Sempre que temos um SBGD de múltiplos níveis, seu catálogo deve ser expandido para incluir informações sobre como mapear solicitações e dados entre os diversos níveis. O SGBD usa software adicional para realizar esses mapeamentos, recorrendo à informação de mapeamento no catálogo. A independência de dados ocorre porque, quando o esquema é alterado em algum nível, o esquema no próximo nível mais alto permanece inalterado; somente o *mapeamento* entre os dois níveis é alterado. Logo, os programas de aplicação que fazem referência ao esquema de nível mais alto não precisam ser alterados.

5. Modelo de dados relacional

Esta seção aborda os bancos de dados relacionais. O modelo de dados relacional foi introduzido inicialmente por Ted Codd, da IBM Research, em 1970, em um artigo clássico (Codd, 1970), que atraiu atenção imediata devido a sua simplicidade e base matemática. O modelo usa o conceito de *relação matemática* – que se parece com uma tabela de valores – como seu bloco de montagem básico, e sua base teórica reside em uma teoria de conjunto e lógica de predicado de primeira ordem. Nesta seção, discutiremos as características básicas do modelo e suas restrições.

Por causa da importância do modelo relacional, toda a disciplina Banco de Dados I é dedicada a esse modelo e alguma das linguagens associadas a ele. No Capítulo 2, descreveremos a linguagem de consulta SQL, que é o *padrão* para SGBDs relacionais comerciais. Outros aspectos do modelo relacional são apresentados em outras partes da Mídia Integrada Convergente. O Capítulo 3 abordará as estruturas de dados do modelo relacional para construções do modelo ER, e apresentará algoritmos para projetar um esquema de banco de dados relacional mapeando um esquema conceitual no modelo ER para uma representação relacional. Esses mapeamentos são incorporados em muitas ferramentas de projeto de banco de dados e CASE[7]. O Capítulo 4 apresentará outro aspecto do modelo relacional, a saber, as restrições formais das dependências funcionais e multivaloradas. Essas

---

tempo de resposta de uma consulta sem que seja preciso alterá-la, embora a consulta seja executada com mais eficiência pelo SGBD ao utilizar o novo caminho de acesso.

[7] CASE significa Computer-Aided Software Engineering (engenharia de software auxiliada por computador).

dependências são usadas para desenvolver uma teoria de projeto de banco de dados relacional baseada no conceito conhecido como *normalização*.

Nesta seção, concentramo-nos em descrever os princípios básicos do modelo de dados relacional. Começamos definindo os conceitos de modelagem e a notação do modelo relacional na Seção 5.1. A Seção 5.2 é dedicada a uma discussão das restrições relacionais que são consideradas uma parte importante do modelo relacional e automaticamente impostas na maioria dos SGBDs relacionais.

5.1 Conceitos do modelo relacional

O modelo relacional representa o banco de dados como uma coleção de *relações.* Informalmente, cada relação é semelhante a uma tabela de valores em que cada linha na tabela representa uma coleção de valores de dados relacionados. Uma linha representa um fato que normalmente corresponde a uma entidade [8] ou relacionamento do mundo real. Os nomes da tabela e de coluna são usados para ajudar a interpretar o significado dos valores em cada linha. Por exemplo, a terceira tabela da Figura 2 é chamada de CLIENTE porque cada linha representa fatos sobre uma entidade particular de cliente. Os nomes de coluna – Cpf, Nome, Sexo e Endereco – especificam como interpretar os valores de dados em cada linha, com base na coluna em que cada valor se encontra. Todos os valores em uma coluna são do mesmo tipo de dado.

Na terminologia formal do modelo relacional, uma linha é chamada **tupla**, um cabeçalho da coluna é chamado de **atributo** e a tabela é chamada de **relação**. O tipo de dado que descreve os valores que podem aparecer em cada coluna é representado por um **domínio** de valores possíveis. Por exemplo, o tipo de dado para o domínio Numeros_telefone_nacional pode ser declarado como uma sequência de caracteres na forma (*dd*)*dddd-dddd*, onde cada *d* é um dígito numérico (decimal) e os dois primeiros dígitos formam um código de área de telefone válido. O tipo de dado para Idades_funcionario pode ser declarado como um inteiro entre 15 e 80. Para Nomes_departamento_academico, o tipo de dado é o conjunto de todas as cadeias de caracteres que representam nomes de departamento válidos. Um domínio, portanto, recebe um nome, tipo de dado e formato.

[8] Uma entidade é qualquer objeto sobre o qual queremos registrar informações (DATE, 2004).

