Módulo 9

Banco de Dados

Lição 3

Projeto Lógico do Banco de Dados

Versão 1.0 - Fev/2009

Autor
Ma. Rowena C. Solamo

Equipe
Rommel Feria
Rick Hillegas
John Paul Petines

Necessidades para os Exercícios

Sistemas Operacionais Suportados

NetBeans IDE 5.5 para os seguintes sistemas operacionais:

- Microsoft Windows XP Profissional SP2 ou superior
- Mac OS X 10.4.5 ou superior
- Red Hat Fedora Core 3
- Solaris™ 10 Operating System (SPARC® e x86/x64 Platform Edition)

NetBeans Enterprise Pack, poderá ser executado nas seguintes plataformas:

- Microsoft Windows 2000 Profissional SP4
- Solaris™ 8 OS (SPARC e x86/x64 Platform Edition) e Solaris 9 OS (SPARC e x86/x64 Platform Edition)
- Várias outras distribuições Linux

Configuração Mínima de Hardware

Nota: IDE NetBeans com resolução de tela em 1024x768 pixel

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	500 MHz Intel Pentium III workstation ou equivalente	512 MB	850 MB
Linux	500 MHz Intel Pentium III workstation ou equivalente	512 MB	450 MB
Solaris OS (SPARC)	UltraSPARC II 450 MHz	512 MB	450 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Série 1.8 GHz	512 MB	450 MB
Mac OS X	PowerPC G4	512 MB	450 MB

Configuração Recomendada de Hardware

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	1 GB
Linux	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	850 MB
Solaris OS (SPARC)	UltraSPARC IIIi 1 GHz	1 GB	850 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Series 1.8 GHz	1 GB	850 MB
Mac OS X	PowerPC G5	1 GB	850 MB

Requerimentos de Software

NetBeans Enterprise Pack 5.5 executando sobre Java 2 Platform Standard Edition Development Kit 5.0 ou superior (JDK 5.0, versão 1.5.0_01 ou superior), contemplando a Java Runtime Environment, ferramentas de desenvolvimento para compilar, depurar, e executar aplicações escritas em linguagem Java. Sun Java System Application Server Platform Edition 9.

- Para **Solaris**, **Windows**, e **Linux**, os arquivos da JDK podem ser obtidos para sua plataforma em http://java.sun.com/j2se/1.5.0/download.html
- Para **Mac OS X**, Java 2 Plataform Standard Edition (J2SE) 5.0 Release 4, pode ser obtida diretamente da Apple's Developer Connection, no endereço: http://developer.apple.com/java (é necessário registrar o download da JDK).

Para mais informações: http://www.netbeans.org/community/releases/60/relnotes.htm

Java™ DB System Requirements

Java™ DB is supported on the Solaris, Linux and Windows operating systems and Sun Java 1.4 or later.

Colaboradores que auxiliaram no processo de tradução e revisão

Aécio Júnior
Alberto Ivo da Costa Vieira
Alexandre Mori
Alexis da Rocha Silva
Aline Sabbatini da Silva Alves
Allan Wojcik da Silva
Angelo de Oliveira
Aurélio Soares Neto
Bruno da Silva Bonfim
Carlos Fernando Gonçalves
Carlos Hilner Ferreira Costa
Daniel Noto Paiva
Daniel Wildt
Denis Mitsuo Nakasaki
Fábio Antonio Ferreira
Givailson de Souza Neves
Jacqueline Susann Barbosa
Jader de Carvalho Belarmino
João Vianney Barrozo Costa
José Francisco Baronio da Costa
Kleberth Bezerra Galvão dos Santos
Luiz Fernandes de Oliveira Junior
Maria Carolina Ferreira da Silva
Maricy Caregnato
Mauricio da Silva Marinho
Paulo Oliveira Sampaio Reis
Ronie Dotzlaw
Seire Pareja
Sergio Terzella
Thiago Magela Rodrigues Dias

Auxiliadores especiais

Revisão Geral do texto para os seguintes Países:

- **Brasil** – Tiago Flach
- **Guiné Bissau** – Alfredo Cá, Bunene Sisse e Buon Olossato Quebi – ONG Asas de Socorro

Coordenação do DFJUG

- **Daniel deOliveira** – JUGLeader responsável pelos acordos de parcerias
- **Luci Campos -** Idealizadora do DFJUG responsável pelo apoio social
- **Fernando Anselmo -** Coordenador responsável pelo processo de tradução e revisão, disponibilização dos materiais e inserção de novos módulos
- **Rodrigo Nunes -** Coordenador responsável pela parte multimídia
- **Sérgio Gomes Veloso -** Coordenador responsável pelo ambiente JEDI™ (Moodle)

Agradecimento Especial

John Paul Petines – Criador da Iniciativa JEDI™
Rommel Feria – Criador da Iniciativa JEDI™

1. Objetivos

Nesta lição discutiremos o projeto lógico do banco de dados. Embora existam muitos modelos de bancos dados que podem ser utilizados, esta lição focará no modelo de banco de dados relacional por dois motivos:

1. O modelo de dados relacional é amplamente utilizado nas aplicações de banco de dados
2. Certos princípios que se aplicam a outros modelos também se aplicam ao modelo relacional

Para entender o modelo de banco de dados relacional, certos conceitos matemáticos serão revistos. Além disso, o processo de transformar o modelo conceitual em modelo relacional será apresentado. Mais importante ainda, o conceito de normalização, integridade e restrições serão lecionadas.

Ao final desta lição, o estudante será capaz de:

- Através da lógica de dados, verificar como projeto do modelo conceitual (geralmente sob a forma de um Modelo Entidade-Relacionamento) é transformado em um projeto de banco de dados
- Compreender o modelo de dados relacionais, a álgebra relacional e o cálculo relacional
- Definir uma boa estrutura das relações através do conceito de normalização
- Conhecer as etapas na transformação de um MER em um conjunto de relações estruturadas

2. Projeto Lógico do Banco de Dados

É o processo de transformação do modelo de dados conceitual em um modelo lógico de dados que pode ser implementado em um sistema de gerenciamento de banco de dados. O resultado é um **modelo lógico de dados**, que é um projeto que está de acordo com os dados do modelo para uma classe de sistema de gerenciamento de banco de dados. Até agora existem quatro tipos de bases de dados lógicos. São eles:

1. **Modelo Hierárquico**. Registros estão organizados em uma estrutura que lembra uma árvore de cabeça para baixo. Os termos pai e filho são muitas vezes utilizados na descrição do modelo. Uma propriedade importante é que um registro filho pode estar relacionado a um único progenitor. A figura mostra um exemplo.

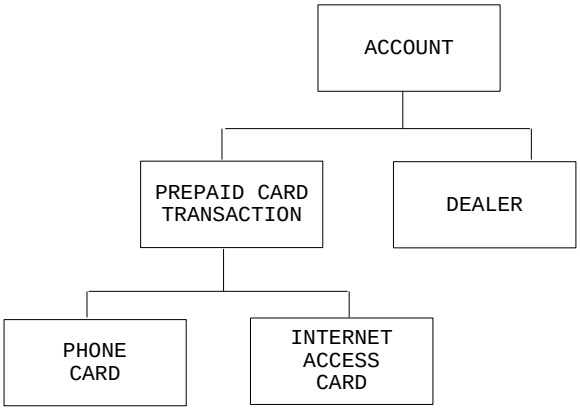


Figura 1: Modelo Hierárquico

Neste exemplo, ACCOUNT é o pai dos registros PREPAID CARD TRANSACTION e DEALER, enquanto PHONE CARD e INTERNET ACCESS CARD têm PREPAID CARD TRANSACTION como seu registro pai.

2. **Modelo de Rede**. É semelhante ao modelo hierárquico exceto que não há nenhuma distinção entre registros pai e filho. Qualquer tipo de registro pode ser associado a um número arbitrário de diferentes tipos de registro. Foi desenvolvido principalmente para superar a limitação no âmbito de aplicação do modelo hierárquico. A figura a seguir mostra o modelo da rede para a transação com cartão.

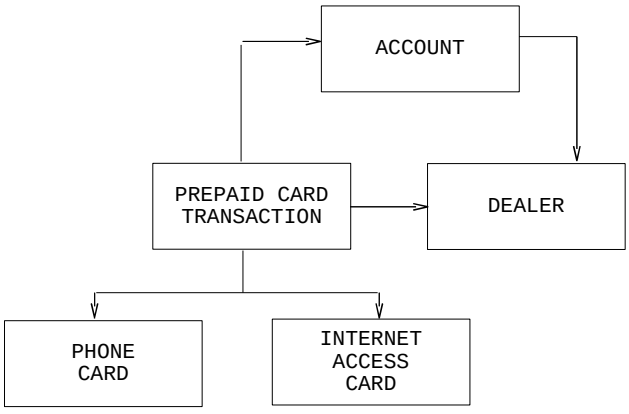


Figure 2: Modelo da Rede

Neste exemplo, PREPAID CARD TRANSACTION tem uma conta pertencente a um DEALER. Um PREPAID CARD TRANSACTION envolve tanto um PHONE CARD ou INTERNET ACCESS CARD.

3. **Modelo Relacional**. Os dados são representados sob a forma de entidades com linhas e colunas. Não há dados físicos representativos das associações entre as entidades, em vez disso as associações são representadas por valores lógicos que estão armazenados no interior nas colunas.

