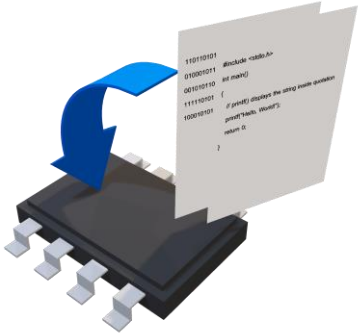
UNISINOS

DESAFIE O AMANHÃ.

BANCO DE DADOS PARA ENGENHARIA

Prof. Armando L. Keller



Modelo Relacional

Modelo Relacional

O modelo relacional foi proposto em 1970 pelo matemático britânico Edgar Frank Codd;

Até então os bancos de dados seguiam o modelo hierárquico ou de rede;

Atualmente o modelo relacional é o mais utilizado pelos SGBDs.

Modelo Relacional

O modelo relacional descreve um banco de dados como uma coleção de uma ou mais relações, onde cada relação é uma tabela com linhas e colunas.

Esta representação simples dos dados permite que as consultas, utilizando uma DDL sejam mais simples de serem expressas.

Modelo Relacional

A **relação** é o principal construtor para representar os dados em um modelo relacional. Estas relações consistem em um esquema de relação e uma instancia de relação;

Uma instancia de relação é uma tabela, onde o esquema de relação descreve os cabeçalhos das colunas desta tabela.

Modelo Relacional

Inicialmente descreve-se o esquema de relação e depois a instancia de relação.

No esquema, especifica-se o nome da relação, o nome de cada campo (atributos) além do domínio de cada campo;

Os domínios descritos pelo nome de domínio, e possuem um conjunto de valores associados.

Modelo Relacional

Utilizando o exemplo do esquema Aluno utilizado na aula 1, temos que o domínio do atributo nome é uma string, onde o conjunto de valores associados são todas as strings de caracteres, assim como para a idade temos o conjunto dos números inteiros.

Aluno(id-aluno: String, nome: String, login: string, idade: integer, média: real)