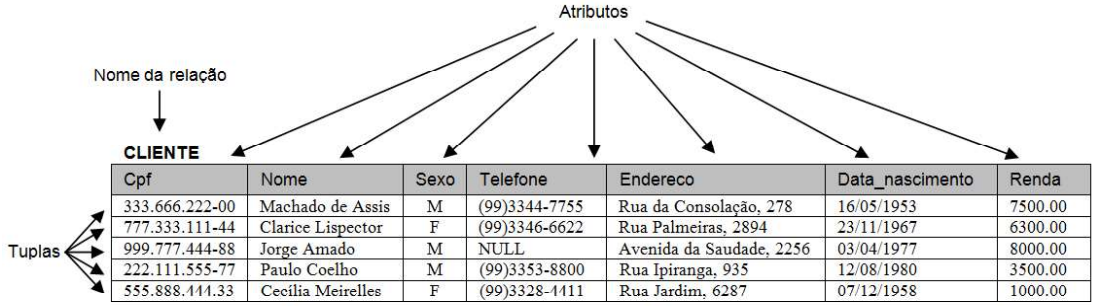
Um **esquema relacional** *R*, indicado por $R(A_1, A_2,\ldots, A_n,)$, é composto de um nome de relação R e uma lista de atributos, $A_1, A_2,\ldots, A_n$. Cada **atributo** A_i é o nome de um papel desempenhado por algum domínio *D* no esquema de relação *R*. *D* é chamado de **domínio** de A_i, e indicado por dom($\boldsymbol{A_i}$). Um esquema de relação é usado para *descrever* uma relação; *R* é chamado de **nome** dessa relação. O **grau** (ou **aridade**) de uma relação é o número de atributos *n* desse esquema de relação (ELMASRI & NAVATHE, 2011).

Uma relação de grau sete, que armazena informações sobre cliente, teria sete atributos descrevendo cada cliente, da seguinte forma:

CLIENTE(Cpf, Nome, Sexo, Telefone, Endereco, Data_nascimento, Renda)

A Figura 5 mostra um exemplo de uma relação CLIENTE, que corresponde ao esquema CLIENTE já especificado.

Figura 5 – Os atributos e tuplas de uma relação CLIENTE



Cpf	Nome	Sexo	Telefone	Endereco	Data_nascimento	Renda
333.666.222-00	Machado de Assis	M	(99)3344-7755	Rua da Consolação, 278	16/05/1953	7500.00
777.333.111-44	Clarice Lispector	F	(99)3346-6622	Rua Palmeiras, 2894	23/11/1967	6300.00
999.777.444-88	Jorge Amado	M	NULL	Avenida da Saudade, 2256	03/04/1977	8000.00
222.111.555-77	Paulo Coelho	M	(99)3353-8800	Rua Ipiranga, 935	12/08/1980	3500.00
555.888.444.33	Cecília Meirelles	F	(99)3328-4411	Rua Jardim, 6287	07/12/1958	1000.00

Fonte: o autor

Cada tupla na relação representa uma entidade de cliente em particular (ou objeto). Apresentamos a relação como uma tabela, onde cada tupla aparece como uma *linha* e cada atributo corresponde a um *cabeçalho de coluna*, indicando um papel ou interpretação dos valores nesta coluna. *Valores NULL* representam atributos cujos valores são desconhecidos ou não existem para alguma tupla individual de CLIENTE.

Em geral, à medida que o estado do mundo real muda, também muda o **estado de relação** – reflete apenas as tuplas válidas que representam um estado em particular do mundo real –, sendo transformada em outro estado de relação. Contudo o esquema de uma relação é relativamente estático e muda com pouca frequência – por exemplo, como resultado da inclusão de um atributo para representar novas informações que não estavam armazenadas originalmente na relação.

5.2 Restrições em modelo relacional

Até aqui, discutimos as características de relações isoladas. No banco de dados relacional, normalmente haverá muitas relações, e as tuplas nessas relações costumam estar relacionadas de várias maneiras. O estado do banco de dados inteiro corresponderá aos estados de todas as suas relações em determinado ponto no tempo. Em geral, existem muitas **restrições** (ou *constraints*) sobre os valores reais em estado no banco de dados. Essas restrições são derivadas das regras no minimundo que o banco de dados representa.