ACCOUNT

CellPhoneNo	PIN	Balance	Limit	Status	Type
09192345678	1234	$500,00	$2.500,00	ACT	2
09174561234	2345	$100,00	$2.000,00	ACT	2
09205467234	4523	$25.000,00	$300.000,00	ACT	1
09165647342	7812	$30.000,00	$300.000,00	ACT	1

STATUS

CODE	DESCRIPTION
ACT	Active Account
BAR	Barred Account
TER	Terminated Account

TYPE

CODE	DESCRIPTION
1	Dealer Account
2	Direct Reseller

PHONECARDTRANSACTION

CELLPHONENO	CARDNO	LOADDATE	RECIPIENT
09205467234	2346253782	DEC-24-2006	9223456173
09205467234	8736237634	DEC-24-2006	9178746345

Figura 3: Modelo Relacional

A Figura 3 representa a transação *Prepaid Card* como um modelo relacional. Neste exemplo, a entidade ACCOUNT possui chaves estrangeiras STATUS e TYPE cujos valores são chaves primárias de STATUS (coluna CODE) e TYPE (coluna CODE). Isto significa que STATUS e TYPE não podem ter valores que não estejam presente em STATUS e TYPE nas entidades respectivas.

4. **Modelo Orientado a Objeto**. Atributos de dados e métodos que operam esses atributos são encapsulados em estruturas chamadas objetos. Figura 4 mostra o modelo de objeto para a transação *Prepaid Card*. Seis objetos são definidos: DEALER, ACCOUNT, PREPAID CARD TRANSACTION, PREPAID CARD, PHONE CARD e INTERNET ACCESS CARD.

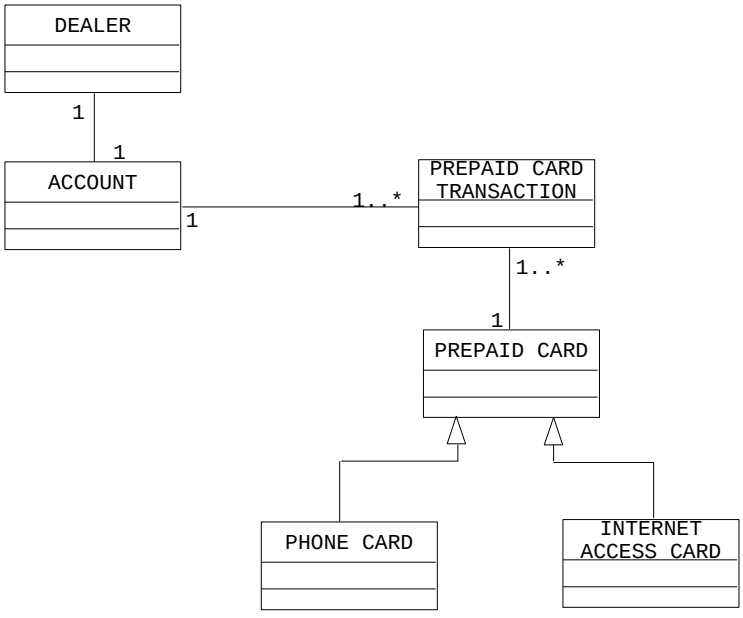


Figura 4: Modelo de Classes

3. Modelo de Dados Relacionais

O modelo de dados relacional foi introduzido pela primeira vez por E.F. Codd, um matemático especializado, em 1970. Baseou-se no conceito matemático de relação, que é fisicamente representado como uma entidade e utiliza terminologias da matemática, especificamente, ao definir a teoria e predicados lógicos. Nesta seção, terminologias e conceitos estruturais do modelo relacional serão discutidos.

Existem três componentes do modelo relacional. São eles:

1. A estrutura de dados é uma coleção de objetos ou relações que armazenam dados. Os dados são organizados sob a forma de entidades ou relações
2. A manipulação de dados consiste em operações (tais como os incorporados na linguagem SQL) que são usados para manipular os dados armazenados nas relações. Elas são baseadas no conjunto dos operadores que atuam sobre as relações
3. A integridade dos dados inclui facilidades para especificar as normas que mantém a integridade dos dados. É utilizado para precisão e a coerência

Uma base de dados é uma coleção de relações ou entidades. As relações no interior da base de dados deverão ser devidamente estruturadas. A adequação é conhecida como normalização, que será discutida mais tarde, neste capítulo.

A relação é uma tabela de dados bidimensional. É constituída por um conjunto de linhas e colunas.

Uma linha ou tupla representa todos os dados necessários para uma determinada relação. Corresponde a um único registro. Na relação representa um grupo horizontal de células.

Uma coluna corresponde a um atributo de uma relação. A relação representa um grupo vertical de células. A Figura 5 mostra um exemplo de uma relação. Cada registro, ou linha, é identificado por uma chave primária, que é uma coluna ou atributo que identifica exclusivamente registros, ou seja, o valor desta coluna deve ser único e não nulo na relação. A chave composta é a chave primária que é constituída por mais de um atributo. Uma chave estrangeira é um atributo de uma relação de um banco de dados que serve como uma chave primária de uma outra função na mesma base de dados.

ACCOUNT

CellPhoneNo	PIN	Balance	Limit	Status	Type
09192345678	1234	$500,00	$2.500,00	ACT	2
09174561234	2345	$100,00	$2.000,00	ACT	2
09205467234	4523	$25.000,00	$300.000,00	ACT	1
09165647342	7812	$30.000,00	$300.000,00	ACT	1

Figura 5: Relação de Contas

Neste exemplo, o nome da relação é ACCOUNT. As colunas de ACCOUNT são CellPhoneNo, PIN, Balance, Limit, Status e Type. Um exemplo de uma linha ou tupla é constituído pelos seguintes valores: (09192345678, 1234, $500.00, $2.500,00, ACT, 2). A chave primária da entidade é CellPhoneNo, e está sublinhada com um traço sólido. As chaves estrangeiras da relação são Status e Type, e estão sublinhadas com um traço pontilhado.

A relação, ou tabela, pode ser expressa pela seguinte abreviação, ou notação:

ACCOUNT(CellPhoneNo, PIN, Balance, Limit, Status, Type)

Um domínio é o conjunto de valores admissíveis para um ou mais atributos. São extremamente poderosos como característica do modelo relacional uma vez que cada atributo em um banco de dados relacional é definido em um domínio. Permitem aos usuários definir em um local central o significado das colunas e a fonte de valores que pode conter e, assim, fornecer mais informação para o sistema na execução de operações relacionais, e as operações que são semanticamente incorretas podem ser evitadas. Como exemplo, não faz sentido comparar o número de telefone

móvel com o último nome do proprietário da conta, apesar de terem o mesmo domínio, ou seja, valores. A Figura 6 mostra um exemplo para o domínio da relação ACCOUNT.

Coluna	Domínio	Significado	Definição
CellPhoneNo	Número do Telefone	Conjunto de todos os números móveis válidos nas Filipinas	Número: tamanho 11
PIN	Número de Identificação Pessoal	Conjunto de todos os quatro dígitos do número de identificação	Número: tamanho 4
Balance	Saldo remanescente	Conjunto que representa valor de saldo de conta que deverá ser positivo	Dinheiro: tamanho 8, Faixa: 0 a $500.000,00
Limit	Limite de Conta	Conjunto que representam o limite da conta em valor monetário	Dinheiro: tamanho 8, Faixa: 0 a $500.000,00
Status	Status atual Conta	Conjunto de três caracteres que representa o status da conta	Caractere: tamanho 3
Type	Tipo de Conta	Conjunto de um dígito do número que representa o tipo de conta (concessionário ou revendedor direto)	Número: tamanho 1, Faixa: 1 a 2

Figura 6: Domínio de Relação de Contas

A estrutura da relação, juntamente com a especificação dos domínios, é chamada intenção de uma relação. **É fixa**, a menos que o significado da relação mude, como a adição de novas colunas. **A extensão da relação** são as informações de uma relação que mudam ao longo do tempo, tais como as informações sobre as linhas da relação.

O grau de uma relação é o número de colunas que contém. A relação ACCOUNT tem seis atributos. Isso significa que cada linha da entidade tem seis tuplas, ou seja, seis colunas. A relação com apenas um atributo é conhecida como uma relação unária. A relação com dois atributos é conhecida como uma relação binária. A relação contendo três atributos é conhecida como uma relação ternária. Mais de três atributos se chama n-ária relação. O grau de relação faz parte da intenção da relação.

A cardinalidade de uma relação é o número de tuplas que esta contém. Muda a medida que linhas são adicionadas ou retiradas da tabela. Trata-se de uma propriedade da extensão da relação e é determinada a partir de um exemplo em particular a qualquer momento.

Há terminologias alternativas. A Figura 7 mostra isso. A segunda alternativa é a mais utilizada nos aspectos físicos do sistema de gerenciamento de banco de dados relacional.

Termos	Alternativa 1	Alternativa 2
Relação	Tabela	Entidade
Tupla	Linha	Registro
Atributo	Coluna	Campo

Figura 7: Terminologia Alternativa

3.1. Relações Matemáticas

Para compreender o verdadeiro significado da expressão "relação", uma revisão de alguns conceitos matemáticos é necessária. Suponha que tenhamos dois conjuntos, D1 e D2, onde

$D_1 = \{2, 4\} \in D_2 = \{1,3,5\}$

O produto cartesiano desses dois conjuntos, escrito como D1 x D2, é o conjunto de todos os pares ordenados tais que o primeiro elemento é um membro da D1 e o segundo elemento é um

membro da D2. É representada como se segue:

D_1 X D_2 = {(2,1), (2,3), (2,5), (4,1), (4,3), (4,5)}

Qualquer subconjunto do produto cartesiano é uma relação. Por exemplo, podemos produzir o seguinte:

R = {(2,3), (4,1)}

Pode-se especificar os pares ordenados que farão parte da relação, dando uma condição para a sua seleção. Exemplo, a relação R deve conter um par ordenado quando o segundo elemento é igual a três (3). Isto pode ser especificado como segue:

R = {(x,y) | x ∈ D1, y ∈ D2, e y = 3}

Pode-se facilmente ampliar a noção de uma relação a três conjuntos. Considerando D1, D2 e D3 como três conjuntos. O produto cartesiano é D1 X D2 X D3 quando se trata de um conjunto de todos os triplos ordenados tal que o primeiro elemento é a partir de D1, o segundo elemento é de D2 e do terceiro elemento é de D3. Qualquer subconjunto do produto cartesiano destes é uma relação. Por exemplo, suponha que temos:

D_1 = {1, 3}
D_2 = {2, 4}
D_3 = {5, 6}
D_1 X D_2 X D_3 = {(1,2,5), (3,2,5), (1,4,5), (3,4,5),
(1,2,6), (3,2,6), (1,4,6), (3,4,6)}

Em geral, seja D_1, D_2, ..., D_n n um conjunto de domínios. Seu produto cartesiano é definido como:

D_1 X D_2 X ... X D_n = {(d_1, d_2,...,d_n) | $d_1 \in D_1$, $d_2 \in D_2$, ... , $d_n \in D_n$}

normalmente é escrito como:

$$\mathop{\boldsymbol{X}}_{i=1}^{n} D_i$$

Qualquer conjunto de n-tuplas deste produto cartesiano é uma relação entre n conjuntos. Na definição dessas relações, deve-se especificar os conjuntos, ou domínios, a partir do qual se escolhe um valor.

