Universidade Federal de Viçosa - Campus Florestal
Banco de Dados - CCF 221
Portifólio 2

Nome: Gabriel Vitor da Fonseca Miranda
Matrícula 3857

Unidade 3 - Modelo Relacional e Projeto Lógico

Aula 3.1 - Modelo Relacional e Projeto Lógico

Modelo relacional

Representa o banco de dados como uma coleção de relações.
Relação é uma tabela de valores

- Linha: Representa uma coleção de valores de dados relacionados; um fato normalmente corresponde a uma entidade ou relacionamento do mundo real; chamada de tupla.

Nome de tabela e de colunas: ajudam a interpretar o significado dos valores em cada linha; são os metadados;

- Domínio: Conjunto de valores atômicos;
- Atômico: Cada valor é indivisível;
- Especificando um domínio: por meio da especificação de um tipo de dado;
- Esquema relacional: Conjunto de esquemas de relações, cada um com sua lista de atributos: R(A1, A2, …, An);
- Atributo Ai: o nome de um papel desempenhado por algum domínio D em um esquema de relação.
- Nomes de atributo: indicam diferentes papeis, ou interpretações, do domínio;
- Grau de uma relação: número de atributos.

Relação (ou estado da relação)

- Conjunto de tuplas r = {t1, t2, …, tn}
- cada tupla é uma lista ordenada de n valores, onde cada valor é um elemento do domínio do respectivo atributo ou é um valor especial “nulo” (NULL);
- é uma relação matemática de grau n sobre os domínios dos n atributos; é portanto um subconjunto do produto cartesiano dos domínios;

Estado da relação atual, estado em determinado momento, contendo apenas as tuplas válidas que representam em estado particular do mundo real;

Interpretação do significado de uma relação:

Cada tuplas é um fato ou uma instância em particular da afirmação.

Ordenação de tuplas em uma relação:
- uma relação é definida como um conjunto de tuplas;
- elementos não possuem ordem;

Ordem dos atributos: deve ser garantida a correspondência entre atributos e seu valores;
uma tupla pode ser considerada um conjunto de pares atributo-valor.

Valores nas tuplas:
- cada valor em uma tuplas é um valor atômico;
- atributos multivalorados precisam ser representados por relações separadas;
- atributos compostos são representados apenas por seus atributos simples;

Significado para valores NULL;
- Valor desconhecido;
- Valor existe mas não está disponível;
- atributos não se aplica à tupla(valor indefinido);

Aula 3.2 Esquema Relacional e Projeto Lógico

Restrições no Modelo Relacional
- Restrições baseadas na aplicação ou restrições semânticas ou regras de negócios
    - Podem ser expressas diretamente nos esquemas do modelo de dados
- Restrições baseadas na aplicação ou restrições semânticas ou regras de negócios
    - Não podem ser diretamente expressas nos esquemas;
    - Expressas e impostas pelos programas de aplicação;

Restrições de Domínio

Normalmente incluem:
- Dados numéricos padrão para inteiro e número reais
- Caracteres
- Booleanos
- Cadeia de caracteres de tamanho fixo
- Cadeias de caracteres de tamanho variável
- Data, hora, marcador de tempo
- Moeda
- Outros tipos de dados especiais

Duas tuplas não podem ter a mesma combinação de valores para todos os seus atributos

- Superclasse: um conjunto dos atributos da relação onde duas tuplas distintas nunca possuirão os mesmo valores para todos os atributos;
- Chave: uma superchave mínima, ou seja, se um de seus atributos for removido, deixa de ser uma superchave;
- Chave candidata: um esquema de relação pode ter mais de uma chave;
- Chave primária: escolhida entre as candidatas, sublinhando-se seus atributos;
- As outras chaves candidatas são chamadas apenas de chave únicas.

Esquema de um Banco de Dados Relacional

É composto por:

- Conjunto de esquemas de relação;
- Conjunto de restrições de integridade;

Estado de um banco de dados relacional: composto pelos estados das relações;
Estado inválido: não obedece a todas as restrições de integridade;
Estado válido: satisfaz a todas as restrições de integridade;

Integridade e Integridade Referencial

Restrições de integridade de entidade;

- Nenhum valor de chave primária pode ser null;

Restrição de integridade referencial;

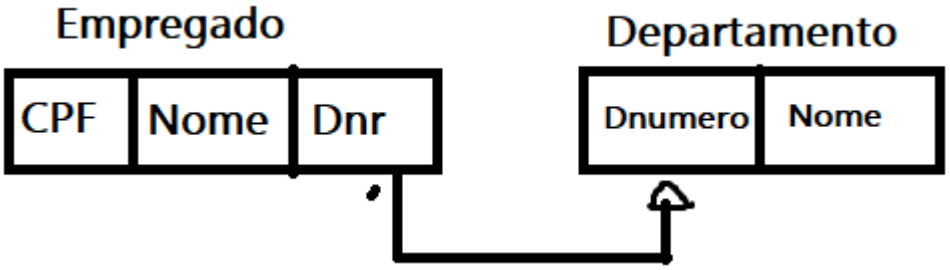
- especificada entre duas relações;
- mantém a consistência entre tuplas das duas relações, por meio do uso de chaves estrangeiras;

Regras de chaves estrangeiras:

- os atributos em ChE têm os(s) mesmo(s) domínio(s) que os atributos da chave primária ChP que referenciam;
- um valor de ChE em uma tupla t1 de um estado de relação de r1 da relação R1 aparece como um valor de ChP para alguma tupla t2 no estado de relação r2 da relação R2 ou é nulo;
- em um diagrama para representar um esquema de banco de dados relacional, um Che é representada por um arco direcionado saindo dela para o ChP que ela referencia;

Projeto Lógico: Esquema relacional

O projeto lógico seria então fazer um esquema relacional, considerando chaves primárias e chaves estrangeiras, mas sem considerar aspectos ligados ao projeto físico, como por exemplo, tipos de dados;



Exemplo:

cpf:15854, nome:gabriel, dnr:5
cpf:15854, nome:gabriel, dnr:null

Outros tipos de restrições
Restrições de integridade semântica:

- mecanismo chamado triggers e assertions podem ser utilizados, mas é comum tratar nos programas da aplicação;

Dependência funcional:

- o valor de X determina um valor exclusivo de Y(veremos mais normalização);
- Restrições de estado: restrições extras que um estado válido do banco de dados precisa satisfazer;
- Restrições de transição: restrições para lidar com mudanças de estado não permitidas;

Operações Básicas
Operações básicas que podem mudar os estados das relações:

- inserir
- excluir
- alterar

Operação Inserir

- Uma nova lista de valores é inserida como uma nova tupla em uma relação;
- Pode violar qualquer uma das restrições;
- Se uma inserção violar uma ou mais restrições, a opção padrão é rejeitá-la;

Operação Excluir

Pode violar apenas a restrição de integridade referencial, no caso da tupla a ser excluída estiver sendo referenciada por chaves estrangeiras:
O que fazer? Pode ser configurado:

- Restrict: rejeita a exclusão;
- Cascade: propaga a exclusão;

- Set null ou Set default: modifica as chaves estrangeiras que causam a violação da restrição para null ou para um valor padrão;

Operação alterar
Seleciona as tuplas a serem modificadas, e se os atributos a serem alterados não fazem parte de uma chave primária nem de uma chave estrangeira, não causa problemas;

Se a alteração envolve alterar chave primária/estrangeira, teremos questões semelhantes às operações inserir/excluir,

Aula 3.3 Modelo Relacional e Projeto Lógico

Mapeamento ER/ERE para relacional
Veremos um algoritmo para obter o projeto lógico para banco de dados relacional, ou seja, o esquemas relacional, a partir de um diagrama ER/ERE.

