trabalhado no projeto *SYSTEM/R* da IBM (OLIVEIRA, 2002 e MANZANO, 2002).

O primeiro padrão SQL foi definido pela ANSI em 1986 (SQL/86) e consistia basicamente na SQL da IBM com poucas modificações. Em 1989 surge a nova versão (SQL/89), já com significativas modificações. Em 1992 surge uma versão (SQL-92 ou SQL2) e em 1999 surge a mais recente versão denominada (SQL-99 ou SQL3) (OLIVEIRA, 2002, p.18 e MANZANO, 2002)

Na proposta de 1992 (SQL-92) o padrão é dividido em quatro grandes grupos: *Entry* (básico)*, Transational* (em evolução)*, Intermediate* (intermediário) *e Full* (completo)*.* A maior parte dos Gerenciadores de Bancos de Dados utilizados atualmente atende ao nível básico. Mesmo existindo uma versão mais nova do padrão, a maior parte dos SGBDs ainda utiliza, de forma básica o padrão anterior. Alguns comandos, contudo, atingem os níveis intermediário e completo (OLIVEIRA, 2002, p.18).

Diz-se que a maioria dos SGBDs Relacionais disponíveis no mercado, utiliza um super-conjunto de um subconjunto do padrão SQL. Em outras palavras, os produtos existentes não chegam a utilizar todos os recursos oferecidos pelo padrão, mas em alguns aspectos, os recursos utilizados estendem em muito ao definido pelo padrão, alcançando características dos padrões seguintes.

A linguagem SQL é dividida nos seguintes componentes (OLIVEIRA, 2002, p.19 e p.20):

- *Data Definition Language* (DDL) – Linguagem de Definição de Dados: Manutenção da estrutura dos dados. Os comandos principais são:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE

- *Data Manipulation Language* (DML) – Linguagem de Manipulação de Dados: Manipulação dos dados armazenados. Os comandos são:
  - INSERT
  - UPDATE
  - DELETE

- *Data Query Language* (DQL) – Linguagem de Consulta de Dados: Extração dos dados armazenados. O comando de extração é:
  - SELECT

- *Data Control Language* (DCL) – Linguagem de Controle de Dados: Provê segurança interna do banco de dados. Os comandos são:
  - CREATE USER
  - ALTER USER

- GRANT
- REVOKE
- CREATE SCHEMA

3.5 INDEXAÇÃO

É comum que tenhamos consultas a um banco de dados que envolva somente uma porção dos dados de um arquivo, por este motivo, não parece ser eficiente que o sistema tenha que ler cada registro existente no arquivo para então retirar a porção que interessa. O ideal é que possamos localizar somente os registros que interessam a determinada consulta e para isso estruturas adicionais associadas aos arquivos são projetadas de forma a propiciar tal busca. Estas estruturas denominam-se **índices.**

Índices em sistemas de computador funcionam de maneira semelhante a um índice de livro. A partir do conteúdo descrito no índice encontramos um determinado conteúdo existente no livro sem a necessidade de ter que percorrer cada página do livro em uma busca seqüencial por tal conteúdo. De forma semelhante agem os índices em banco de dados. Quando uma determinada consulta é executada sobre um determinado arquivo, o SGBD analisa se o arquivo possui algum índice associado ao argumento da busca. Se possuir, ele primeiro percorre o arquivo de índice para encontrar o elemento desejado e depois se desloca para o arquivo principal para recuperar os demais dados da busca (SILBERSCHATZ, KORTH, SUDARSHAN, 2006).

Um índice é um arquivo no qual, cada entrada (registro), compõe-se de dois valores, um **valor de dados** e um **ponteiro**. Cada estrutura de índice está associada a uma chave de busca particular.

Existem dois tipos básicos de índices:

- Índices Ordenados → Os valores estão classificados em determinada ordem.
- Índices de Hash → Os valores estão distribuídos uniformemente em buckets, cuja determinação se dá pela função hash. Não trataremos de Índices Hash nesse material.

Uma tabela poderá ter **índices primários**, que são índices normalmente associados à **Chave Primária**, ou seja, índices sobre uma chave de busca que está **ordenada** na tabela principal. Poderá ter também **índices secundários**, que são índices associados a outras chaves de busca que não são Chave Primária da tabela e que certamente estarão **desordenados** dentro da tabela principal.

Índices Esparsos e Densos

Existem dois tipos de índices ordenados: Os índices esparsos e os índices densos.

Índices Esparso → São também chamados de índices não-densos. Esses **não** possuem um registro na tabela de Índice para cada linha da tabela. Um registro de índice aparece para somente alguns dos valores da chave de busca. Cada registro de índice contém um valor de chave de busca e um ponteiro para o primeiro registro de dados com esse valor de chave de busca. Um determinado arquivo armazenado não pode ter mais que um índice esparso, pois o mesmo baseia-se na (única) seqüência física do arquivo em questão. Todos os outros índices, necessariamente, devem ser densos. Os índices esparsos são exclusivos para chaves de busca que estejam ordenadas na tabela principal. A figura2 abaixo é um exemplo de índice esparso.

Figura2 – Exemplo de Índice Esparso

Índices Densos → possuem um registro na tabela de índice para cada linha da tabela em questão. Este tipo de índice é o mais comum nas tabelas, pois não está vinculado a ordem física com que os dados estão armazenados na tabela de dados. Vale lembrar que os elementos repetidos na tabela de dados não serão repetidos na tabela de índices. Estes índices ocupam mais espaço de armazenamento, no entanto permitem o teste de existência de um determinado dado somente lendo a tabela de índice. Geralmente é mais rápido localizar um registro se temos um índice denso em vez de um índice esparso. No entanto, o índice esparso ocupa menos espaço e impõem menos sobrecarga da manutenção para inserções e exclusões. Os índices densos tanto podem ser aplicados a chaves de busca ordenada, quanto para chaves de busca desordenadas. A figura3 abaixo ilustra um exemplo de índice denso.