3.2. Relações de Bancos de Dados

Aplicando estes conceitos para o banco de dados, um esquema de relação é um nome de relação seguido por uma série de colunas e os pares do nome do domínio. Seja A_1, A_2, ... , A_n colunas nos domínios D_1, D_2, ... , D_n. Portanto, o conjunto {A_1:D_1, A_2:D_2, ..., A_n:D_n} é um esquema de relação.

Uma relação R definida por um esquema de relação S é um conjunto de mapeamentos dos nomes das colunas correspondentes aos seus domínios. Assim, a relação R é um conjunto de n-tuplas.

(A_1:D_1, A_2:D_2, ... , A_n:D_n) tal que $d_1 \in D_1$, $d_2 \in D_2$, ... , $d_n \in D_n$

Cada elemento do n-tuplas consiste de uma coluna e um valor para esse atributo. Com relação a ACCOUNT, como exemplo, o esquema é o seguinte:

(**CellPhoneNo**:09192345678, **PIN**:1234, **Balance**:$500,00, **Limit**:$2.500,00, **Status**:ACT, **Type**:2)

No modelo relacional, as relações têm propriedades que deve existir. Ou seja, são os seguintes:

1. A relação tem um nome que é único na relação de todos os outros nomes dentro do banco de dados
2. Cada campo de uma relação contém exatamente um valor atômico (único)

3. Cada coluna tem um nome diferente
4. O valor de uma coluna são todos de um mesmo domínio
5. A ordem das colunas não tem qualquer significado
6. Cada linha é única, ou seja, não existem registros duplicados
7. A ordem da fila não tem qualquer significado, teoricamente. Na prática, a ordem das linhas tem efeito sobre a eficiência de acesso aos registros.

Quando bem construídas as relações contêm mínima redundância e permitem inserções, alterações e exclusões de linhas sem quaisquer erros ou inconsistências. Três tipos de anomalias devem ser evitadas quando se constrói relações de um banco de dados. São elas:

1. Anomalias de Inserção
2. Anomalias de Exclusão
3. Anomalias de Modificação

Para ilustrar anomalias, considere a seguinte figura em relação ao que não foi bem construído:

ACCOUNT1

CELLPHONENO	Type	CARDNO	LOADDATE	RECIPIENT
09205467234	2	2346253782	DEC-24-2006	9223456173
09205467234	2	8736237634	DEC-24-2006	9178746345
09165647342	1	9347536455	DEC-25-2006	9165234234
09174561234	1	2345234123	DEC-25-2006	9226352345
09174561234	1	6354239564	DEC-27-2006	9185343838

Figura 8: Tabela de Anomalias

1. **Anomalia de Inserção**. Considere a adição de um novo número de telefone celular. Nesta entidade, a chave primária é CELLPHONENO e CARDNO. O registro não pode ser adicionado porque CARDNO porque não foi informado ainda. Chaves primárias não podem ser nulas ou inexistentes.
2. **Anomalia de Exclusão**. Considere a possibilidade de excluir o registro com o valor de CELLPHONENO 09165647342. Uma vez removida, o CARDNO 9347536455 será perdido.
3. **Anomalia de Modificação**. Considere a modificação do tipo de conta com a mudança em CELLPHONENO de 09205467234 para 1. Neste exemplo, dois registros precisam ser atualizados.

3.3. Chaves Relacionais

Para identificar uma linha dentro de um relação, primeiro devemos identificar valores sem atributos iguais dentro desta relação. Uma **superchave** é uma coluna ou um conjunto de colunas que identifica exclusivamente uma linha dentro de uma relação.

Uma chave candidata é uma **superchave** tal que nenhum subconjunto é uma **superchave** dentro da relação. Uma chave candidata K, para uma relação R, possui duas propriedades:

1. **Única ou Distinta**. Em cada linha de R, os valores de K identificam-na exclusivamente
2. **Irredutível**. Nenhum subconjunto de K tem uma propriedade de singularidade

Uma chave composta é uma chave candidata que consiste em duas ou mais colunas.

Uma **chave primária** é uma chave candidata que é escolhida para identificar exclusivamente uma linha dentro de uma relação. Considerando que uma relação não possui nenhuma fila duplicada, sempre é possível identificar cada fila que usa a chave primária exclusivamente. Isto significa que uma relação sempre possui uma chave primária. No pior caso seria uma relação que contém todas as colunas como chave primária; porém, normalmente em algum subconjunto

menor de colunas é suficiente para distinguir as filas. As chaves candidatas que não são selecionadas como chaves primárias são conhecidas como chaves substitutas. Considere a relação de PHONECARDTRANSACTION. A chave primária também é uma chave composta que consiste nas colunas CELLPHONENO e CARDNO.

PHONECARDTRANSACTION

CELLPHONENO	CARDNO	LOADDATE	RECIPIENT
09205467234	2346253782	DEC-24-2006	9223456173
09205467234	8736237634	DEC-24-2006	9178746345

Figura 9: Relação PHONECARDTRANSACTION

Uma **chave estrangeira** é uma coluna ou um conjunto de colunas dentro de uma relação que emparelha a chave candidata de alguns (possivelmente no mesma) relação. Quando uma coluna aparecer em mais de uma relação, normalmente isto representa uma relação entre as filas das duas relações.

Por exemplo, considere as relações ACCOUNT e PHONECARDTRANSACTION. A coluna de CELLPHONENO aparece em ambas relações. Isto indica um relacionamento que uma conta pode ter transações de cartão telefônico.

4. Conceito de Normalização

Para construir relações bem-estruturadas, o conceito de normalização é usado. Normalização é o processo de converter estruturas complexas de dados em estruturas estáveis e simples. É realizado em fases chamadas de **formas normais**. Está baseado na análise das dependências funcionais.

Uma **forma normal** é um estado de uma relação que pode ser determinado aplicando-se regras simples relativas às dependências daquela relação.

Dependências funcionais são relações particulares entre dois atributos. Para qualquer relação **R**, o atributo **B** é funcionalmente dependente do atributo **A**, se, para todo exemplo válido de **A**, aquele valor de **A** determina exclusivamente o valor de **B**. Pode ser representado por:

```
A → B
```

Onde **A** é um determinante.

O determinante **A** é o atributo achado ao lado esquerdo da seta em uma dependência funcional. Pode ser usado como a chave primária da relação. Como um exemplo, considere a relação:

```
EMPLOYEE_COURSE (EMPLOYEE_ID, COURSE, DATE_COMPLETED)
```

Neste exemplo, EMPLOYEE_ID e COURSE formam a chave composta que determina DATE_COMPLETED funcionalmente. A dependência funcional é escrita a seguir:

```
EMPLOYEE_ID, COURSE → DATE_COMPLETED
```

DATE_COMPLETED de um COURSE é determinada pelo EMPLOYEE_ID que o efetuou.

Dependência funcional possui as seguintes regras:

1. **Regra Reflexiva**. Um atributo é funcionalmente dependente de si mesmo. É representado como:

   ```
   X → X
   ```

2. **Regra de Aumento**. Se X → Y, então X,Z → Y. Considere a dependência funcional:

   ```
   STUDENT NUMBER → STUDENT NAME
   ```

   Pela regra de aumento, um estudante pode ter a seguinte dependência funcional:

   ```
   STUDENT NUMBER, COURSE → STUDENT NAME
   ```

3. **Regra de União**. Se X → Y e X → Z, então X → Y,Z. Considere as duas dependências funcionais:

   ```
   STUDENT NUMBER → STUDENT NAME
   STUDENT NUMBER → ADDRESS
   ```

   Pode-se combinar em uma única dependência funcional, como a seguir:

   ```
   STUDENT NUMBER → STUDENT NAME, ADDRESS
   ```

4. **Regra de Decomposição**. Se X → Y e Z é um subconjunto de Y, então X → Z

5. **Regra de Transitividade**. Se X → Y e Y → Z, então X → Z. Considere as duas dependências funcionais:

   ```
   STUDENT NUMBER → MAJOR
   MAJOR → ADVISOR
   ```

   Da regra de transitividade, podemos ter a seguinte dependência funcional:

```
STUDENT NUMBER → ADVISOR
```

6. **Regra de Pseudotransitividade**. Se X → Y e Y, Z → W, então X, Z → W. Considere as duas dependências funcionais seguintes:

```
STUDENT NUMBER → MAJOR
MAJOR, CLASS → ADVISOR
```

Da regra de pseudotransitividade, pode-se ter a seguinte dependência funcional:

```
STUDENT NUMBER, CLASS->ADVISOR
```

4.1. Passos em Normalização

Normalização pode ser entendida e executada em estágios. Cada estágio corresponde a uma forma normal que é um estado de uma relação que resulta da aplicação de simples regras em relação a dependências funcionais. Como representação de nossa estrutura de dados, usaremos o diagrama de *Warnier-Orr* para mostrar as dependências funcionais das colunas. O diagrama de Warnier-Orr usa linhas para separar o que está sendo descrito e os itens que o compõem. A Figura 10 mostra os diagramas e o significado dos diagramas.