Etapa 1:
- Para cada tipo de entidade forte E, crie uma relação E que inclua todos os atributos simples.
- Inclua somente os atributos de componentes simples de um atributo composto.
- Escolha um dos atributos chave como chave primária.

Etapa 2:
- Para cada tipo de entidade fraca W com uma entidade “proprietária” E, crie uma relação R, e inclua todos os atributos simples (ou componentes simples de atributos de chave como primária da relação ou relações que correspondem às entidade “proprietárias”.
- Inclua também como chave estrangeira de R os atributos de chave primária da relação ou relações que correspondem às entidade “proprietárias”.
- A chave primária de R é a combinação das chaves primárias das entidade proprietárias e a chave parcial de W, caso haja alguma.

Etapa 3:
- Para cada relacionamento R 1:1, identifique as relações S e T que correspondem às entidade participantes.
- Escolha uma das relações (S, por exemplo) e inclua como chave estrangeira em S a chave primária de T.
- É melhore escolher a entidade que tenha participação total em R no papel de S.
- Inclua todos os atributos simples(ou componentes simples de atributos compostos) do relacionamento de R como atributos de S.

Mapeamento alternativo: um relacionamento R 1:1, no qual ambas as entidade tem participação total, pode ser mapeado incorporando-se dois tipos de entidade e o relacionamento em uma única relação.

Etapa 4:

- Para cada relacionamento R: 1:N, identifique a relação S que representa o tipo de entidade participante do lado-N do relacionamento.
- Inclua como chave estrangeira em S a chave primária da outra relação (T). Isso porque cada instância da entidade do lado-N é relacionada a no máximo uma instância da entidade no lado-1 do relacionamento.
- Inclua quaisquer atributos simples(ou componentes simples de atributos compostos) do relacionamento com atributos de S.

Etapa 5:

- Para cada relacionamento R M:N, crie uma nova relação S para representar R.
- Inclua como chaves estrangeiras em S as chaves primárias das relações que representam as entidade participantes; suas combinação irá formar a chave primária de S.
- Inclua também quaisquer atributos simples (ou componentes simples de atributos compostos) do relacionamento como atributo de S.

- Observação: relacionamento 1:1 e 1:N podem ser mapeados de maneira similar ao M:N. Isso pode ser útil quando existem poucas instâncias de relacionamentos, no sentido de evitar muitos valores nulos em chaves estrangeiras.
- Nesse caso, a chaves primárias da relação “relacionamento” será somente uma das chaves estrangeiras que referenciam as entidades participantes.
- Para um relacionamento 1:N a chave primária será a chave estrangeira que referencia a relação da entidade do lado-N.
- Para um relacionamento 1:1 a chave primária será a chave estrangeira que referencia a relação da entidade com participação total(caso haja algum).

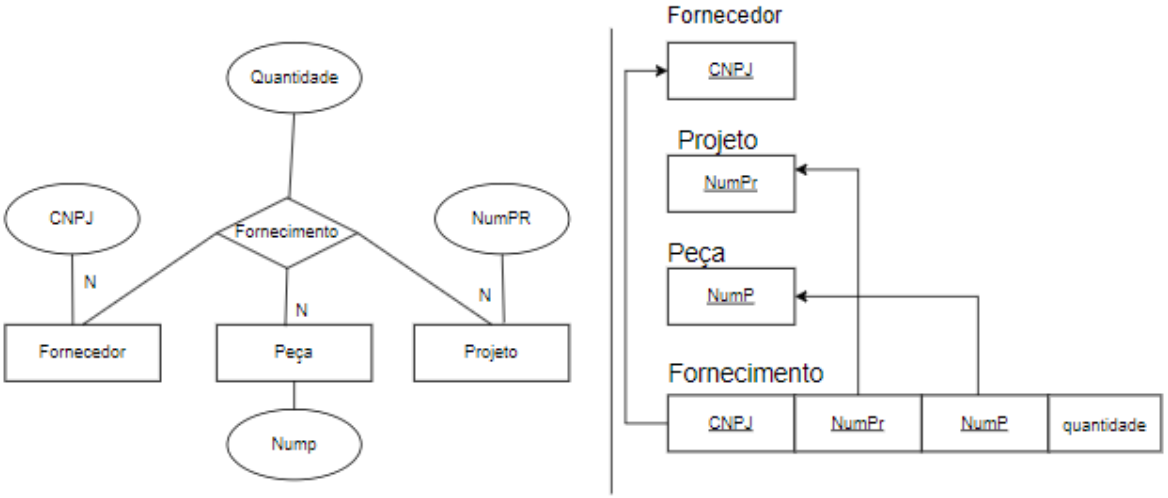
Etapa 6:

- Para cada atributo A multivalorado, crie uma nova relação R. Essa relação R irá incluir um atributo correspondente a A, além de uma chave estrangeira K que referencia a chave primária da relação que representa a entidade ou relacionamento que possui A como um atributo.
- A chave primária de R é a combinação de A e K.
- Se o atributo multivalorado for composto, deve-se incluir seus componentes simples.

Aula 3.4 Mapeamento ER/ERE para Relacional

Etapa 7:

- Para cada relacionamento n-ário R, quando n > 2 , crie uma nova relação S para representar R.
- Inclua como atributos de chave estrangeira em S as chaves primárias das relações que representam as entidades participantes.
- Inclua também quaisquer atributos simples ou componentes simples de atributos compostos do relacionamento como atributo de S.
- A chave primária de S é a combinação de todas as chaves estrangeiras de entidades com cardinalidade N.



Modelo ER/ Esquema Relacional

Etapa 8 (referente aos conceitos do ERE): Converta cada especificação com m subclasses e a superclasse para esquemas de relação, utilizando uma das 4 opções a seguir:

Opção 8A: Crie uma relação L para a superclasse C e seus atributos, além de uma relação Li para cada subclasse Si. Cada Li inclui os atributos específicos (ou locais) de Si além da chave primária da superclasse C, que é propagada para Li e se torna a sua chave primária. Pode ser usada para quaisquer restrições: disjunção ou sobreposição, parcial ou total.

Opção 8B: nesta opção, a relação L é suprimida, ou seja, seus atributos(atributos originalmente da superclasse) são colocados em cada relação Li. Esta opção só funciona bem para disjunção total.

Opção 8C: faz um mapeamento da superclasse e da subclasse para uma única relação, contendo todos os atributos genéricos e específicos e um atributos do tipo identificador. Recomendado quando as subclasses possuem atributos específicos, pois estes poderão ter valores nulos.

Opção 8D: o mesmo que a opção C porém própria para lidar com subclasses com sobreposição, incluindo m campos do tipo booleano, um para cada subclasse.

Quando temos uma hierarquia ou grade multinível, pode-se usar várias destas opções de mapeamento, um para cada caso particular.

Etapa 9: união ou categoria. Quando as classes definidoras da categoria possuem a mesma chave, basta criar uma relação para cada classe definidora e uma relação para a categoria, todas com a mesma chave primária.