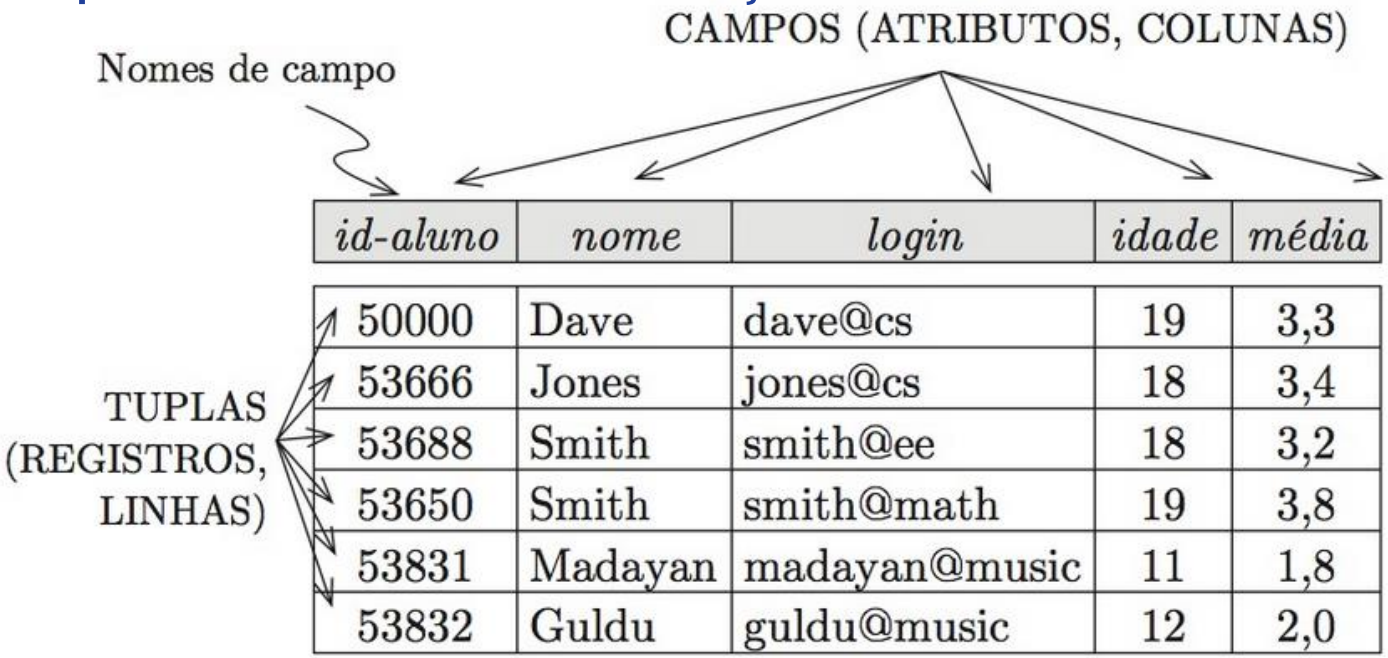
Modelo Relacional

As instâncias de uma relação, são um conjunto de tuplas, também conhecidas como registros, onde cada tupla tem o mesmo número de campos que o esquema de relação.

A instancia de relação pode ser vista como uma tabela, onde cada tupla é uma linha, e todas as linhas possuem o mesmo número de campos.

Modelo Relacional

Exemplo de instância da relação Alunos



id-aluno	*nome*	*login*	*idade*	*média*
50000	Dave	dave@cs	19	3,3
53666	Jones	jones@cs	18	3,4
53688	Smith	smith@ee	18	3,2
53650	Smith	smith@math	19	3,8
53831	Madayan	madayan@music	11	1,8
53832	Guldu	guldu@music	12	2,0

Modelo Relacional

O número de campos que uma relação possui é chamado de **grau** ou **aridade**;

Assim como á quantidade de tuplas de uma instancia de relação é chamada de **cardinalidade**;

Considerando a instancia Alunos apresentada, pode-se dizer que possui grau 5 e cardinalidade 6.

Criando e modificando relações usando SQL

Como visto anteriormente, os bancos de dados podem ser descritos utilizando uma DDL, a DDL utilizada será a linguagem SQL (*Structured Query Language*) que possui sua definição na ISO/IEC 9075.

A versão utilizada nos exemplos é a SQL:1999 (SQL3)

Ferramenta online para testar os comandos SQL: http://www.sqlfiddle.com

Criando e modificando relações usando SQL

Maquina virtual para teste offline “BancoDeDados” (link da imagem disponível no Moodle). Usuário: usuario Senha: senha

Criando e modificando relações usando SQL

Para criar uma nova tabela é utilizado o comando **CREATE TABLE**.
Exemplo para a relação Alunos:

```
CREATE TABLE Alunos (   id_aluno char(20),
                        nome char(30),
                        login char(20),
                        idade integer,
                        media real);
```

Criando e modificando relações usando SQL

A inserção das tuplas é realizada pelo comando **INSERT**.
Exemplo de inserção:

INSERT INTO Alunos(id_aluno, nome, login, idade, media) **VALUES** (53688, “Smith”, “smith@ee”,18,6);

Criando e modificando relações usando SQL

Para remover uma tupla utiliza-se o comando **DELETE** com o auxilio do comando **WHERE**, o qual verifica a condição para a exclusão.

Exemplo:

DELETE FROM Alunos **WHERE** Alunos.nome = “Smith”;

Criando e modificando relações usando SQL

As modificações são realizadas com o comando **UPDATE**. Por exemplo para alterar a média de um determinado aluno:

UPDATE Alunos **SET** Alunos.media=8 **WHERE** Alunos.id_aluno = 53688;

Criando e modificando relações usando SQL

Para consultar os dados inseridos no SGBD utiliza-se o comando SELECT:

Exemplo:
SELECT * FROM Alunos;

Obs: * é o equivalente a todos os campos, pode ser substituído pelo nome(s) do campo que se deseja consultar.
O comando SELECT será visto com mais detalhes.