Diagrama	***Significado***
D ⟨ A, B, C	D consiste de A, B, e C
E ⟨ F (or), G (or), H	Seleciona somente um dos itens Que pode ser F, G, ou H
E ⟨ F (and/or), G (and/or), H	Qualquer um dos itens Que pode ser qualquer combinação.
J (1:3) ⟨ K, L, M, O (opcional)	Iteração ou Repetição J consiste de 1 a 3 iterações de K, L ,M ou O que é opcional

Figura 10: Diagrama Warnier-Orr

Considere o seguinte visão da Figura 11 que possui seu correspondente com o diagrama *Warnier-Orr* na Figura 12. Normalização será usada para definir um conjunto bem definido de relações.

CONTRACT SCREEN

Contract No.: 49836 Customer Name: Tipp, Q.
Customer No.: TIP32 Address: 214th Avenue Cubao Quezon City
Contract Date: 16 Feb 1993
Sales Office: CUB001

Appliance No.	Model No.	Manufacturer	Date	Services
140446	CTV27	Hitachi	26 Aug 1992	2
310543	VHS03	JVC Phil	03 Feb 1993	0
140221	CTV35	Sony	01 Mar 1993	1

Figura 11: Lista de Contrato por Cliente

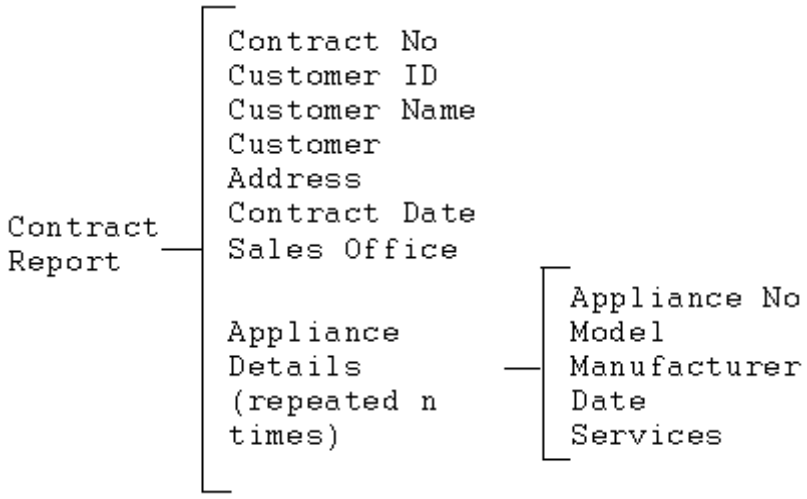


Figura 12: Diagrama Warnier-Orr da Lista de Contrato por Cliente

4.1.1. Primeira Forma Normal (FNF)

Uma relação está na primeira forma normal (FNF) se não contém grupos repetidos. O que segue é o procedimento:

```
** Un-normalized Form to First Normal Form
** Objective: Remove Repeating Groups
From non-repeated data items
 Identify a Primary Key
 IF none is found
  Create a Primary Key
 ENDIF
 IF there is/are repeating group (s)
  DO WHILE there are repeating groups
   From the outer-most repeating groups,
    Identify a Group Key
    IF none is found
     Create a Group Key
    ENDIF
    ** Break un-normalized form into two relations
    Relation 1 = Primary Key + non-repeating data items
    Relation 2 = Primary Key + Group Key + group data items
    IF Relation 2 still has repeating groups
     Primary Key = Old Primary Key + Group Key
    ENDIF
  ENDDO
 ENDIF
 Rename Resultant Relation
 Draw First Normal Form Data Model
** Relations are now in the First Normal Form
```

Diretivas para escolher uma chave:

- Deve identificar unicamente componentes, itens ou entidades representadas em dados não normalizados
- Não deve estar repetida dentro de dados não normalizados
- A chave deve ser única para cada ocorrência de um grupo de dados não normalizados
- Se existe uma escolha a ser feita entre itens de chave, escolha baseado em:
  - um campo, ao invés de vários
  - numérico, ao invés de alfanumérico
  - comprimento fixo, ao invés de comprimento variável

- formato fixo, ao invés de formato variável
- itens curtos, ao invés de itens longos

O campo APPLIANCE DETAIL é considerado um grupo repetido. Foi removido para formar outra relação com sua chave primária própria. A Figura 13 mostra as relações resultantes.

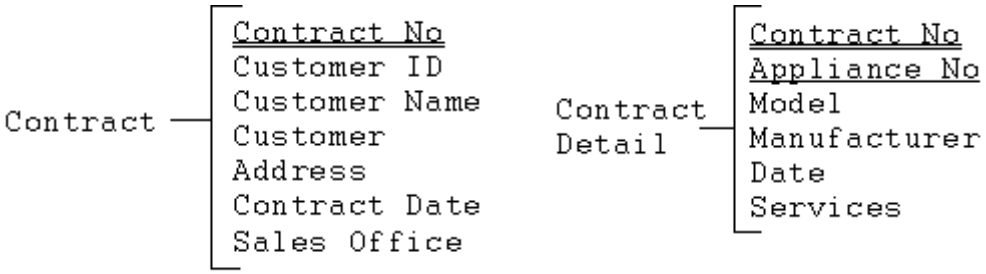


Figura 13: Primeira Forma Normal (FNF) – Lista de Contratos por Cliente

4.1.2. Segunda Forma Normal (SNF)

Uma relação está na Segunda Forma Normal se está em FNF e cada atributo não-chave é completamente funcionalmente dependente da chave primária. Um **atributo não-chave** é um atributo que não é a chave primária ou é parte de uma chave composta.

As relações identificadas não podem ser usadas como base para o projeto de dados, pois possuem propriedades indesejáveis que levam a atualizações anômalas. Para mudar de FNF para SNF, remova as dependências de partes da chave. Envolve examinar as relações que têm uma chave composta ou concatenada e para cada campo faça as seguintes perguntas: Pode o campo ser unicamente identificado por uma parte da chave, ou é necessária a sua totalidade (chave composta) ?

```
** First Normal Form to Second Normal Form Procedure
** Objective: Remove attributes not dependent on the Relation's Composite
**            keys.
From the First Normal Form Relations
 Identify Relations with Composite Keys
 FOR EACH Relation with Composite Keys
  Identify Composite Key
  Mark with a left-side double bracket
  Identify Part Keys
  Mark with a right-side single bracket
  FOR EACH Non-key data item
   IF data item is dependent on the Composite Key
    Draw a left-side line to Composite Key
   ELSE
    Draw a right-side line to Part Key
   ENDIF
  ENDFOR
  ** Break First Normal Form Relation into two Relations
  Relation 1 = Composite Key + Left-side data items
  Relation 2 = Part Key + Right-side data items
 ENDFOR
 Rename Resultant Relations
 Draw Second Normal Form Data Model
** Relations are now in the Second Normal Form
```

A relação CONTRACT tem somente uma chave, conseqüentemente, já está na segunda forma normal. Entretanto, CONTRACT DETAIL possui um chave composta, de modo que ela deve estar sujeita ao procedimento SNF.

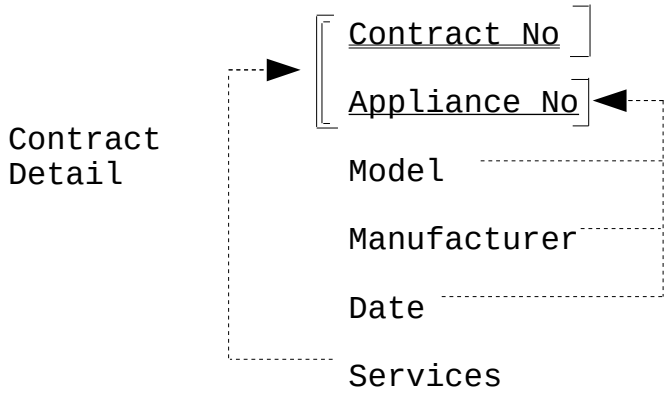


Figura 14: Dependências Funcionais de Contract Details

Somente SERVICES é completamente funcionalmente dependente da chave composta. O resto (MODEL, MANUFACTURER e DATE) é funcionalmente dependente de APPLIANCE NO. As relações resultantes podem ser vistas na Figura 15.

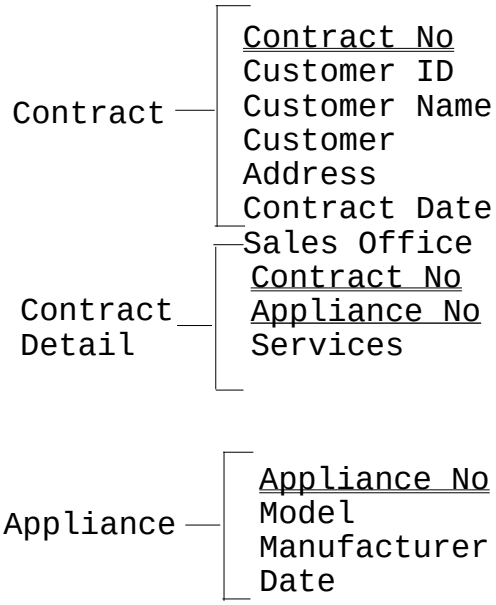


Figura 15: Segunda Forma Normal da Lista de Contratos por Cliente

4.1.3. Terceira Forma Normal (TNF)

Uma relação está na terceira forma normal se é SNF e não existem dependências transitivas. Uma ***dependência transitiva*** em uma relação é uma dependência funcional entre dois (ou mais) atributos não-chave.