Quando as classes definidoras da categoria possuem chaves diferentes, uma nova chave deve ser criada, chamada “chave substituta”. Esta será a chave primária da relação que irá representar a categoria. Será também chave estrangeira em cada uma das relações que representam as classes definidoras da categoria.

Pessoas:
|CPF| … pid

Proprietário
|pid|

Empresa:
|CNPJ| … pid

Banco
|NumB| … pid

Aula 3.5 Modelo Relacional e Projeto Lógico

Algebra Relacional

- Conjunto básico de operações para o modelo relacional, com base na teoria de conjuntos;

Expressão da álgebra relacional

- Sequência de operações da álgebra relacional;

Operações unárias: aplicadas a uma relação.

- Seleção;
- Projeção;

A operação SELEÇÂO:σ

Subconjunto das tuplas de uma relação que satisfaça uma condição de seleção:

σ<condição_seleção>(R)

<condição_seleção> é aplicada independentemente para cada tupla individual t em R. Se a condição for avaliada como TRUE, tupla selecionada para a relação resultante;

A condição pode envolver nomes de atributos e /ou constantes, operadores de comparação e operadores booleanos(AND OR NOT);

Exemplo:σ(salario >= 10000 AND salario <=20000)(Empregado)-Recuperando as tuplas Empregados que tenham entre 10000 e 20000 de salário.

Seletividade:

- Fração de tuplas selecionadas por uma condição de seleção

Operação de SELEÇÃO é comutativa;
Cascata de operações SELEÇÃO: pode ser substituída por uma única operação SELEÇÃO com operadores AND;

A operação Projeção:π

Selecionar colunas da tabela(atributos)especificados em <lista_atributos>

π<lista_atributos>(R)

Grau: Número de atributos em <lista_atributos>
Eliminação de duplicatas: o resultado da operação PROJEÇÃO é um conjunto de tuplas distintas. Tuplas duplicadas não são permitidas no modelo relacional formal, apesar que podem aparecer na SQL.
Exemplo: **π**nome, salario(**Empregado)**

Exemplo de expressão da álgebra relacional em linha:

πnome, salario(**σ**dnr = 3(**Empregado))**

Sequência de operações, com relações intermediários:
Emp_Dep3 ← **σ**dnr = 3(Empregado)
Resultado ← **π**nome, salario(Emp_Dep3)

Operação RENOMEAR: p
ps(R)
p(A1, A2, A3)(R)
ps(A1,A2,A3)(R)

Aula 3.6 Modelo Relacional e Projeto Lógico

UNIÃO, INTERSEÇÃO e DIFERENÇA

- Operações sobre o conjuntos, binárias;
- Relações usadas como entrada devem ter o mesmo tipo de tupla;

UNIÃO

- R u S;
- Inclui todas as tuplas que estão em R ou em S ou tanto em T quanto em S;

- Tuplas duplicadas são eliminadas;

INTERSEÇÃO
- R ∩ S
- Inclui todas as tuplas que estão tanto em R quanto em S

Diferença de Conjunto(ou Subtração)
- R- S
- inclui todas as tuplas que estão em R, mas não em S.

Produto cartesiano(Produto cruzado ou Junção cruzada):
- Indicado por x
- Operação de cojunto binária
- Relações que não precisam ser compatíveis na união
- Resultado: ao invés de par ordenado, uma nova tupla contendo os atributos das duas relações;
- Útil quando seguida por uma seleção que combina com valores de atributos.

A operação JUNÇÃO:
- indicada por ⋈
- Combina tuplas relacionadas de duas relações em um única tupla maior;
- Condições de junção geral tem a forma <condição> AND<condição>AND…AND<condições>

Exemplo:

Departamento⋈Dnumero=Dnr(Empregado)

JUNÇÃO NATURAL: Indicado por * faz a junção com atributos de mesmo nome nas dua relações.

Aula 3.7 - Álgebra Relacional.

Operação DIVISÃO: indicado por ➗

Aprovação	
Aluno	**Disciplina**
Gabriel	aeds 1
Gabriel	aeds 2
Érike	BD
Érike	PAA

Felipe	aeds 1
Felipe	aeds 2

aeds		Aprovação ÷ aeds
Disciplina		Aluno
Aeds1		**Gabriel**
Aeds 2		**Felipe**

Funções de agregação e agrupamento

- Funções comuns aplicadas a coleções de várias numéricos;
- Inclui SOMA, MÉDIA, MÁXIMO, MÍNIMO, CONTA;
- Agrupa tuplas pelo valor de alguns de seus atributos;

<atributos_agrupamento>f<lista_funções>(R)

Exemplo
<Dnr>f <MEDIA Salario, Conta Cpf>(EMPREGADO)

Empregado Resultado

CPF	DNR	Salario		DNR	Media	Conta
1	1	1000		1	1500	2
2	1	2000		2	2000	3
3	2	1000				
4	2	2000				
5	2	3000				

Operação de JUNÇÃO EXTERNA:

- Mantém todas as tuplas em R, ou todas em S, ou todas aquelas nas duas relações no resultado da JUNÇÃo, independentemente de ela possuirem ou não tuplas correspondentes na outra relação;
- Pode ser à esquerda, à direita ou completa.

A operação UNIÃO EXTERNA:

- Fazer a união de tuplas de duas relações que possuem alguns atributos comuns, mas que não são compatíveis na união(tipo);
- Parcialmente compatíveis:

- Todas as tuplas de ambas as relações são incluídas no resultado.
- Tuplas com os mesmo valores de combinação aparecerão apenas uma vez(atributo X)
- R(X,Y) e S(X,Z) → T(X, Y, Z)
- O mesmo que junção externa completa sobre X.

Unidade 4 - Linguagem SQL

Aula 4.1 - Linguagem SQL

Considerando um dos principais motivos para o sucesso dos bancos de dados relacionais comerciais.

SQL: Linguagem de Consulta Estruturada

Instruções para definição de dados, consultas e atualizações (LDD e LMD);

Especificação núcleo;

Mais extensões especializadas;

Terminologia:

- Tabela, linha e coluna usados para os termos do modelo relacional relação, tupla e atributo

Instruções CREATE

- Principal comando SQL para a definição de dados (LDD - Linguagem de definição de dados);

Esquema SQL:

- Identificado por um nome de esquema;
- Incluí um identificador de autorização e descritores para cada elemento

Esquema de elementos incluem: Tabelas, restrições, views, domínios e outras construções.

Cada instrução em SQL termina com um ponto e vírgula.