Exercício

Crie uma tabela contendo as informações de Alunos e:

- Insira pelo menos 3 registros;
- Apague o registro de um aluno;
- Atualize a idade de um aluno;
- Consulte as informações sobre os alunos;

Exercício

Crie uma tabela para cada esquema abaixo:

Alunos(*id-aluno:* `string`, *nome*: `string`, *login:* `string`, *idade:* `integer`, *média:* `real`)
Professores(*id-prof:* `string`, *nomep:* `string`, *sal:* `real`)
Cursos(*id-curso:* `string`, *nomec:* `string`, *créditos:* `integer`)
Salas(*num-sala:* `integer`, *endereço:* `string`, *capacidade:* `integer`)
Matriculado(*id-aluno:* `string`, *id-curso:* `string`, *nota:* `string`)
Ministra(*id-prof:* `string`, *id-curso:* `string`)
Aula(*id-curso:* `string`, *num-sala:* `integer`, *horário:* `string`)

Restrições de Integridade sobre relações

Para garantir que somente sejam inseridas informações válidas no banco de dados, o SGBD utiliza **Restrições de Integridade (RI)**;

As RIs são especificadas quando o esquema de banco de dados está sendo definido, e o SGBD valida estas restrições para cada instância.

Restrições de chave

A **restrição de chave** é definida como uma declaração de que um determinado subconjunto mínimo dos campos de uma relação é um identificador único para uma tupla.

O conjunto de campos que identifica a tupla de acordo com a restrição de chave é chamado de **chave candidata** da relação, que normalmente é abreviada para somente **chave**.

Restrições de chave

No exemplo da relação Alunos, dois alunos não podem ter o mesmo identificador, assim como duas pessoas não podem ter o mesmo número de RG ou CPF, neste caso o campo id_aluno é a chave candidata.

A escolha das chaves deve ser realizada com bastante cuidado para evitar problemas com dados válidos, como por exemplo dois alunos com o mesmo nome.

Restrições de chave

Caso o conjunto de campos contenha um campo que é uma chave, este conjunto é chamado de **superchave**, como por exemplo o conjunto {id_aluno,nome}, pois id_aluno já é uma chave.

As relações podem conter diversas chaves candidatas, como por exemplo o conjunto {login, idade} nos diz que podem ter dois alunos com o mesmo login ou com a mesma idade, mas não ambos.

Restrições de chave

Apesar de poder possuir diversas chaves candidatas é necessário que o projetista do banco de dados defina uma **chave primária**, que será a mais utilizada para se referenciar a uma tupla.

O SGBD criará os índices para que a consulta utilizando a chave primaria seja otimizada.

Restrições de chave

Para especificar as restrições de chave em SQL utiliza-se a restrição **UNIQUE**, onde no máximo uma das chaves candidatas será declarada como chave primária utilizando a restrição **PRIMARY KEY**.

Exemplo: CREATE TABLE Alunos(id_aluno char(20), nome char(30), login char(20), idade integer, media real, **UNIQUE** (nome, idade), **CONSTRAINT** Chave **PRIMARY KEY**(id_aluno));

Restrições de chave estrangeira

Em alguns casos, as informações armazenadas em uma relação estão diretamente ligadas a outras informações que estão em outra relação. Caso uma destas relações sofrer alterações, a outra deve ser verificada para garantir a consistência dos dados.

Isto pode ser obtido utilizando uma RI, no caso o que é chamado de **chave estrangeira**.