Algumas anomalias podem ainda existir nos dados. O terceiro passo de normalização lida com a identificação de chaves estrangeiras e atributos, removendo-os de modo que não haverá mais dependências entre os itens de dados ou chaves.

```
** Second Normal Form to Third Normal Form Procedure
** Objective: Remove Foreign Keys and their attributes
FOR EACH Second Normal Form Relation
 IF there is a possible Foreign Key
  Identify Foreign Key
  IF Foreign Key can be replaced with a Code
   Add Code to data items
   Replace Foreign Key with Code
  ENDIF
```

```
    FOR EACH Non-Key data item
     IF data item is dependent on the Primary/Composite Key
      Draw a left-side line to Primary/Composite Key
     ELSE
      Draw a right-side line to Foreign Key
     ENDIF
    ENDFOR
    ** Break Second Normal Form into two relations
    Relation 1 = Primary/Composite Key + Foreign Key + left-side data items
    Relation 2 = Foreign Key + right-side data items
   ENDIF
   Rename resultant relation
  ENDFOR
  Draw Third Normal Form Data Model
  ** Relations are now in the Third Normal Form
```

Olhando as relações SNF, as possíveis chaves estrangeiras são CONTRACT e APPLIANCE. Para CONTRACT, CUSTOMER ID é considerada uma chave estrangeira. Para APPLIANCE, MODEL é considerada uma chave estrangeira. A Figura 16 mostra a dependência transitiva das relações.

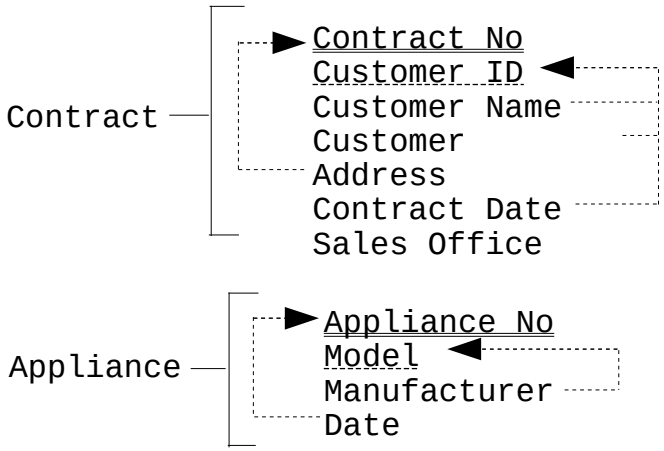


Figura 16: Dependência Transitiva da Lista de Contratos do Cliente

As relações resultantes são mostradas na Figura 17.

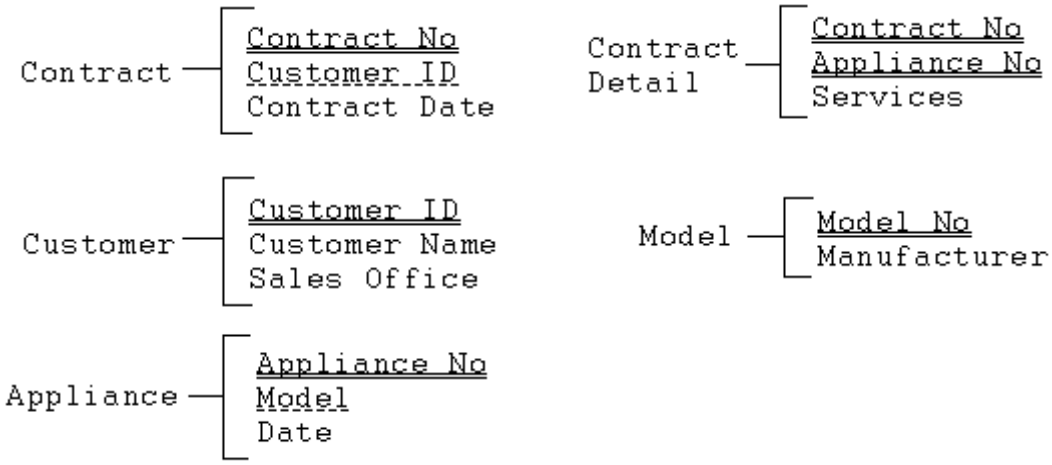


Figura 17: Terceira Forma Normal da Lista de Contratos por Clientes

Relações em TNF são suficientes na maioria das aplicações práticas de bancos de dados. Entretanto, TNF não garante que todas as anomalias foram removidas. Existem outras formas normais que permitem a remoção de certas anomalias.

4.1.4. Forma Normal de Boyce-Codd (BCNF)

Uma relação está em BCNF se, e somente se, cada determinante é um candidato a chave. Como revisão, um determinante é um atributo ou grupo de atributos no qual algum outro atributo é completamente e funcionalmente dependente. Certas pistas nos permitem determinar se uma relação viola a BCBF. São elas:

- A relação contém dois (ou mais) candidatos a chaves compostas
- O candidato a chave composta sobrepõe ou compartilha no mínimo um atributo em comum

Como exemplo, considere a relação definida na Figura 18. PATIENT_APPOINTMENT.

PATIENT_APPOINTMENT

Patient_No	Interview_Date	Interview_Time	Staff_No	Room_No
1077	May 13, 2004	9:30	105	101
1056	May 13, 2004	10:30	105	101
1065	May 13, 2004	1:00	137	102
1056	July 1, 2004	9:30	105	102

Figura 18: Relação Patient Appointment

As dependências funcionais da relação são:

Patient_No, Interview_Date → Interview_Time, Staff_No, Room_No
Staff_No, Interview_Date → Patient_No
Staff_No, Interview_Date → Room_No

Os determinantes de duas dependências funcionais são candidatos a chave composta da relação. Entretanto, o último não é porque ROOM_NO é mais funcionalmente dependente em STAFF_NO, INTERVIEW_DATE do que PATIENT_NO, INTERVIEW_DATE. Para transformar a relação para BCNF, crie duas novas relações. A Figura 19 mostra as relações resultantes.

PATIENT_APPOINTMENT

Patient_No	Interview_Date	Interview_Time	Staff_No
1077	May 13, 2004	9:30	105
1056	May 13, 2004	10:30	105
1065	May 13, 2004	1:00	137
1056	July 1, 2004	9:30	105

ROOM_ASSIGNMENT

Staff_No	Interview_Date	Room_No
105	May 13, 2004	101
137	May 13, 2004	102
105	July 1, 2004	102

Figura 19: Relação BCNF Patient Appointment

Pode não ser sempre desejável transformar uma relação em BCNF. Se existe uma dependência funcional que não está preservada quando a decomposição é executada, não a converta em BCNF.

4.1.5. Quarta Forma Normal

Esta é a forma de normalização a qual é uma relação BCNF e contém dependências de multivalores não triviais. **Dependências de Multivalores (*DMV)*** representa uma dependência entre atributos A, B, e C numa relação que para cada valor de A, existe um conjunto de valores de B de C. No entanto, os conjuntos de valores de B e C são independentes. A notação de dependência de multivalores é a seguinte:

```
A ->> B
A ->> C
```

Uma DMV trivial é definida por DMV A ->> B em relação a R se satisfaz a seguinte condição:

- B é um subconjunto de A ou
- $A \cup B = R$

Uma DMV não-trivial acontece quando as regras especificadas numa DMV trivial são violadas. Por exemplo: B não é um subconjunto A ou $A \cup B$ não resulta em R.

Como exemplo, considere a relação BRANCH_STAFF_CUSTOMER_ASSIGNMENT na Figura 20. Considere as seguintes premissas.

1. Um BRANCH possui vários STAFF.
2. Um BRANCH possui vários COSTUMERS.
3. Não há nenhuma relação exclusiva entre STAFF and CUSTOMER. Por exemplo, Elaine Stan pode se inscrever tanto para Ann Bautista como para David Fort.

BRANCH_STAFF_CUSTOMER_ASSIGNMENT

Branch_No	Staff_Name	Customer_Name
1003	Ann Bautista	Elaine Stan
1003	David Fort	Elaine Stan
1003	Ann Bautista	Mike Ross
1003	David Ford	Mike Ross

Figura 20: Relacionamento Branch-Staff-Customer

O DMV que existe nessa relação é:

```
Branch_No ->> Staff_Name
Branch_No ->> Customer_Name
```

Para normalizar, remova o DMV criando duas novas relações. Figura 21 Mostra o resultado.

BRANCH_STAFF

Branch_No	Staff_Name
1003	Ann Bautista
1003	David Fort

BRANCH_CLIENT

Branch_No	Customer_Name
1003	Elaine Stan
1003	Mike Ross

Figura 21: Quarta Forma de Normalização da relação Branch-Staff-Customer

4.1.6. Quinta Forma Normal

Uma relação que não há nenhuma dependência de união. ***Dependência Lossless-Join*** é uma propriedade de decomposição, a qual assegura que nenhuma linha falsa será gerada quando as relações forem reunidas numa junção natural. Considere a relação SUPPLIER_SHIPMENT na Figura 22. A relação descreve os items providos pelo fornecedor (supplier) para uma loja (store) particular. Não suporta nenhuma restrição que certos tipos de fornecedores (suppliers) deveriam prover certos items para uma loja (store) em particular. Tirando o fato que um determinado fornecedor pode prover todos items requeridos para uma determinada loja. Por exemplo, para a loja 104, apenas o fornecedor 10 pode prover *sofa beds* para aquela loja. Embora o fornecedor 20 também possa prover sofa bed.