Instrução CREATE SCHEMA:

CREATE SCHEMA EMPRESA AUTHORIZATION “Jsilva”;

CREATE DATABASE;

Ambiente SQL:

- Instalação de um SGDBR compatível com SQL;
- Alguma forma de conexão a esta SGDB;

Comando **CREATE TABLE:**
- Especificar uma nova relação
- Dar um nome;
- Especificar atributos e restrições iniciais

CREATE TABLE EMPRESA, EMPREGADO …
OU
CREATE TABLE EMPREGADO …
Tabelas da base (relações da base)
- A tabela e suas tuplas são realmente criadas e armazenadas como um arquivo pelo SGDB;

Relações virtuais
- Criadas por meio da instrução CREATE VIEW;

```
CREATE TABLE empregado(
        cpf             CHAR(11)        NOT NULL,
        nome            VARCHAR(30)     NOT NULL,
        datanasc        DATE,
        sexo            CHAR,
        cpf_supervisor  INT             NOT NULL,
        dnr             INT,
        PRIMARY KEY(cpf),
        FOREIGN KEY(cpf_supervisor) REFERENCES EMPREGADO(cpf)
        FOREIGN KEY(Dnr)        REFERENCES DEPARTAMENTO(Dnumero));
```

Aula 4.2 - Linguagem SQL

Algumas chaves estrangeiras podem causar erros;
- Especificadas por referência circulares;
- Porque dizem respeito a uma tabela que ainda não foi criada;

Tipos de Dados:
- Numéricos: INTEGER OU INT, REAL, DOUBLE PRECISION;
- Cadeia de caracteres: CHAR, CHARACTER, VARCHAR CHARACTER VARYING;
- Cadeia de bits;
- Date AAAA-MM-DD;
- Timestamp: Date e Time, Qualificador opcional: WITH TIME ZONE;

Domínio
- Nome usado com a especificação de atributo

- Torna mais fácil mudar o tipo de dado para um domínio que é usado por diversos atributos
- Melhora a legibilidade do esquema

Exemplo:

CREATE DOMAIN TIPO_CPF AS CHAR(11);

Restrições básicas em SQL:

- Restrições de chave e integridade referencial
- Restrições sobre domínios de atributos e NULLs,
- Restrições sobre tuplas individuais dentro de uma relação

NOT NULL

- NULL não é permitido para determinar atributo

Valor padrão

- DEFAULT <valor>

Cláusula CHECK

- Dnumero INT NOT NULL CHECK(Dnumero > 0 AND Dnumero < 21);

Cláusula PRIMARY KEY

- Especifica um ou mair atributos que compõem a chave primária de uma relação
  Dnumero INT PRIMARY KEY;

Cláusula UNIQUE

- Especifica chaves alternativas(secundárias)

Dnome VARCHAR(15) UNIQUE;

Cláusula auto increment

- Especifica que a chave primária do tipo int será sempre incrementada de uma unidade, a partir do maior

Clausula auto increment

- Especificar que a chave primária do tipo int será incrementado de uma unidade, a partir do maior valor já atribuído a ela,
- Pode ser especificado o valor em uma comando INSERT, mas a partir daí o incremento se dará a parti deste valor.

Cláusula FOREIGN KEY

- Ação default: rejeita atualização sobre violação
- Opções incluem SET NULL, CASCADE e SET DEFAULT

- Ação tomada pela SGBD para SET NULL ou SET DEFAULT é a mesma para ON DELETE e ON UPDATE, a não ser que seja especificado separadamente.
- Opção CASCADE adequada para relações de “parentesco”.

Palavra-chaver CONSTRAINT
- Nome de restrição
- útil para alterações posteriores

Cláusula CHECK ao final de uma instrução CREATE TABLE
- Aplicam-se a cada tupla individualmente

CHECK (Dep_data_criacao <= Data_inicio_gerente);

Aula 4.3 - Linguagem SQL

LMD - Linguagem de manipulação de dados

Consulta ao banco de dados:
- Instrução SELECT;

Comando SELECT
- Uma instrução básica para recuperar informações de um banco de dados;

Forma básica do comando (cláusula WHERE é opcional):
SELECT <lista_atributos>
FROM <lista_tabelas>
WHERE <condição>;

Operadores de comparação:=; <; <=; >; >= e <>

Atributos de projeção
- Atributos cujos valores devem ser recuperados

Condição de seleção
- Condição booleana que deve ser verdadeira para qualquer tupla recuperada

Exemplo;

```
SELECT dnome, enome, endereco, datanasc, salario
FROM empregado, departamento
WHERE dnumero = dnr AND salario >= 10000;
```

A falta de uma cláusula WHERE
- Indica que não há condição sobre a seleção de tuplas

PRODUTO CARTESIANO
- Todas as combinações de tuplas possíveis

O mesmo nome pode ser usado para dois(ou mais) atributos:

- Desde que os atributos estejam em relações diferentes;
- É preciso qualificar o nome do atributo com o nome da relação, para evitar ambiguidade;

SELECT departamento.nome, empregado.nome, endereco, datanasc, salario
FROM empregado, departamento
WHERE dnumero = dnr AND salario >= 10000;

Apelidos ou variáveis de tupla

- Declarar nomes de relação alternativos F e S;
- empregado AS E
- empregados AS E(N, Cpf, Dn, End, Sexo, Sal, Dnr);

Exemplo:
SELECT D.nome, E.nome, endereco, datanasc, salario
FROM empregado AS E, departamento AS D
WHERE dnumero = dnr AND salario >= 1000;

SQL não elimina automaticamente tuplas duplicadas nos resultados das consultas

- Usar-se a palavra-chave DISTINCT na cláusula SELECT e então apenas as tuplas distintas irão permanecer no resultado;

Exemplo:
SELECT ALL salario
FROM empregado;

SELECT DISTINCT salario
FROM empregado;

Aula 4.4 - Linguagem SQL

Operador de Comparação LIKE

- Usado para combinação de padrão de cadeia
- % substitui um número qualquer de zero ou mais caracteres
- Sublinhando (_) substitui um único caractere

nome LIKE ‘Daniel%’ -> chamaria Daniela e Daniele

Operadores aritméticos padrão:

- Adição (+), subtração(-), multiplicação(*) e divisão (/)

Operador de comparação BETWEEN

WHERE salario BETWEEN 5000 AND 10000;

Use a cláusula ORDER BY

- Palavra-chave DESC para ver o resultado em uma ordem decrescente de valores.
- Palavra-chave ASC para especificar a ordem crescente explicitamente;
- Padrão ASC;

Exemplo:
ORDER BY D.Dnome DESC, F.Unome ASC, F.Pnome ASC

Forma do comando SELECT (cláusulas WHERE e ORDER BY são opcionais);
SELECT <lista_atributos>
FROM <lista_tabelas>
WHERE <condição>
ORDER BY <lista_atributos>;

LMD- Linguagem de manipulação de dados;
Atualização do bando de dados:

- INSERT;
- DELETE
- UPDATE;

Comando INSERT: inserir novas tuplas.
Especificar o nome da relação e uma lista de valores para a tupla:

```
INSERT INTO empregado (cpf, nome, datanasc, dnr)
VALUES('00000000-00', 'Jose', '2000-01-01', 1);
INSERT INTO empregado
VALUES ('00000000-00','Jose','2000-01-1',1);
```

Especificar o nome da relação e várias listas de valores para as tuplas;

```
INSERT INTO empregado (cpf, nome, datanasc, dnr)
VALUES('00000000-00', 'Jose', '2000-01-01', 1),
        ('00000000-00', 'Mario', '2000-01-01', 1);
```

Comando DELETE: remove tuplas de um tabela.
Inclui uma cláusula WHERE para selecionar as tuplas a serem excluidas, caso contrário todas as tuplas da tabelas serão removidas;

```
DELETE
FROM empregado
WHERE salario >= 10000;
```

Comando UPDATE: modifica valores atributos de uma ou mais tuplas selecionadas.

- Cláusula SET: especifica os atributos a serem modificados e seus novos valores.

UPDATE empregado
SET salario = 12000, dnr = 5
WHERE salario = 10000;

Aula 4.5 - Linguagem SQL

- Definição de dados em SQL(mais comandos)
- Junções.
- Funções de agregação.
- Cláusulas Group by e Having.
- Cláusula LIMIT

DROP TABLE: remove todos os dados e o próprio esquema da relação.
Exemplo:

DROP TABLE empregado;

ALTER TABLE: usado para alterar o esquema da relação.