Nesta seção, discutiremos as diversas restrições sobre os dados que podem ser especificadas em um banco de dados relacional na forma de restrições. Elmasri & Navathe (2011) classificam as restrições nos bancos de dados em três categorias principais:

1. Restrições que são inerentes no modelo de dados. Chamamos estas de **restrições inerentes baseadas no modelo ou restrições implícitas.** Por exemplo, a restrição de que uma relação não pode ter tuplas duplicadas é uma restrição inerente.
2. Restrições que podem ser expressas diretamente nos esquemas do modelo de dados, em geral especificando-as na DDL (linguagem de definição de dados; na qual a veremos no Capítulo 2). Chamamos estas de **restrições baseadas em esquema** ou **restrições explícitas**.
3. Restrições que *não podem* ser expressas diretamente nos esquemas do modelo de dados, e, portanto, devem ser expressas e impostas pelos programas de aplicação. Chamamos estas de **restrições baseadas na aplicação** ou **restrições semânticas** ou **regras de negócios**.

As restrições que discutimos nesta seção são da segunda categoria, a saber, restrições que podem ser expressas no esquema do modelo relacional por meio da DDL. As restrições baseadas em esquema incluem restrições de domínio, restrições de chave, restrições sobre NULLs, restrições de integridade de entidade e restrições de integridade referencial.

5.2.1 Restrições de domínio

As restrições de domínio especificam que, dentro de cada tupla, o valor de cada atributo A deve ser um valor indivisível do domínio dom(A). Os tipos de dados associados ao domínio normalmente incluem os tipos de dados numéricos padrão

para inteiros (como *short integer*, *integer* e *long integer*) e números reais (*float* e *double*). Caracteres, booleanos, cadeia de caracteres de tamanho fixo e cadeia de caracteres de tamanho variável também estão disponíveis, assim como data, hora, marcador de tempo, moeda ou outros tipos de dados especiais. Outros domínios possíveis podem ser descritos por um subintervalo dos valores de um tipo de dados ou como um tipo de dado enumerado, em que todos os valores possíveis são explicitamente listados.

5.2.2 Restrições de chave

No modelo relacional formal, uma *relação* é definida como um *conjunto de tuplas*. Por definição, todos os elementos de um conjunto são distintos; logo, todas as tuplas em uma relação também precisam ser distintas. Isso significa que duas tuplas não podem ter a mesma combinação de valores para todos os seus atributos. Normalmente, existem outros **subconjuntos de atributos** de um esquema de relação *R* com a propriedade duas tuplas em qualquer estado de relação *r* de *R* não deverão ter a mesma combinação de valores para esses atributos. Suponha que indiquemos um subconjunto de atributos desse tipo como SCh; então, para duas tuplas *distintas* quaisquer t_1 e t_2 em um estado de relação *r* de *R*, temos a restrição de que:

$$t_1[\text{SCh}] \neq t_2[\text{SCh}]$$

Qualquer conjunto de atributos SCh desse tipo é chamado de **superchave** do esquema de ralação *R*. Uma superchave SCh especifica uma *restrição de exclusividade* de que duas tuplas distintas em qualquer estado *r* de *R* não podem ter o mesmo valor de SCh. Cada relação tem pelo menos uma superchave padrão – o conjunto de todos os seus atributos. Contudo, uma superchave pode ter atributos redundantes, de modo que um conceito mais útil é o de uma *chave*, que não tem redundância. Uma **chave** *Ch* de um esquema de relação *R* é uma superchave de *R* com a propriedade adicional de que a remoção de qualquer atributo *A* de *Ch* deixa um conjunto de atributos *Ch*' que não é mais uma superchave de *R*. Logo, uma chave satisfaz duas propriedades:

1. Duas tuplas distintas em qualquer estado da relação não podem ter valores idênticos para (todos) os atributos na chave. Essa primeira propriedade também se aplica a uma superchave.
2. Ela é uma *superchave mínima* – ou seja, uma superchave da qual não podemos remover nenhum atributo e ainda mantermos uma restrição de

exclusividade na condição 1. Essa propriedade não é exigida por uma superchave.

Embora a primeira propriedade se aplique a chaves e superchaves, a segunda propriedade é exigida apenas para chaves. Assim, uma chave também é uma superchave, mas não o contrário. Considere a relação CLIENTE da Figura 5. O conjunto de atributos {Cpf} é uma chave de CLIENTE porque duas tuplas de cliente não podem ter o mesmo valor para Cpf.[9] Qualquer conjunto de atributos que inclua Cpf – por exemplo, {Cpf, Nome, Renda} – é uma superchave. No entanto a superchave {Cpf, Nome, Renda} não é uma chave de cliente, pois remover Nome ou Renda, ou ambos, do conjunto ainda nos deixa com uma superchave. Em geral, qualquer superchave formada com base em um único atributo também é uma chave. Uma chave com múltiplos atributos precisa exigir que *todos* os seus atributos juntos tenham uma propriedade de exclusividade.