SUPPLIER_SHIPMENT

Store_No	Item_Description	Supplier_No
104	Sofa Bed	10
104	Computer Chair	20
116	Sofa Bed	20
116	Table	10
136	Computer Chair	30

Figura 22: Relacionamento Supplier-Shipment

Esta relação pode ainda adicionar linhas (104, Sofa Bed, 20) e (104, Computer Chair, 30). Para transformar a relação em 5NF, decomponha a relação em três que suportem as seguintes restrições:

- Items providos por um fornecedor (supplier).
- Lojas (Stores) que o fornecedor por suprir items.
- Items achados nas lojas.

STORE_ITEM

Store_No	Item
104	Sofa Bed
104	Computer Chair
116	Sofa Bed
116	Table
136	Computer Chair

SUPPLIER_ITEM

Supplier_No	Item
10	Sofa Bed
10	Computer Chair
20	Sofa Bed
20	Table
30	Computer Chair

STORE_SUPPLIER

Store_No	Supplier_No
104	10
104	20
116	10
116	20
136	30

Figura 23: Quinta Forma de Normalização- Store-Item-Supplier Shipment

A Figura 23 mostra o resultado dos relacionamentos.

5. Transformando E-R Diagramas em Relacionamentos

Como exemplo, iremos transformar o diagrama de Entidade-Relacionamento do capítulo anterior e será mostrado a seguir:

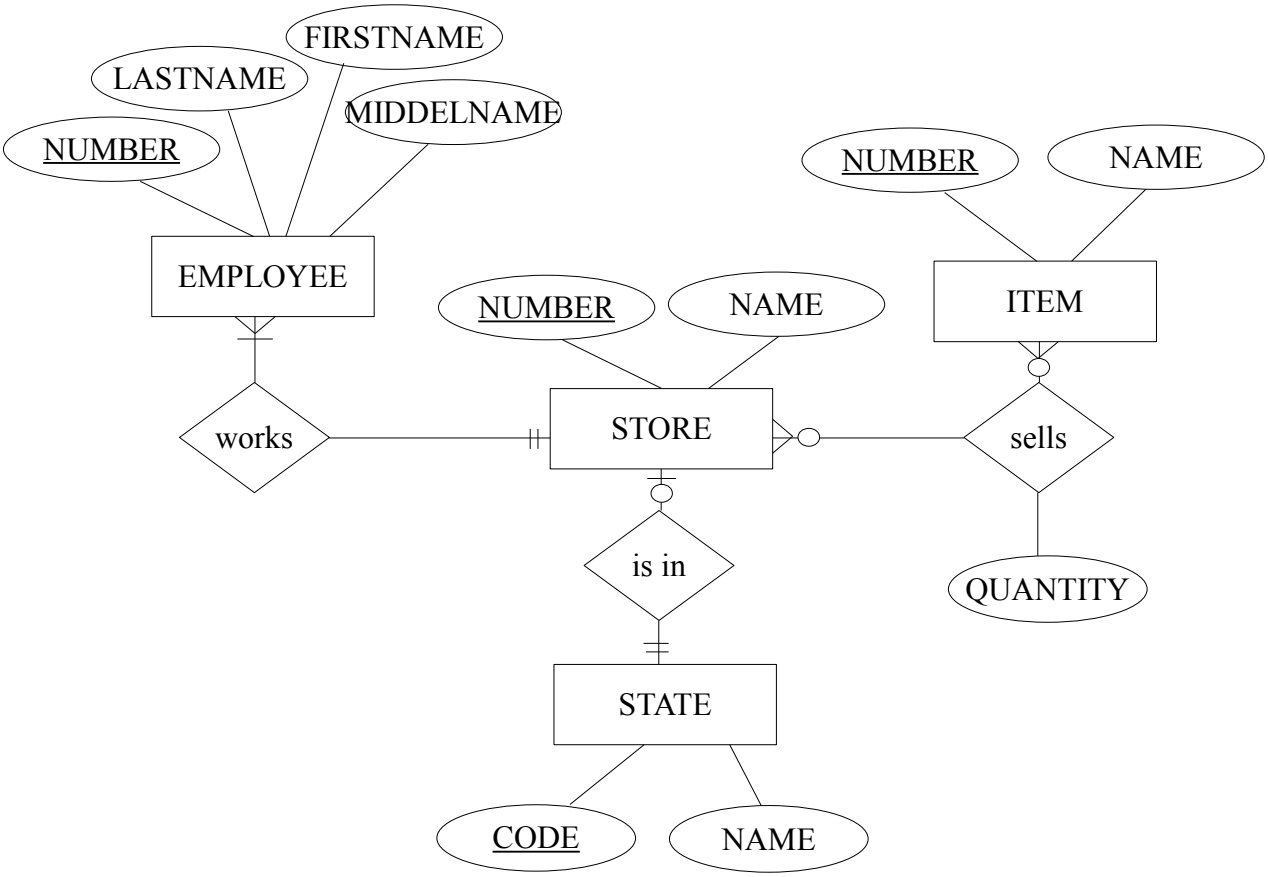


Figura 24: Diagrama Entidade - Relacionamento Organic Shop

5.1. PASSO 1: Representar Entidades

Cada tipo de entidade se torna uma relação. A chave primária da entidade se torna a chave primária da relação. Como revisão, chave primária deve ter as seguintes propriedades:

- O valor da chave deve ser um identificador único para cada linha da relação.
- A chave não pode ser redundante; ou seja, nenhum atributo da chave pode ser removido sem destruir identificação única.

Cada atributo não-chave da entidade se torna atributo não-chave da relação.

Como exemplo, considere a entidade modelada na Figura 25.

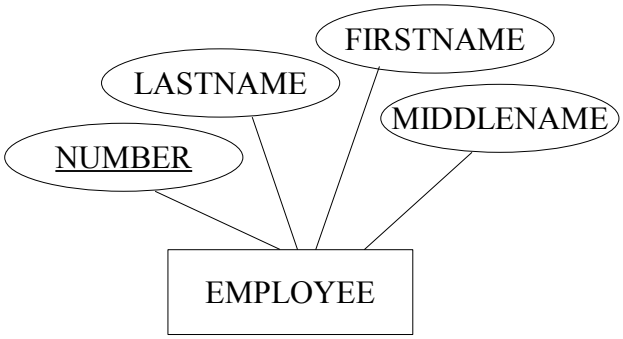


Figura 25: Employee DER

Podemos representá-la usando a tabala, shorthand or Warnier-Orr. No exemplo, a chave primária EMPLOYEE é Number enquanto os outros atributos (LastName, FirstName, MiddleName) são atributos não-chave da relação. O resultado da relação é mostrado na Figura 26.

EMPLOYEE

Number	LastName	FirstName	MiddleName
5000	Stone	Benjamin	J.
5001	Cruz	Juan	M.
5002	Enriquez	Sheila	S.P.
5003	Choo	James	Y
5004	Mendes	Lani	M,

EMPLOYEE(Number, LastName, FirstName, MiddleName)

Figura 26: Relação Employee

5.2. PASSO 2: Representando Relacionamentos

A representação dos relacionamentos depende do nível dos *relacionamentos* (unário, binário ou ternário) e as *cardinalidades dos relacionamentos* (mandatório um, optionalmente zero etc.).

5.2.1. Representando um relacionamento Binário 1:N

No lado **N** da entidade, coloque a chave primária no lado **1** da entidade, tornando-a chave estrangeira. No exemplo da Figura 27, a chave primária de STORE (Number) se torna a chave estrangeira de EMPLOYEE (Store). Os valores dos atributos de Store contém valores achados no domínio de Number na relação de STORE.

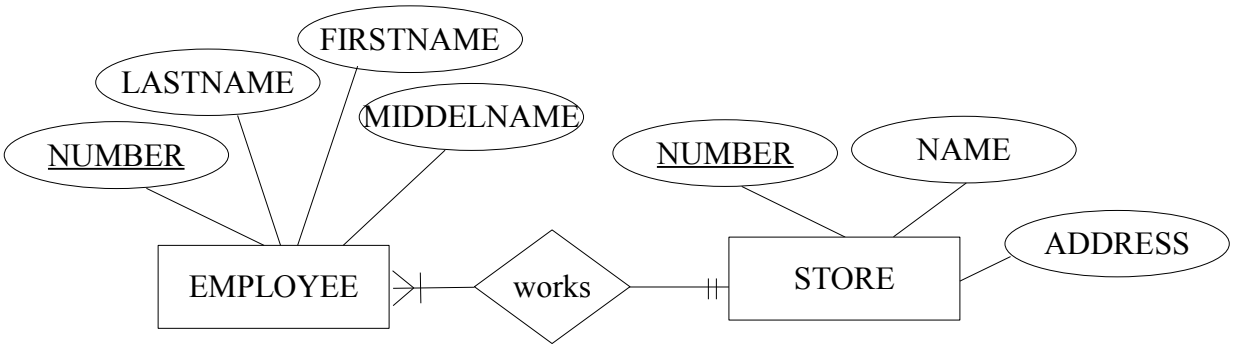


Figura 27: DER de Employee works in a store

O resultado da relação são mostrados na Figura 28.

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store
5000	Stone	Benjamin	J.	10
5001	Cruz	Juan	M.	10
5002	Enriquez	Sheila	S.P.	10
5003	Choo	James	Y.	10
5004	Mendes	Lani	M.	10

EMPLOYEE(Number, LastName, FirstName, MiddleName, Store)

STORE

Number	Name	Address
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	North Dakota

STORE (Number, Name, Address)

Figura 28: Relações achadas no Employee works in a store.

5.2.2. Representando uma relação Binária M:N

O relacionamento de duas entidades se torna uma relação. A chave primária de uma relação é a combinação das chaves primárias das duas entidades. Qualquer atributo associado com o relacionamento se torna um atributo da relação. No exemplo da Figura 29, o relacionamento sells (Vende) torna-se uma relação chamada INVENTORY, e sua chave primária é a chave composta a qual é uma combinação com as chaves primárias de STORE e ITEM. Count e Operational Level são atributos de INVENTORY. O resultado das relações são mostrados na Figura 30.