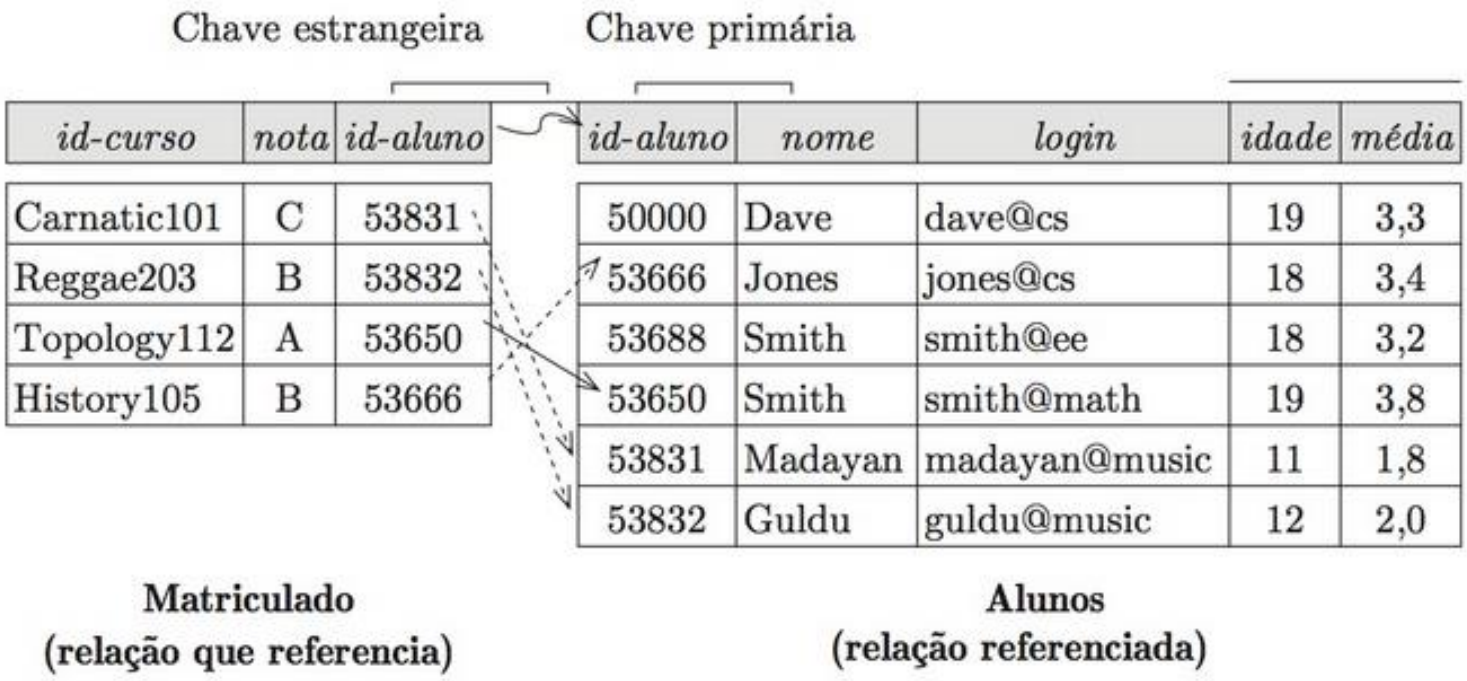
Restrições de chave estrangeira

Por exemplo no caso da Universidade, além da relação Alunos, podemos ter uma outra relação

Matriculado(id_aluno: string, id_curso: string, nota: string)

Neste caso somente alunos existentes podem estar matriculados em um curso, para isso existirá um vínculo com a relação Alunos através do atributo id_aluno.

Restrições de chave estrangeira



id-curso	*nota*	*id-aluno*
Carnatic101	C	53831
Reggae203	B	53832
Topology112	A	53650
History105	B	53666

Matriculado
(relação que referencia)

id-aluno	*nome*	*login*	*idade*	*média*
50000	Dave	dave@cs	19	3,3
53666	Jones	jones@cs	18	3,4
53688	Smith	smith@ee	18	3,2
53650	Smith	smith@math	19	3,8
53831	Madayan	madayan@music	11	1,8
53832	Guldu	guldu@music	12	2,0

Alunos
(relação referenciada)

Restrições de chave estrangeira

Neste exemplo se tentarmos inserir a tupla {12345, Art104, A} o SGBD deve rejeitar a inclusão pois não existe nenhum registro em Alunos com o id_aluno igual a 12345.

Do mesmo modo caso o aluno seja removido da relação Alunos o SGBD deve identificar que o aluno ainda está vinculado a uma tupla da relação Matriculado e evitar a inconsistência de dados.

Restrições de chave estrangeira

Para especificar restrições de chave estrangeira em SQL utiliza-se a restrição **FOREIGN KEY** juntamente com o comando **REFERENCES**.
Exemplo da relação Matriculado:

CREATE TABLE Matriculado (id_aluno char(20), id_curso char(20), nota char(10), **PRIMARY KEY** (id_aluno, id_curso), **FOREIGN KEY** (id_aluno) **REFERENCES** Alunos(id_aluno));

Restrições gerais

As Ris vistas até agora recebem atenção especial na maioria dos sistemas, no entanto as vezes é necessário especificar outras restrições mais gerais.

Um exemplo de restrição mais geral é exigir que a idade dos alunos esteja contida dentro de um determinado intervalo de valores. O SGBD deve rejeitar a inserção de dados que violem esta restrição. Esta restrição é chamada de restrição de domínio estendida.