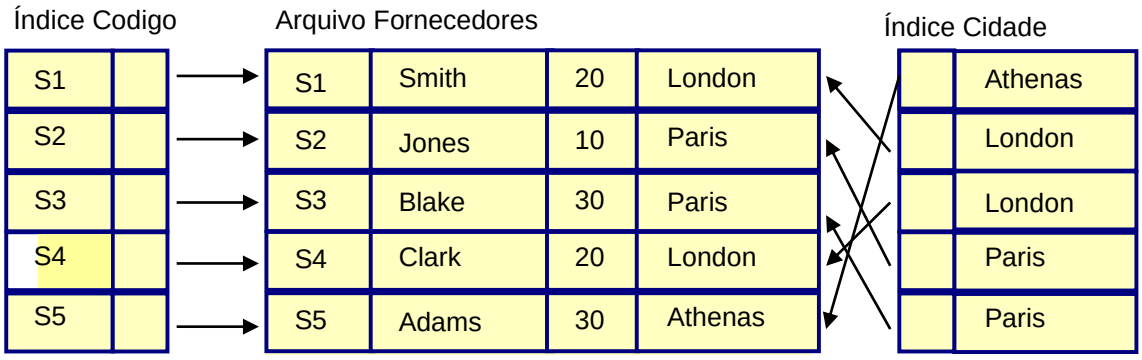

Figura3 – Exemplo de Índice Denso

A **vantagem** fundamental do índice é que o uso do mesmo acelera a recuperação, pois sempre que for feita uma leitura na tabela baseado numa determinada coluna para a qual exista um índice associado, o SGBD irá inicialmente procurar o elemento na tabela de índice e posteriormente, tendo-o encontrado nessa tabela, irá se deslocar para a tabela de dados para recuperar as demais colunas, utilizando para este deslocamento o ponteiro que acompanha cada chave de busca na tabela de índice. Caso a leitura seja somente para verificar se existe determinado registro, determinada linha com um conteúdo especifico para uma determinada coluna, e esta coluna tiver um índice associado, a validação da existência do dado poderá ser feita lendo-se somente a tabela de índice, sem a necessidade de haver leitura na tabela de dados. **Vale lembrar que o teste de existência é somente válido se o índice for denso**. Entretanto, há também uma **desvantagem**, pois a presença dos índices reduz o tempo das atualizações, já que sempre que for feita uma atualização na tabela de dados, a mesma deverá ser replicada para todas as tabelas de índices associadas às colunas da tabela alterada.

De posse dessas duas informações sobre vantagens e desvantagens, a questão que deve ser respondida quando um campo é considerado como candidato à indexação é: O que seria mais importante: **A recuperação eficiente baseada no valor do campo em questão ou a perda em atualização, em função da recuperação eficiente?**

Em geral, a indexação de uma coluna deve levar em consideração alguns aspectos importantes. O primeiro deles é o alto volume de consultas efetuadas sobre a coluna candidata a indexação. Colunas que são alvo de consultas constantes são candidatas a serem indexadas. Além disso, outro fator que se deve levar em consideração para justificar a indexação da coluna é o índice de repetições que podemos encontrar nos valores dos dados dessa coluna. Algumas vezes, ainda que uma coluna seja indexada, se o valor de busca possui um volume de repetições muito grande, o próprio SGBD poderá decidir por ignorar o índice e fazer um acesso seqüencial. Observe que, mesmo havendo o índice, a estratégia de acesso do SGBD optou pela leitura seqüencial por julgar ser este método mais eficiente.

Índices Multiníveis

Em algumas situações se o volume de dados da tabela principal for muito grande, a própria tabela de índices poderá também ficar muito volumosa. Se um índice for pequeno o suficiente para ficar na memória uma busca

sobre o mesmo será rápida, mas se o índice for muito grande então deverá ser armazenado em disco e uma busca sobre ele causará vários acessos a disco. Em alguns casos poderá ser aplicada uma pesquisa binária, mas em geral, o processo de busca de um índice grande pode ser dispendioso. Para lidar com esse problema, os índices são tratados como qualquer outro arquivo seqüencial, construído um índice esparso sobre um índice ordenado. Agora a partir do índice esparso encontramos valores que apontam para outro nível de índice e esse novo nível de índice aponta para a tabela de dados.

BIBLIOGRAFIA

DATE, C.J. **Introdução a Sistemas de Banco de Dados.** 8ed. Rio de Janeiro: Elsevier, 2003.

ELMASRI, R., NAVATHE, S. B. **Sistemas de Banco de Dados.** 4ed. São Paulo: Adisson Wesley, 2005.

KORTH, H. F. & SILBERSCHATZ, A. **Sistema de Bancos de Dados.** 3ed. São Paulo: Makron Books, 1999.

HARRINGTON, J. L. **Projetos de Bancos de Dados Relacionais**. Teoria e Prática. 2ed. Rio de Janeiro: Campus, 2002.

MANZANO, J. A. N. G. **SQL – Structured Query Language.** São Paulo: Érica, 2002.

OLIVEIRA, C. H. P. de. **SQL. Curso Prático.** São Paulo: Novatec Editora Ltda, 2002.

PATRICK, J.J. **SQL Fundamentos.** 2ed. São Paulo: Berkeley Brasil, 2002.