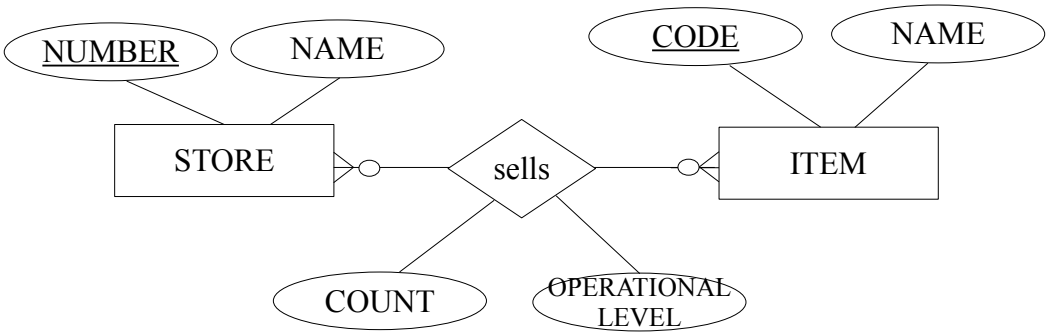


Figura 29: DER Store's inventory

STORE

Number	Name	Address
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	North Dakota

STORE (Number, Name, Address)

ITEM

Code	Description
1001	Wheat Germs
1002	Tarragon
1003	Yacon
1004	Thyme
1005	Bay Leaves

ITEM(Code, Description)

INVENTORY

Store No	Item Code	Count	Operational Level
10	1001	2345	500
20	1001	4085	1000
10	1006	2115	300
20	1006	664	300

INVENTORY(Store No, Code, Count, Operational Level)

Figura 30: Relacionamento Store's Inventory

5.2.3. Representando Relacionamento Unário

Um tipo de entidade é modelada como um relacionamento. A chave primária da relação é a chave primária da entidade. Uma chave estrangeira é adicionada à relação que referencia os valores da chave primária da mesma relação(chave estrangeira recursiva). No exemplo mostrado na Figura 31, um EMPLOYEE está sendo gerenciado por um MANAGER que é também um EMPLOYEE. O atributo Manager da relação é, na verdade, o employee Number do manager. O resultado das relações são mostrados na Figura 32.

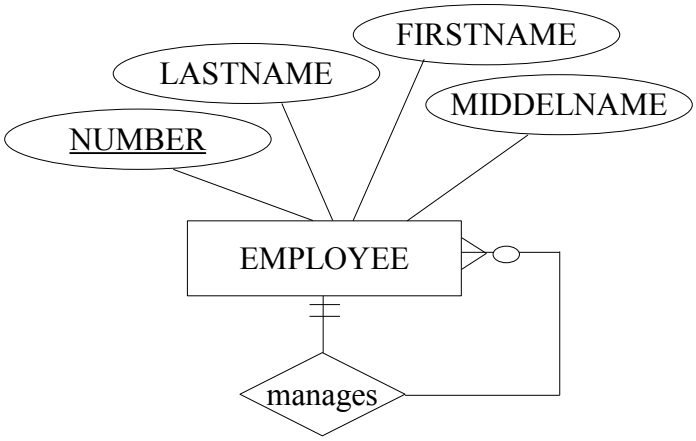


Figura 31: DER Employee is managed by a manager

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager
5000	Stone	Benjamin	J.	10	NULL
5001	Cruz	Juan	M.	10	5001
5002	Enriquez	Sheila	S.P.	10	5001
5003	Choo	James	Y.	10	5001
5004	Mendes	Lani	M	10	5001

EMPLOYEE(Number, LastName, FirstName, MiddleName, Store, Manager)

Figura 32: Relações achadas em Employee is managed by manager.

Um outro exemplo é mostrado na Figura 33 que mostra um relacionamento unário de muitos-para-muitos. Como revisão, o relacionamento se torna a relação. O resultado do relacionamento se torna uma relação COMPOSTA. Neste exemplo, um train consiste em panels, wheels e nails. A chave primária da COMPOSIÇÃO consiste no item NUMBER (Item_No), e a parte NUMBER (Component_No) do item.

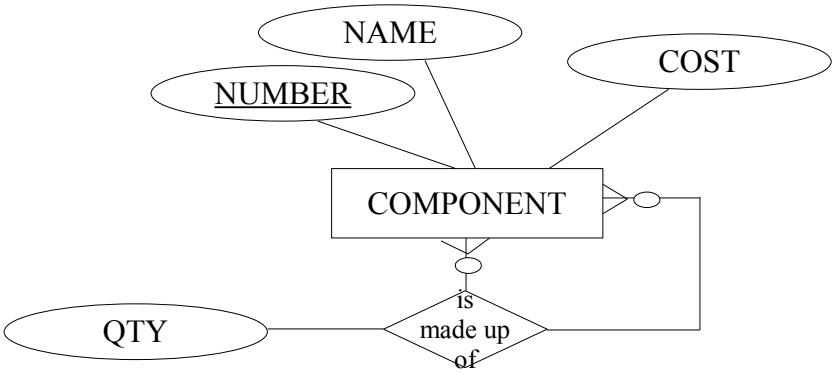


Figura 33: Componentes consistem de pequenos compoentes ERD

Component

Number	Name	Cost
1001	Toy Train	Php900,00
1002	Panel of Train	Php50,00
1003	Train Wheels	Php20,00
1004	Nails	Php1,00

COMPONENT (Number, Name Cost)

Composition

Item_Number	Component_No	Qty
1001	1002	7
1001	1003	4
1001	1004	20

COMPOSITION(Item_No, Component_No, Qty)

Figure 34: Relations found in Components consists of smaller components.

5.2.4. Representando a Relação Is-A (Classe/Subclasse)

Os dados do modelo relacional não suporta diretamente relacionamento classe / subclasse. No entanto, existem diversas estratégias que podem se utilizar nos projetos. As estratégias que podem ser empregadas são as seguintes:

1. Criar uma relação separada para a classe e para cada uma das subclasse
2. A entidade ou relação ser constituída apenas para a classe dos atributos que são comuns a todos da subclasse
3. A entidade para cada subclasse conter apenas a sua chave primária e as colunas únicas da subclasse
4. As chaves primárias da classe e de cada uma das subclasses serem do mesmo domínio

Como um exemplo, considere-se a relação Is-A de EMPLOYEE na Figura 38. Atributos comuns a todas as subclasse de trabalhadores estão localizados na superclasse EMPLOYEE. Atributos específicos para a subclasse são encontradas na relação da subclasse. As relações são mostradas na Figura 36.

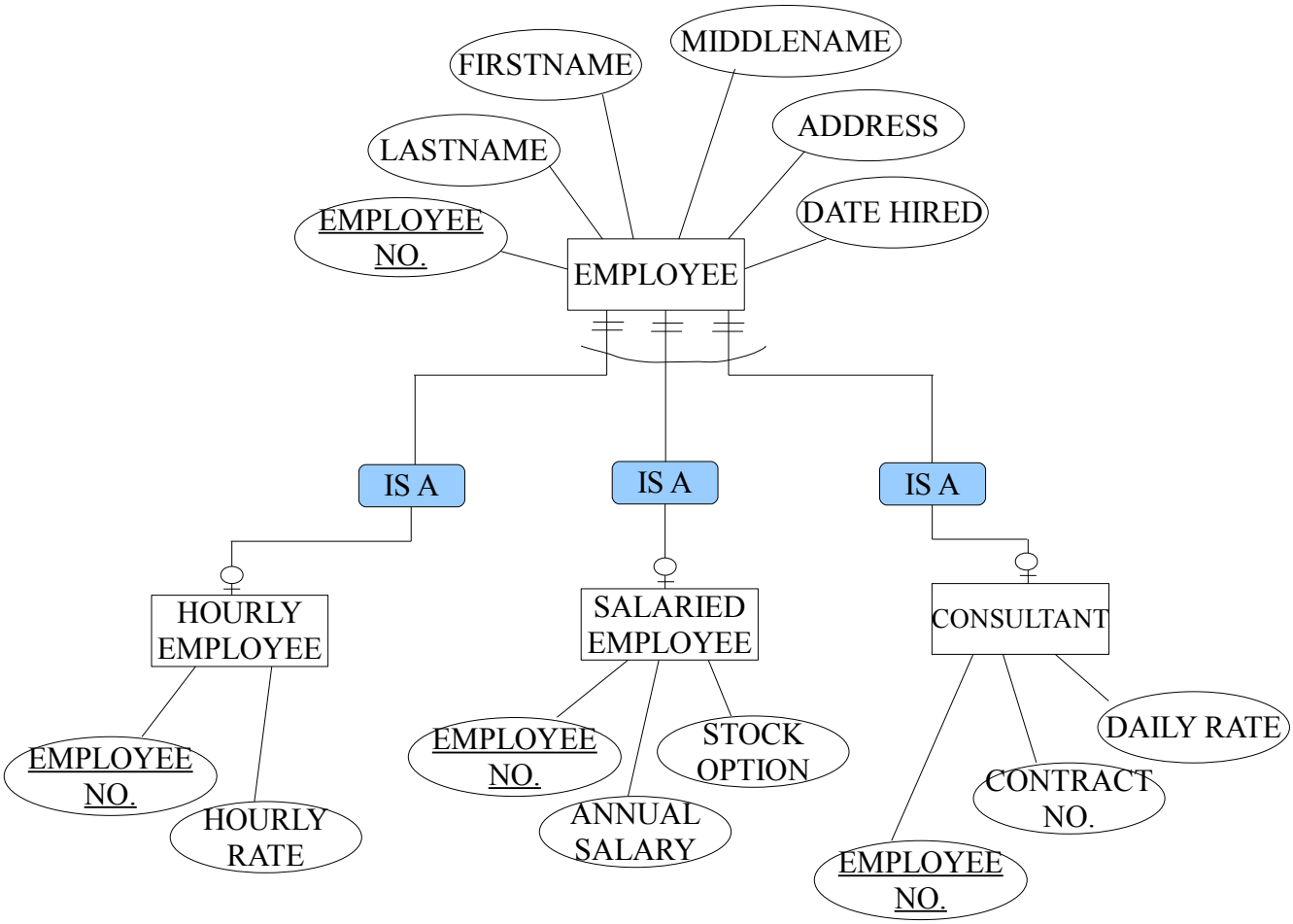


Figura 35: Categorização Employee ERD

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
5000	Stone	Benjamin	J.	10	NULL	12/25/1997
5001	Cruz	Juan	M.	10	5001	01/10/1998
5002	Enriquez	Sheila	S.P.	10	5001	01/10/1998
5003	Choo	James	Y.	10	5001	01/10/1998
5004	Mendes	Lani	M	10	5001	01/10/1998