Restrições gerais

Os SGBDs atuais suportam restrições mais gerais, as quais são divididas em dois grupos:

- **Restrições de tabela:** São associadas a uma única tabela, e toda vez que a tabela for modificada será verificada;

- **Assertivas:** São associadas a um conjunto de tabelas, e quando uma destas for alterada, a restrição será verificada.

VERIFICANDO RESTRIÇÕES DE INTEGRIDADE

Verificando restrições de integridade

O comportamento padrão do SGBD ao encontrar uma violação de integridade é não permitira a alteração.

No caso de uma inserção onde seja duplicado um valor de chave primária este comportamento é aceitável, mas no caso de uma atualização ou exclusão de um registro pode-se esperar um comportamento diferente.

Verificando restrições de integridade

Utilizando o exemplo das relações Alunos e Matriculado, podemos verificar algumas opções de como tratar as violações de chave estrangeira:

1 - O que fazer se uma tupla de Matriculado é inserida com um valor de id_aluno que não existe na tabela Alunos?
Rejeitar a inserção é o comportamento esperado.

Verificando restrições de integridade

2 – O que fazer se uma linha de Alunos for exluida?

- A) Excluir todas as linhas de Matriculados que estão vinculadas a aquele aluno;
- B) Rejeitar a exclusão da linha de Alunos enquanto ainda estiver vinculado a uma linha de Matriculado;
- C) Alterar o valor da coluna id_aluno de matriculado para um valor existente que represente “sem aluno”;

Verificando restrições de integridade

3 – E se o valor da chave primária de uma tupla de Alunos for atualizado? Neste caso as opções são semelhantes ao caso 2.

Verificando restrições de integridade

A linguagem SQL nos permite escolher o comportamento a ser adotado no caso de violação de chave em um comando **DELETE** ou **UPDATE.**

Exemplo:

CREATE TABLE Matriculado (id_aluno char(20), id_curso char(20), nota char(10), **PRIMARY KEY** (id_aluno, id_curso), **FOREIGN KEY** (id_aluno) **REFERENCES** Alunos(id_aluno) **ON DELETE CASCADE ON UPDATE NO ACTION**);

Verificando restrições de integridade

Neste exemplo o comando **ON DELETE** indica o que será feito caso uma tupla de Matriculado seja deletada, neste caso foi escolhido o comportamento **CASCADE** que excluirá a tupla de Alunos que está vinculada com a linha removida de Matriculados.

Já no caso de atualização da tupla (**ON UPDATE**) foi escolhido o comportamento **NO ACTION** que rejeitará a atualização (que é o comportamento padrão e poderia ser omitido neste caso).

Transações e restrições

Em alguns casos existirão restrições que necessitariam que os dados fossem inseridos exatamente ao mesmo tempo para que não fossem violadas.

Para que seja possível realizar a inserção dos dados pode-se colocar a verificação das restrições no modo adiado (**DEFERRED**) que normalmente são realizadas no modo imediato (**IMMEDIATE**).

Transações e restrições

Este modo adiado permite uma inconsistência temporária no banco até que a transação seja efetivada (commit).

Exemplo:
SET CONSTRAINT nome_restrição **DEFERRED**

Consultando dados relacionais

Consultando dados relacionais

As consultas de banco de dados relacionais, ou simplesmente chamadas de **consultas** são perguntas feitas ao SGBD onde a resposta será uma nova relação.

Para realizar estas consultas são utilizadas as linguagens de consulta como o SQL.

Consultando dados relacionais

As consultas são realizadas utilizado o comando **SELECT**, considerando a relação alunos, podemos consultar todos os campos de todos os alunos com idade inferior a 19 anos;

SELECT * FROM Alunos A **WHERE** A.idade < 19;

Consultando dados relacionais

É possível especificar os campos desejados substituindo o * pelo nome do campo.
Exemplo, selecionando apenas o nome e o login dos alunos com menos de 19 anos:

SELECT A.nome, A.login **FROM** Alunos A **WHERE** A.idade < 19;

Consultando dados relacionais

As informações das relações também podem ser combinadas em uma única consulta.
Por exemplo, para obter o nome dos alunos com nota A e o curso no qual estes obtiveram esta nota:

SELECT A.nome, M.id_curso **FROM** Alunos A, Matriculado M **WHERE** A.id_aluno = M.id_aluno AND M.nota="A";

OBRIGADO.

UNISINOS

DESAFIE
O AMANHÃ.