Exemplos:
ALTER TABLE empregado ADD identidade varchar(10);
ALTER TABLE empregado DROP COLUMN ultimonome;
ALTER TABLE empregado CHANGE COLUMN nomem nomedomeio VARCHAR(45) NULL
ALTER TABLE departamento ALTER mgrssn SET DEFAULT ”333445555”;

- Junções: implementadas como definido pela álgebra relacional.

Existe o operador especial “JOIN”. É importante utilizá-lo, porque tira da cláusula WHERE condições que saõ estritamente das junções (chave primária igual a chave estrangeira, por exemplo).

Existem as variações natural join (mesmo nome para as chaves primária e estrangeiras) e outer join (junção externa). Ou seja, temos:

INNER JOIN (o mesmo que join, porque a palavra inner pode ser omitida).
NATURAL JOIN
LEFT OUTER JOIN
RIGHT OUTER JOIN
FULL OUTER JOIN
Obs: A palavra outer pode ser omitida.

Exemplos:

```
SELECT nome, endereco
FROM empregado JOIN departamento ON dnr = dnumero
WHERE dnome = “computação”;

SELECT nome, endereco
FROM empregado NATURAL JOIN
        departamento AS depto(dnr, dnome, gerente)
WHERE dnome = “computação”;
```

Exemplos:

```
SELECT E.nome AS nome_empregado, S.nome AS nome_supervisor
FROM empregado AS E LEFT OUTER JOIN
        empregado AS S ON E.cpf_supervisor = S.cpf;


SELECT nome, endereco, dnome, projnome
FROM (projeto JOIN departamento ON dnum = denumero)MAX
        JOIN empregado ON cpf_gerente = cpf
WHERE plocalização = “Florestal”;
```

Aula 4.6 Linguagem SQL

As funções de agregação servem para recuperar dados agregados, ou seja, dados que foram trabalhados sobre os dados armazenados, calculados a partir de toda uma relação ou para grupos de tuplas:

- MAX
- MIN
- AVG
- SUM
- COUNT

Encontrar a soma dos salários de todos os empregados, o maior salário, o menor salário e a média salarial:

```
SELECT SUM(salario), MAX(salario), MIN(salario), AVG(salario)
FROM empregado;
```

Encontrar a soma dos salários de todos os empregados do departamento de computação, bem como o maior salário, o menor salário e a média salarial:

```
SELECT SUM(salario), MAX(salario), MIN(salario), AVG(salario) AS media
FROM empregado JOIN departamento ON dnr = denumero
```

WHERE dnome = ‘computação’;

Recuperar o número de empregados da empresa no departamento de computação.

```
SELECT COUNT(*)
FROM empregado JOIN departamento ON dnr = dnumero
WHERE dnome = ‘computação’;
```

Contar o número de valores de salários distintos no banco de dados.

```
SELECT COUNT(DISTINCT salario)
FROM empregado;
```

Recuperar o nome dos empregado que possuem dois ou mais dependentes.

```
SELECT nome
FROM empregado
WHERE ( SELECT COUNT(*)
        FROM dependente
        WHERE cpf = dcpf) >= 2;
```

Cláusula GROUP BY: agrupa as tuplas selecionadas com base em um ou mais de seus atributos(quando este(s) atributos é são igual em duas ou mais tuplas, elas são colocadas em um mesmo grupo.

Exemplos:
Para cada departamento, recuperar o número do departamento, o número de empregados no departamento e sua media salarial.

```
SELECT dnr, COUNT(*), AVG(salario) AS media
FROM empregado
GROUP BY dnr;
```

Aula 4.7 Linguagem SQL

Cláusula GROUP BY: agrupa as tuplas selecionadas com base em um ou mais de seus atributos(quando estes(s) atributo(s) é (são) igual(is) em duas ou mais tuplas, elas são colocadas em um meso grupo).

Normalmente utilizada em conjunto com as funções de agregação. Desta forma, a função de agregação será aplicada a cada grupo, e não a todas as tuplas;

Exemplo:

Para cada departamento, recuperar o número do departamento, o número de empregados no departamento e sua média salarial.

```
SELECT dnr, COUNT(*), AVG(salario)
FROM empregado
GROUP BY dnr;
```

Para cada projeto, recuperar o número do projeto, o nome do projeto, e o número de empregados que trabalham naquele projeto.

```
SELECT pnumero, projnome, COUNT(*)
FROM projeto JOIN trabalha_em ON pnumero = pnr
GROUP BY pnumero, projnome;
```

Para cada projeto, recuperar o número do projeto, o nome do projeto, e o número de empregados do departamento 5 que trabalham naquele projeto.

```
SELECT pnumero, projnome, COUNT(*)
FROM ( projeto JOIN trabalha_em ON pnumero = pnr) JOIN empregado ON ecpf = cpf)
WHERE dnr = 5
GROUP BY pnumero, projnome;
```

Cláusula HAVING: Usada em conjunto com a cláusula GROUP BY, quando necessária.

Esta cláusula é uma condição para se selecionar os grupos que farão parte da resposta (relação resultante), ao invés de selecionar todos (o que é feito por GROUP BY sem HAVING).

```
SELECT pnumero, projnome, COUNT(*)
FROM projeto JOIN trabalha_em ON pnumero = pnr
GROUP BY pnumero, projnome
HAVING COUNT(*) > 2;
```

Imagine que se queira recuperar os nomes e salários de empregados, mas apenas daqueles cujos salários sejam maiores que a média dos salários;

```
SELECT nome, salario
FROM empregado
WHERE salario > (SELECT AVG(salario) FROM empregados);
```

Aula 4.8 - Linguagem SQL

Mais operadores: IS, IN, UNION, ALL, EXISTS, EXCEPT
Mais exemplos de sub-consultas.

Valor nulo: null pode ter um dos três significados a seguir:

- Valor desconhecido: ex: dara de nascimento desconhecida.
- Valor não disponível: ex: pessoas tem um telefone, mas não quer disponibilizá-lo.
- Atributo não aplicável: ex: último diploma, sendo que a pessoas não tem nenhum diploma.

Em todos os casos, este valor é representado por null no banco de dados, e há uma lógica especial para se trabalhar com 3 valores lógicos possíveis: verdadeiro, falso e desconhecido (nulo representando desconhecido).

Operador IS: verificar se um atributo é ou não nulo.
Exemplo: Recuperar os nome de todos empregados que não tem supervisores.

```
SELECT nome
FROM empregado
WHERE cpf_supervisor IS NULL;
```

Cada consulta em SQL retorna um conjunto de tuplas. Existem alguns operadores justamente para trabalhar com conjuntos.

Um deles é o operador UNION, que é a união de conjuntos.

```
SELECT DISTINCT pnumero
FROM (projeto JOIN departamento ON dum = dnumero) JOIN empregado ON cpf_gerente = cpf
WHERE ultimo_nome = ‘Barbosa’
UNION
SELECT DISTINCT pnumero
FROM(projeto JOIN trabalha_em ON pnumero = pnr) JOIN empregado ON ecpf = cpf
WHERE ultimo_nome = ‘Barbosa’;
```

Aula 4.9 - Linguagem SQL

Outro operador é o “IN” que significa pertencer a um conjunto.