EMPLOYEE(Number, LastName, FirstName, MiddleName, Store, Date_Hired)

HOURLY_EMPLOYEE

Number	Hourly_Rate
5002	Php100,00
5003	Php200,00

HOURLY_EMPLOYEE(Number, Hourly_Rate)

SALARIED EMPLOYEE

Number	Annual_Salary	Stock_Option
5001	Php345.000,00	N
5005	Php330.000,00	N

SALARIED_EMPLOYEE(Number, Annual_Salary, Stock_Option)

CONSULTANT

Number	Contract_No	Daily_Rate
5013	AH0001-10001	Php1.000,00

CONSULTANT(Number, Contract_No, Daily_Rate)

Figura 36: Categorização das Relações de Employee

5.2.5. PASSO 3: Normalizar as Relações

Após representando o ERD para as relações, o próximo passo é normalizar as relações. Normalizando-se a TNF forma usualmente suficiente. No entanto, se existe a necessidade de normalizar ainda mais, continuar até chegar à quinta forma normal. A lista completa de relações normalizadas (sem fusão) da Organic Shop é mostrado na Figura 37.

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store
5000	Stone	Benjamin	J.	10
5001	Cruz	Juan	M.	10
5002	Enriquez	Sheila	S.P.	10
5003	Choo	James	Y.	10
5004	Mendes	Lani	M.	10

STORE

Number	Name	Address
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	North Dakota

ITEM

Code	Description
1001	Wheat Germs
1002	White Peppers
1003	Iodized Salts
1004	Oregano

INVENTORY

Store_No	Item_Code	Quantity	Operational Level
10	1001	50	20
20	1001	2300	500
10	1004	4500	100
20	1004	90	100

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
5000	Stone	Benjamin	J.	10	NULL	12/25/1997
5001	Cruz	Juan	M.	10	5001	01/10/1998
5002	Enriquez	Sheila	S.P.	10	5001	01/10/1998
5003	Choo	James	Y.	10	5001	01/10/1998
5004	Mendes	Lani	M	10	5001	01/10/1998

HOURLY_EMPLOYEE

Number	Hourly_Rate
5002	Php100,00
5003	Php200,00

SALARIED EMPLOYEE

Number	Annual_Salary	Stock_Option
5001	Php345.000,00	N
5005	Php330.000,00	N

CONSULTANT

Number	Contract_No	Daily_Rate
5013	AH0001-10001	Php1.000,00

Figura 37: Normalizando as Relações de Organic Shop

5.2.6. PASSO 4: Juntando as Relações

Após normalizado, é necessário fundir as relações que se referem à mesma entidade. No entanto, deve-se verificar se existem problemas de integração, tais como as seguintes:

1. **Sinônimos**. Eles são dois atributos que têm nomes diferentes, mas o mesmo significado. Ao fundir as relações que contém sinônimos, você deverá obter acordo (se possível) de usuários em nomes únicos e padronizados, para o atributo afim de eliminar sinônimos. No exemplo, o EMPLOYEE utiliza relação Number e Employee_no como a chave primária. Ambos têm o mesmo significado. Neste caso, escolhe-se um nome para o atributo; iremos usar Number.
2. **Homônimos**. Um único atributo pode ter mais do que um significado. Para resolver o conflito, é necessário criar novos nomes de atributos.
3. **Dependência de Transitividade**. Assegurar que, quando as relações são fundidos a trasitividade das dependências não ocorram novamente.
4. **Classe/Subclasse (Is-A)**. Se uma relação não parece correta, ou seja, falta determinados atributos, verifica-se se é uma subclasse de outra classe.

No exemplo, precisamos de fundir a relação em EMPLOYEE. A seguir temos o último conjunto de relações.

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
5001	Cruz	Juan	Martinez	10	5000	Dec. 25 2005
5002	Enriquez	Sheila	San Pedro	10	5001	Jun. 04 1998
5003	Ferrer	Grace	Atienza	20	5000	Jan. 16 1997

STORE

Number	Name	Address
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	North Dakota

ITEM

Code	Description
1001	Wheat Germs
1002	White Peppers
1003	Iodized Salts
1004	Oregano

INVENTORY

Store_No	Item_Code	Count	Operational Level
10	1001	50	20
20	1001	2300	500
10	1004	4500	100
20	1004	90	100

HOURLY_EMPLOYEE

Number	Hourly_Rate
10002	Php250,00
10005	Php250,00

SALARIED EMPLOYEE

Number	Annual_Salary	Stock_Option
10001	Php345.000,00	Yes
10006	Php546.000,00	Yes

CONSULTANT

Number	Contract_No	Daily_Rate
10004	AH0001-10001	Php1.000,00

Figura 38: O Projeto Lógico do Banco de Dados de Organic Shop

6. Exercícios

6.1. Normalizar

Normalizar de forma que o usuário visualize o conjunto de relações na 3ª Forma Normal (TNF).

CUSTOMER SERVICE REPORT

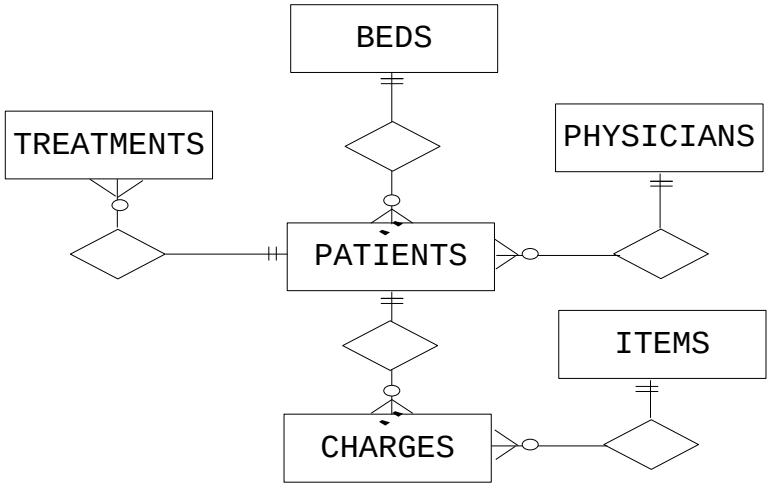
Customer ID: SMI47 Customer Name: Smith, J. Service Depot: N0V02

Address: 49 West Triangle Road Palmeras III Novaliches Quezon City — 349 Rizal Ave., Pepsi Bldg. Novaliches

Contract No.	Contract Date	Monthly Payment	Service Visits Date	Job Number
27015	27 Mar 1992	345.50	22 Aug 1992	34/3290
			18 Mar 1993	34/4321
24992	20 May 1992	1,345.00	13 Jul 1992	34/1766
			18 Dec 1992	34/2541
			06 Aug 1993	34/3115

6.2. ERD

Transformar o seguinte ERD para uma relação bem estruturada. Atributos não são incluídos no diagrama afim de manter clareza. No entanto, os atributos são listados após o diagrama. Assegurar que as relações serão normalizadas.



Os atributos da entidade são os seguintes tipos:

- Bed
  - Bed Number
  - Bed Type
- Patients
  - Patient Number
  - Patient Name (Last Name, First Name, Middle Name)
  - Address
  - Telephone or Mobile Number
- Treatments
  - Treatment Number
  - Diagnosis
  - Treatment Description
- Physician
  - Physician Number

    - Physician Name (Last Name, First Name, Middle Name)
    - Clinic Address
    - Telephone or Mobile Number
- Item
    - Item Number
    - Description
- Charges
    - Date charges are applied
    - Amount

6.3. Diagrama de Entidade e Relacionamento

- Transformar o diagrama entidade-relacionamento desenvolvido a partir do problema 1 no capítulo 2 (The Apartment Manager) em um conjunto de relações normalizadas.
- Transformar o diagrama entidade-relacionamento desenvolvido a partir do problema 2 no capítulo 2 (The Electronic Shop Manager) em um conjunto de

Parceiros que tornaram JEDI™ possível

Instituto CTS
Patrocinador do DFJUG.

Sun Microsystems
Fornecimento de servidor de dados para o armazenamento dos vídeo-aulas.

Java Research and Development Center da Universidade das Filipinas
Criador da Iniciativa JEDI™.

DFJUG
Detentor dos direitos do JEDI™ nos países de língua portuguesa.

Banco do Brasil
Disponibilização de seus *telecentros* para abrigar e difundir a Iniciativa JEDI™.

Politec
Suporte e apoio financeiro e logístico a todo o processo.

Borland
Apoio internacional para que possamos alcançar os outros países de língua portuguesa.

Instituto Gaudium/CNBB
Fornecimento da sua infra-estrutura de hardware de seus servidores para que os milhares de alunos possam acessar o material do curso simultaneamente.