O valor de um atributo de chave pode ser usado para identificar exclusivamente cada tupla na relação. Por exemplo, o valor de Cpf 333.666.222-00 identifica exclusivamente a tupla correspondente a Machado de Assis na relação CLIENTE. Observe que um conjunto de atributos constituindo uma chave é uma propriedade do esquema de relação; essa é uma restrição que deve ser mantida sobre *cada* estado de relação válido do esquema. Uma chave é determinada com base no significado dos atributos, e a propriedade é *invariável no tempo*: ela precisa permanecer verdadeira quando inserimos novas tuplas na relação. Por exemplo, não podemos e não devemos designar o atributo Nome da relação CLIENTE da Figura 5 como uma chave porque é possível que dois clientes com nomes idênticos existam em algum ponto em um estado válido.

Em geral, um esquema de relação pode ter mais de uma chave. Nesse caso, cada uma das chaves é chamada de **chave candidata.** Por exemplo, a relação ALUNO da Figura 6 tem duas chaves candidatas: Rga e Cpf. É comum designar uma das chaves candidatas como **chave primária** da relação. Essa é a chave cujos valores são usados para *identificar* tuplas na relação. Usamos a convenção de que os atributos que formam a chave primária de um esquema de relação são sublinhados, como mostra a Figura 6.

[9] Observe que Cpf também é uma superchave.

Figura 6 – A relação ALUNO, com duas chaves candidatas: Rga e Cpf

ALUNO

Rga	Nome	Cpf	Sexo	Telefone	Endereco	Data_nascimento
2889435	Fernando Braga	222.555.111-99	M	(99)3344-7755	Rua Aquidauana, 325	09/08/1996
8034201	Julia Benson	666.444.000-33	F	(99)3346-6622	Rua Major Capilé, 2350	14/12/1990
6323849	Roberto Passos	999.888.777-66	M	NULL	Avenida da Amizade, 560	29/07/1986
1702136	Paulo Lima	111.555.333-00	M	(99)3353-8800	Rua Ipiranga, 1925	10/10/1992
5466972	Alice Nogueira	444.666.222-33	F	(99)3328-4411	Rua das Camélias, 890	12/04/1985

Fonte: o autor

Observe que, quando um esquema de relação tem várias chaves candidatas, a escolha de uma para se tornar a chave primária é um tanto arbitrária; porém normalmente é melhor escolher uma chave primária com um único atributo ou um pequeno número de atributos. As outras chaves candidatas são designadas como chaves únicas (*unique keys*), e não são sublinhadas.

5.2.3 Restrições sobre valores NULL

Outra restrição sobre os atributos especifica se valores NULL são permitidos ou não. Por exemplo, se cada tupla de CLIENTE precisar ter um valor válido, diferente de NULL, para o atributo Nome, então Nome de CLIENTE é restrito a ser NOT NULL.

5.3.4 Restrições de integridade

A restrição de integridade de entidade afirma que nenhum valor de chave primária pode ser NULL. Isso porque o valor da chave primária é usado para identificar tuplas individuais em uma relação. Ter valores NULL para a chave primária implica que não podemos identificar algumas tuplas. Por exemplo, se duas ou mais tuplas tivessem NULL para suas chaves primárias, não conseguiríamos distingui-la ao tentar referenciá-las por outras relações.

5.3.5 Restrições de integridade referencial

A restrição de integridade referencial é especificada entre duas relações e usada para manter a consistência entre tuplas nas duas relações. Informalmente, a restrição de integridade referencial afirma que uma tupla em uma relação que referencia outra relação precisa se referir a uma *tupla existente* nessa relação. Por exemplo, na Figura 2, o atributo Cod_banco de AGENCIA fornece o código do banco que a agência pertence; logo, seu valor em cada tupla de AGENCIA precisa combinar com o valor de Codigo de alguma tupla na relação BANCO.

Para definir a integridade referencial de maneira mais formal, primeiro estabelecemos o conceito de uma *chave estrangeira* (*ChE – foreign key*). As

condições para uma chave estrangeira, dadas a seguir, especificam a restrição de integridade referencial entre os dois esquemas de relação R_1 e R_2. Um conjunto de atributos ChE no esquema de relação R_1 é uma **chave estrangeira** de R_1 que referencia a relação R_2 se ela satisfizer as seguintes regras:

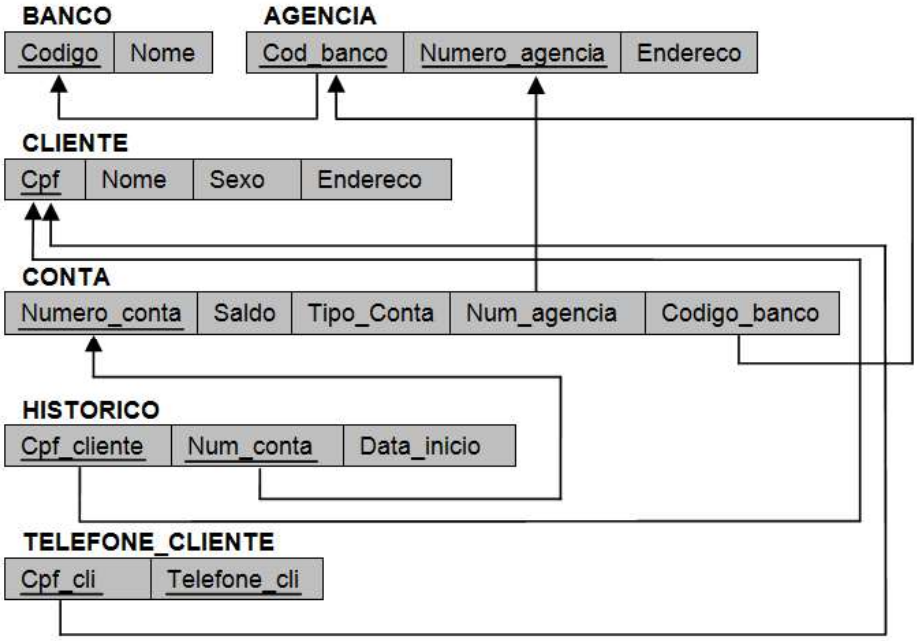
1. Os atributos em ChE têm o mesmo domínio (ou domínios) que os atributos de chave primária ChP de R_2; diz-se que os atributos ChE **referenciam** ou **referem-se à** relação R_2.
2. Um valor de ChE em uma tupla t_1 do estado atual $r_1(R_1)$ ocorre como um valor de ChE para alguma tupla t_2 no estado atual $r_2(R_2)$ *ou é NULL*. No primeiro caso, temos t_1[ChE] = t_2[ChP], e dizemos que a tupla t_1 **referencia** ou **refere-se à** tupla t_2.

Nessa definição, R_1 é chamada de **relação que referencia** e R_2 é a **relação referenciada.** Se essas condições se mantiverem, diz-se que é mantida uma **restrição de integridade referencial** de R_1 para R_2. Em um banco de dados de muitas relações, normalmente existem muitas restrições de integridade referencial.

Para especificar essas restrições, primeiro devemos ter um conhecimento claro do significado ou papel que cada atributo, ou conjunto de atributos, que fazem parte nos diversos esquemas de relação do banco de dados. As restrições de integridade referencial surgem com frequência dos *relacionamentos entre as entidades* representadas pelos esquemas de relação. Por exemplo, considere o banco de dados mostrado na Figura 2. Na relação AGENCIA, o atributo Cod_banco refere-se ao banco para o qual a agência pertence; portanto designamos Cod_banco para ser a chave estrangeira de AGENCIA que referencia a relação BANCO. Isso significa que um valor de Cod_banco em qualquer tupla t_1 da relação AGENCIA precisa combinar com um valor da chave primária de BANCO – o atributo Codigo – em alguma tupla t_2 da relação BANCO, ou o valor de *pode ser NULL* se for atribuído a um banco mais tarde. Por exemplo, na Figura 2, a tupla para a agência '4336' referencia a tupla para o banco 'Banco do Brasil', indicando que '4336' pertence a esse banco.

Podemos *exibir em forma de diagrama as restrições de integridade referencial*, desenhando um arco direcionado de cada chave estrangeira para a relação que ela referencia. Para ficar mais claro, a ponta da seta pode apontar para a chave primária da relação referenciada. A Figura 7 mostra o esquema da Figura 2 com as restrições de integridade referencial mostradas dessa maneira.

Figura 7 – Restrições de integridade referencial exibidas no esquema de banco de dados relacional BANCO



Fonte: o autor

Todas as restrições de integridade deverão ser especificadas no esquema de banco de dados relacional (ou seja, definidas como parte de sua definição) se quisermos impor essas restrições sobre os estados do banco de dados. Logo, a DDL inclui meios para especificar os diversos tipos de restrições de modo que o SGBD possa impô-las automaticamente. A maioria dos SGBDs relacionais admite restrições de chave, integridade de entidade e integridade referencial. Essas restrições são especificadas como uma parte da definição de dados na DDL.