O conjunto pode ser o resultado de uma sub-consulta ou então um conjunto explicito de valores.

```
SELECT DISTINCT ecpf
FROM trabalha_em
```

```
WHERE pnr IN (1,2,3);
```

Exemplo de pertencer a um conjunto de tuplas recuperadas por sub-consultas: “Recupere todos os números de projeto para projetos que envolvam um empregado cujo último nome é “Barbosa”, seja como funcionário do projeto ou como gerente do departamento que controla o projeto. (já é respondida anteriormente).

```
SELECT pnumero
FROM projeto
WHERE pnumero IN ( SELECT pnumero
                   FROM (projeto JOIN departamento ON dnum = denumero)
                   JOIN empregado ON cpf_gerente = cpf
                   WHERE ultimo_nome = ‘Barbosa’)
OR
pnumero IN (SELECT pnr
            FROM trabalha_em JOIN empregado ON ecpf = cpf
            WHERE ultimo_nome = ‘Barbosa’);
```

O operador “IN” pode ser usado não somente com o único operando, mas com um conjunto de operandos(comparando tuplas).

Exemplo:

```
SELECT DISTINCT ecpf
FROM trabalha_em
WHER (pnr, horas) IN (SELECT pnr, horas
                      FROM trabalha_em
                      WHER ecpf = “000000-00”);
```

Recuperar o primeiro nome de cada empregado que tenha um dependente com o mesmo primeiro nome e o mesmo sexo do empregado do qual ele é dependente.

```
SELECT pnome
FROM empregado AS E
WHERE E.cpf IN ( SELECT D.ecpf
                 FROM dependente AS D
                 WHERE E.pnome = D.pnome AND E.sexo = D.sexo);
```

Aula 4.10 - Linguagem SQL

O operador ALL permite comparações com todas as tuplas pertencentes a um conjunto.

Recuperar o primeiro nome de cada empregado cujo salário seja maior que todos os salários de empregados do departamento 5.

```
SELECT pnome
FROM empregado
WHERE salario > ALL ( SELECT salario
                      FROM empregado
                      WHERE dnr = 5);
```

O operador EXISTS é utilizado para verificar se o resultado de uma sub-consulta correlacionada é vazio ou não.

Recuperar o primeiro nome de cada empregado que tenha um dependente com o mesmo o primeiro e o mesmo sexo do empregado do qual ele é dependente.

```
SELECT pnome
FROM empregado AS E
WHERE EXISTS (   SELECT *
                 FROM dependente AS D
                 WHERE E.pnomde = D.pnome AND E.sexp = D.sexo);
```

Recuperar o primeiro nome dos empregados que não tenham dependentes.

```
SELECT pnome
FROM empregados AS E
WHERE NOT EXISTS ( SELECT *
                   FROM dependente AS D
                   WHERE E.cpf = D.ecpf);
```

Lista os primeiros nomes de gerentes que tenham pelo menos um dependente.

```
SELECT pnome
FROM empregado AS E
WHERE      EXISTS( SELECT *
                   FROM dependente AS D
                   WHERE E.cpf = D.ecpf)
           AND
           EXISTS ( SELECT *
                    FROM departamento AS T
                    WHERE E.cpf = T.cpf_gerente);
```

Aula 4.11 - Linguagem SQL

O exemplo a seguir mostra o uso de um operador chamado “CONTAINS”(contém), não existente em SQL.
Portanto deverá ser substituído por outros.

Recuperar o primeiro nome de cada empregado que trabalha em todos os projetos controlados pelo departamento de número 5.

```
SELECT pnome
FROM empregado AS E
WHERE ( ( SELECT pnr
          FROM trabalha_em AS T
            WHER E.cpf = T.ecpf)
            CONTAINS
            (SELECT pnumero
            FROM projeto
            WHERE dnum = 5) );
```

Usando o EXCEPT

Recuperar o primeiro nome de cada empregado que trabalha em todos os projetos controlados pelo departamento de número 5.

```
SELECT pnome
FROM empregado AS E
WHERE ( ( SSELECT pnumero
            FROM projeto
            WHERE dnum = 5)
            EXCEPT
            (SELECT pnr
            FROM trabalha_em AS T
            WHER E.cpf = T.ecpf) );
```

Para cada departamento que tenha mais de 5 empregados, recuperar o número do departamento e o número de seus empregados (a quantidade) que ganham mais de 40000.

Suponha que nós escrevemos a seguinte consulta incorreta:

```
SELECT dnumero, COUNT(*)
FROM empregado AS E JOIN departamento AS D ON E.dnr = D.dnumero
WHERE salario > 40000
GROUP BY dnumero
HAVING COUNT(*) > 5;
```

O que está errado?

Consulta corrigida:

```
SELECT dnumero, COUNT(*)
FROM empregado AS E JOIN departamento AS D ON E.dnr = D.dnumero
WHERE salario > 40000 AND dnr IN ( SELECT dnr
                                   FROM empregado AS E2
                                   GROUP BY E2.dnr
                                   HAVING COUNT(*) > 5)
GROUP BY dnumero;
```

Aula 4.12 - Linguagem SQL

Visões:
Uma visão ou "tabela virtual" é uma tabela derivada de outras tabelas(de base ou visões previamente definidas).

- Deve-se usar visão, por exemplo, quando são frequentes as consultas a uma junção entra várias relações, bastando definir esta junção como uma visão, e simplismente consulta a visão.
- Após ser criada(com o comando CREATE VIEW), um visão pode ser consultada normalmente, como se fosse uma relação de base.
- Ela ficará disponível até que seja executado o comando DROP VIEW.

Criando a visão:

```
CREATE VIEW trabalha_em_1 AS
SELECT pnome, unome, projnome, horas
FROM (empregado JOIN trabalha_em ON cpf = ecpf) JOIN projeto ON pnr = pnumero;
```

Consultado a visão:

```
SELECT pnome, unome
FROM trabalha_em_1
WHERE pnome = 'Daniel';
```

Esta consulta é, na verdade, mapeada pelo SGDB para a seguinte consulta, caso a visão não tenha sido "materializada", ou seja, apenas referência as relações de base.

```
SELECT pnome, unome
FROM  (empregado JOIN trabalha_em ON cpf = ecpf) JOIN projeto ON pnr = pnumero;
WHERE pnome = 'Daniel';
```

Para visões muito complexas, com várias junções, pode ser muito demorado sempre refazer a junções, pode ser muito demorado sempre refazer a junção.

Uma outra técnica, que evita o problema acima, é a implementação pelo SGBD da visão materializada, ou seja, a visão é de fato criado e armazenada temporariamente como uma nova relação no banco de dados.

Existem várias técnicas que utilizam atualização incremental para atualizar as visões sempre que as relações de base tenham sido atualizadas, no caso de terem sido materializadas

Para apagar a visão que criamos anteriormente:
DROP VIEW trabalha_em_1;

Atualizar visões(quando implementado) é complicado e pode ser ambíguo.
MySQL implementa, com restrições.

Se a visão tiver apenas uma relação definidora, e sem funções de agregação, normalmente é possível mapear a atualização da visão para uma atualização da relação definidora.

Problema: para uma visão que envolva junções, normalmente tem-se várias maneiras de se atualizar as relações definidoras!

CREATE VIEW depto_info(nome_depto, numero_de_empregados, total_salarios) AS
SELECT dnome, COUT(*), SUM(salario)
FROM departamento JOIN empregado ON denumero = dnr
GROUP BY

UPDATE dept_info
SET total_salarios = 1000
WHERE dnome = 'computação';

Aula 4.12 - Linguagem SQL

Funções:
Todo SGDB possui um grande número de funções pré-definidas.
Já vimos algumas funções do MySQL, e que estão presentes em quase todos os SGDB: as funções de agregação.
Mas existem muitas outras funções, para se trabalhar com strings, datas, etc..

Funções do MySQL: disponível no site do MySQL

Além das funções existentes, novas funções podem ser criadas no SGDB pelo usuário.
Vantagens:

- Desempenho.
- Menor tempo de desenvolvimento da aplicação.

Desvantagens:

- Dependência do SGDB.

Criar funções em MySQL:

Exemplo de funções em MySQL:

```
CREATE FUNCTION hello (s CHAR(20))
RETURNS CHAR(50)
RETURNS concat('Hello, ',s,'!');
```

Uso da função:

```
SELECT hello('world');
```

Retorno:
hello('world')
Hello, world!

Unidade 5 - Normalização de dados

Aula 5.1 Normalização de Dados

Noção central: qualidade do projeto.

- Medir formalmente porque um agrupamento de atributos em um esquema de relação é melhor que outro.
- Dois níveis:
- Lógico: interpretação dos esquemas e atributos(relações de base e visões)
- Armazenamento: como as tuplas são armazenadas e atualizadas(somente relações de base).

Bottom-up:

- considerar relacionamentos básico entre atributos como ponto de partida;
- Não é bem vista na prática: coleta grande número de relacionamento binários como ponto de partida;
- É chamado também de projeto por síntese

Top-down:

- Inicia-se com um número de agrupamentos de atributos em relações, que já tenham sido obtidas a partir do projeto conceitual
- É chamada também de projeto por análise.
- As relações vão sendo decompostas, até que as propriedade desejáveis sejam atingidas.

Semântica dos atributos da relação:
- Especifica como interpretar os valores dos atributos armazenados nas relações.
- Quanto mais fácil for explicar a semântica da relação, melhor será o projeto do esquema da relação.

Por exemplo:
- As relações empregado, departamento e projeto tem uma semântica clara e simples, cada uma representando uma entidade do mundo real.
- Relação localização_dep: é um atributo multivalorado de departamento.
- Relação trabalha_em: relacionamento M:N entre empregado e projeto

Dados redundantes em tuplas e anomalia de modificação:
- Modificar um atributo do departamento obriga a atualização de todos os empregados, para que os dados do departamento permaneçam consistentes

Aula 5.2 Normalização de Dados

Se X é uma chave candidata, isso implica que X→Y para qualquer subconjunto de atributos de Y de R.
É uma propriedade da semântica(significado dos atributos).
- Deve ser definida explicitamente: os projetistas devem usar seu entendimento da semântica dos atributos para especificar as dependências funcionais que devem se manter.
- {Orgão Emissor, Identidade} → CPF.

Inferência de DF's:
F ={cpf→{enomr, datanasc, endereco, dnumero}, dnumero → {dnome, cpf_gerente}}
Podemos inferir que:
- cpf → {dono, cpf_gerente}
- dnumero → dnome

Será utilizada a notação F | = X → Y para representar que a DF X→ Y é inferida a partir de um conjunto de dependências funcionais F.

Aula 5.3 Normalização de Dados

Primeira Forma Normal(1FN):

- Um esquema de relação R está na 1FN se todos os seus atributos forem atômicos.

- Primeira maneira de alcançar a 1FN: Expandir a chave de modo que haja uma tupla para cada localização do departamento. Problema: introduz a redundância na relação.
- Segunda maneira de se alcançar a 1FN: se um número máximo de valores for conhecido para o atributo, substitua o atributo por esse número de atributos atômicos. Problema: pode se introduzir muitos valores nulos.
- Terceira maneira de se alcançar a 1FN: remova o atributo e coloque-o em uma relação separada, juntamente com a chave primária. A chave primária desta nova relação é formada pela chave primária da relação original mais o atributo que foi movido.

Segunda Forma Normal (2FN): um esquema de ralação R está na 2FN se todo atributo de R não pertence a uma de suas chaves for totalmente dependente da chave primária.

O esquema de relação
emp_proj(cpf, pnumero, horas, enome, pnome, plocalização)
não está na 2FN porque
CPF → enomr e pnumero → {pnome, plocalização}, ou seja, há atributos em R que não são totalmente dependentes da chave primária{cpf, pnumero}.

Ou seja, a 2FN baseia-se no conceito de dependência funcional total. Se a dependência for parcial(dependente apenas de um dos atributos da chave primária), a relação não está na 2FN.

Terceira Forma Normal(3FN): Um esquema de relação R está na 3FN se estiver na 2FN e nenhum atributo de R não pertencente a uma de suas chaves for transitivamente dependente da chave primária.

Uma dependência X→ Y é transitiva se exite um conjunto de atributos Z que não é a chave candidata ou um subconjunto de qualquer chave de R para os quais ambas as dependências existem: X → Z e Z → Y.

O esquema da relação
emp_depto(cpf, enome, datanasc, endereco, dnome, cpf_gerente)
não está na 3FN porque
CPF → dnumero e dnumero {dnome, cpf_gerente}
ou seja, há atributos em R que são transitivamente dependentes da chave primária cpf.

Aula 5.4 Normalização de Dados

1FN

- Teste: A relação não deve ter qualquer atributo não-atômico nem relações agrupadas.
- Normalização: Forme novas relações para cada atributos nã-atômico ou relação aninhada.

2FN

- Teste: Para relações nas quais a chave primária contém múltiplos atributos, nenhum atributo não-chave deve ser funcionalmente dependente de uma parte da chave primária.
- Normalização: Decomponha e monte uma relação para cada chave parcial com seus atributos dependentes. Certifique-se de manter uma relação com a chave primária original e quaisquer atributos que sejam completamente dependentes dela em termos funcionais.

Aula 5.5 Normalização de Dados

Dependência Multivaloradas:

- A dependência funcional é o tipo de dependência mais importante na teoria de projeto de banco de dados relacionais.
- Mas algumas restrições não podem ser especificadas com dependências funcionais.

Essa redundância de dados é obviamente indesejável

- Mas o interessante é que esta relação"emp" está na FNBC, porque nenhuma dependência funcional se mantém em EMP.
- Entretanto, nesta relação "emp" as dependências multivaloradas enome ->> pnome e enome->>dnome (ou enome ->> pnome | dnome) se mantêm.
- Portanto é necessário definir uma forma normal mais robusto que a FNBC e que não permita esquemas de relações com emp.

Aula 5.6 Normalização de Dados

Dependências de Junção: Uma dependência de junção especifica uma restrição nos estados r de R, onde todo estado legal r de R deve ter uma decomposição de Junção sem perda (ou não aditiva).
Suponha que sempre se mantenha a seguinte restrição: sempre que um fornecedor f fornece a peça p, e um projeto j utiliza a peça p, e o fornecedor f fornece pelo menos uma peça para o projeto j, então o fornecedor f também estará fornecendo a peça p para o projeto j.

Forma Normal de chave de Domínio (FNCD):

- A ideia seria a de especificar a “última forma normal”, que leva em consideração todos os tipos possíveis de restrições e dependências que devem se manter na relação podem ser impostar, simplesmente, imponto as restrições de domínio e as restrições de chaves na relação.
- Devido à dificuldade de incluir restrições complexas em uma relação na FNCD, sua utilidade prática é limitada.

Unidade 6 - Aspectos de Implementação

Aula 6.1 Aspectos de Implementação

Indexação: árvore x pesquisa sequencial;
CREATE INDEX;
Quanto mais índices;

- Mais consultas(com mais condições) poderão ser executadas em menos tempo;
- Mais tempo será gasto em atualizações do banco de dados.

Processamento entrelaçado:

- A maior parte da teoria correspondente ao controle de ocorrência em bancos de dados é desenvolvida em termos de concorrência(modelo assumido no restante da apresentação).
- Em um SGDB, os itens de dados armazenados são os recursos que podem ser acessados concorrentemente por vários usuários e aplicações.

Conceito de transação: Uma transação é uma unidade lógica de processamento do banco de dados, que inclui uma ou mais operações de acesso ao banco de dados - essas operações podem incluir operações de inclusão, exclusão, modificação ou recuperação de dados.

Aula 6.2 Aspectos de Implementação

TIpos de falhas:

- Uma falha do computador: erro de hardware, e software ou de rede;
- Um erro de transação ou de sistema : overflow, divisão por zero, erro lógico de programação;
- Condição de exceção detectada pela transação: Uma transação pode ter que ser cancelada devido a uma condição de exceção(Saldo insuficiente em uma conta bancária). Se o programada corretamente, não constituem falhas de transações.
- Imposição do controle de concorrência;
- Falha de disco; alguns blocos de disco podem perder seus dados devido a uma disfunção de leitura ou gravação;
- Problema físicos e catástrofes: falta de energia, incêndio, roubo, sabotagem, etc.

O Log do sistema
Ponto de confirmação (Commit) de uma transação:
Gravar uma entrada no log significa um acesso a mais ao disco. Para aumentar a performance, buffers na memória principal representam blocos do log no disco, e são gravados à medida que estejam preenchido com entradas de log.
No entando, no momento do commit da transação, é feita uma gravação forçada, antes de concluí-la.
Atomicidade: unidade atômica de processamento.
Preservação de Consistência: Leva o banco de dados de um estado consistente para outro estado consistente.
Isolamento: não deve sofrer interferência de quaisquer outras transações executadas concorrentemente.
Durabilidade ou persistência:as alterações aplicadas por uma transação devem persistir no banco de dados.

Aula 6.3 Aspectos de Implementação
Escalonamento de Transações: Operações Conflitantes
Diz-se que duas operações em um escalonamento conflitam se se satisfizerem três condições:

- Pertencem a diferente transações;
- Acessam o mesmo item x;
- Perlo menos uma das operações dever ser write_item;

Caracterizando Escalonamento com base na Recuperação:
Em alguns escalonamentos, é fácil se recuperar de falhas de transações, enquanto em outros o processo pode ser bastante complicado.
Por isso, é importante caracterizar os tipos de escalonamento para os quais é possível a recuperação.

Aula 6.4 Aspectos de Implementação

Escalonamento seriais:

- Considerado que as transações sejam independentes, todo escalonamento serial é considerado correto!
- Problema: desempenho! se uma transação esperar uma operação de Entrada/Saída, não podem alternar o processador para outra transação.
- Ou seja, na prática são inaceitáveis,

Usos de seriabilidade:

- Se um escalonamento é seriável, é serial, e portanto é correto,
- Mas ao contrário do escalonamento serial, o escalonamento seriável é viável, no que diz respeito ao desempenho.

Aula 6.6 Aspectos de Implementação

Sensibilidade de Visões:

- Um escalonamento é seriável de visões ser for equivalente em visão a um escalonamento serial.
- Existe um algoritmo para testar a seriabilidade de visão, mas não tem ordem de complexidade polinomial.

No SQL Server:

- O padrão é Read COMMITTED;
- Tem-se os 4 níveis do padrão SQL, mais o nível snapshot: a transação enxerga apenas os dados confirmados logo antes de seu início. A diferença para SERIALIZABLE está nos bloqueios: Transações SNAPSHOT que lêem dados não bloqueiam outras transações de gravar dados. Transações que gravam dados não bloqueiam transações SNAPSHOT de ler dados.

Aula 6.7 Aspectos de Implementação

Controle de concorrência:

- Algumas das principais técnicas utilizadas para controlar a execução concorrente de transações baseia-se no bloqueio (locking) em itens de dados.
- Um bloqueio é uma variável associada a um item de dado que descreve o status do item com relação a possíveis operações que podem ser aplicadas ao mesmo.

Granularidade de itens de Dados:

- Quanto maior o tamanho do item de dado, menor o grau de concorrência permitido.
- Quanto menor o tamanho do item de dados, maior o número de itens no banco de dados. Consequentemente um número maior de bloqueios terá que ser gerenciado(mais operações serão realizadas e mais armazenamento será consumido)

Aula 6.8 Aspectos de Implementação

Recuperação de Banco de Dados:

- A recuperação de falhas de transações geralmente significa que o banco de dados é desenvolvido ao estado consistente mais recente, imediatamente anterior ao momento da falha.
- Informções mantidas no log do sistema são utilizadas neste processo

Atualização imediata:

- O banco de dados pode ser atualizado por operações de uma transação que antes que esta atinja seu ponto de commit.

- Mas estas operações são geralmente registradas no log em disco(através de gravação forçada) antes de serem aplicadas ao banco de dados.

Unidade 7 - Sistema de Gerenciamento de Banco de Dados(SGDB)

Microsoft SQL Server:
O Microsoft SQL Server é um sistema gerenciador de Banco de dados relacional desenvolvido pela Sybase em parceria com a Microsoft. Esta parceria durou até 1994, com o lançamento da versão para Windows NT e desde então a Microsoft mantém a manutenção do produto.

Oracle:
O Oracle é um SGBD que surgiu no fim dos anos 70, quando Larry Ellison vislumbrou uma oportunidade que outras companhias não haviam percebido, quando encontrou uma descrição de um protótipo funcional de um banco de dados relacional e descobriu que nenhuma empresa tinha se empenhado em comercializar essa tecnologia.

H2:
H2 é um banco de dados relacional escrito em Java. Ele pode ser integrado em aplicativos Java ou executado no modo cliente-servidor. Todos os modos contam com suporte para bancos de dados persistentes e na memória.

PostgreeSQL:
PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional baseado no POSTGRES, Versão 4.2, desenvolvido na Universidade da Califórnia no Departamento de Ciências da Computação em Berkeley, o qual foi pioneiro em muitos conceitos que vieram a estar disponíveis em alguns bancos de dados comerciais mais tarde

IBM DB2:
Traduzido do inglês-Db2 é uma família de produtos de gerenciamento de dados, incluindo servidores de banco de dados, desenvolvidos pela IBM. Inicialmente, eles suportavam o modelo relacional, mas foram estendidos para oferecer suporte a recursos objeto-relacionais e estruturas não relacionais como JSON e XML.

Maria DB:
MariaDB é mantido atualizado com a última versão do MySQL. Na maioria dos aspectos o MariaDB vai funcionar exatamente como o MySQL: todos os comandos, interfaces, bibliotecas e APIs que existem no MySQL também existem no MariaDB. Não há nenhuma necessidade de converter um bancos de dados para migrar para o MariaDB.