

UFSC-CTC-INE-PPGCC

INE 410131 – Gerência de Dados para Big Data

Aula 6 – Bancos de Dados NewSQL

UFSC-CTC-INE-PPGCC

INE 410131 – Gerência de Dados para Big Data

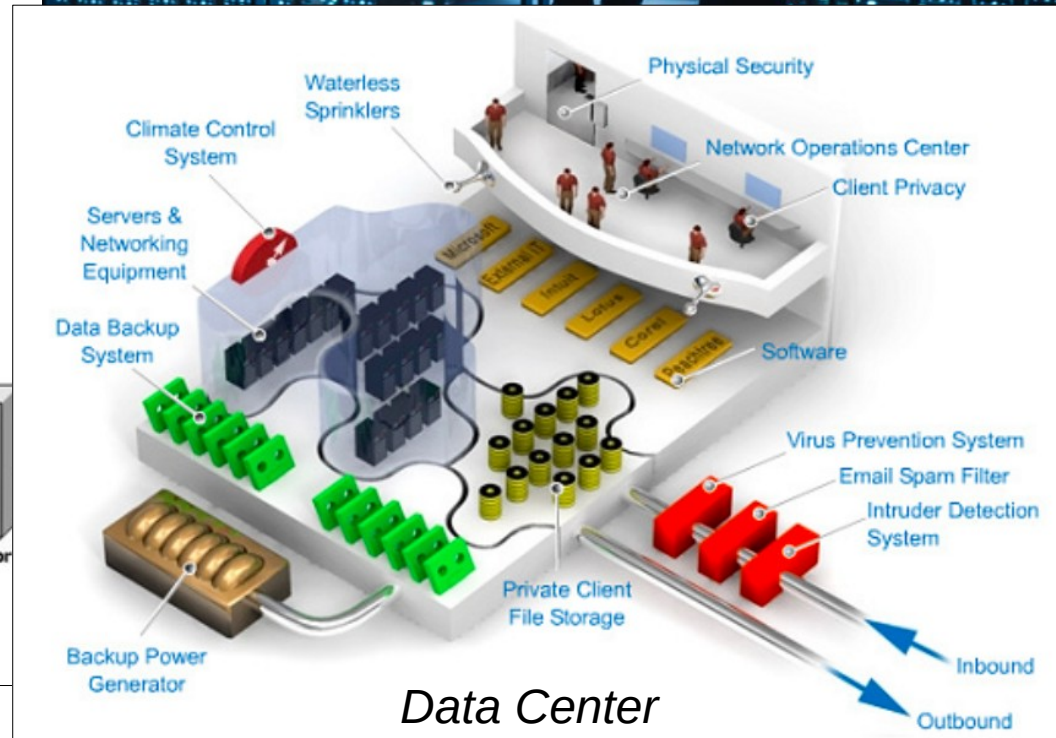
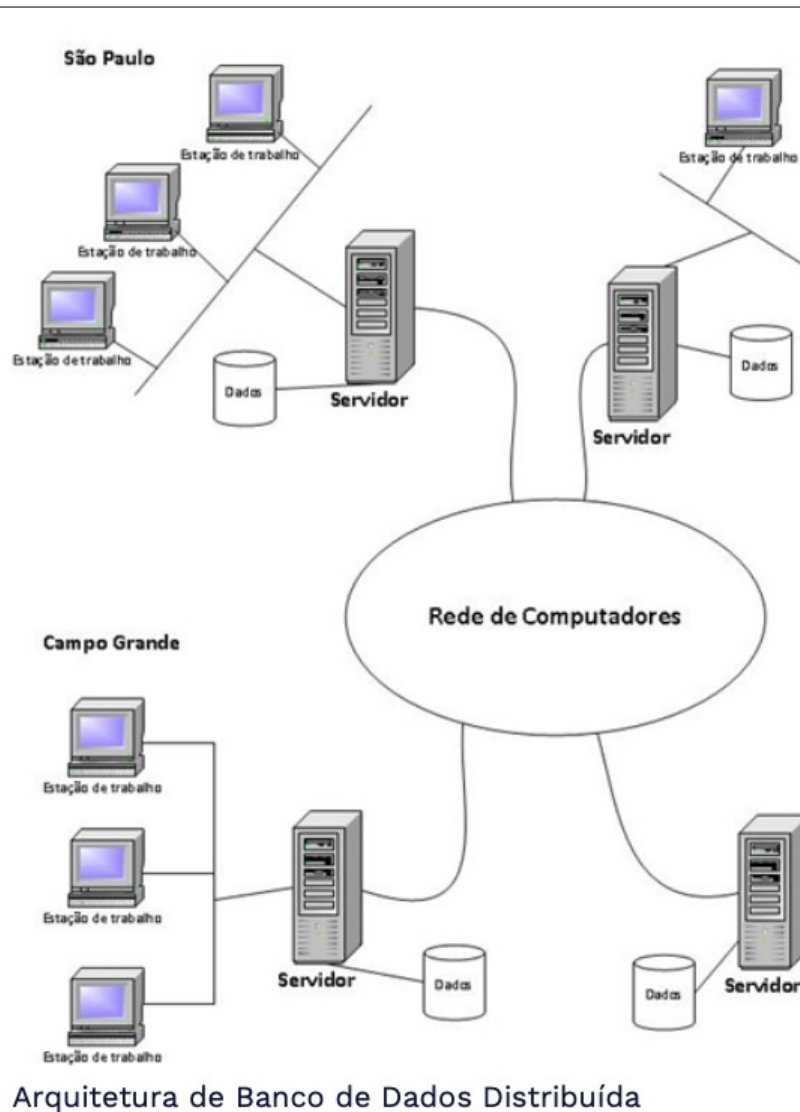
Aula 6 – Bancos de Dados NewSQL

Parte I - Conceitos

Bancos de Dados NewSQL

- Movimento relativamente recente pelo desenvolvimento de **BDs SQL de alto desempenho** visando o processamento OLTP eficiente de *Big Data*
 - novos SGBDs também baseados na **interface de acesso SQL**
- Eles são também chamados de ***scalable SQL*** ou ***newOLTP***
- BDs fortemente **distribuídos**
 - *data centers*, em alguns casos

BDD & Data Center



Diferenças em relação a BDDRs

- Maioria são **BDs em memória**
 - todo ou grande parte do BD é mantido em memória RAM
- **Adaptação de técnicas de gerenciamento de dados**
 - *scheduler, recovery*, particionamento de dados
- Modelo lógico relacional e interface de acesso SQL, porém, o **modelo físico não necessariamente é relacional**
 - exemplo: formato de armazenamento *chave-valor* é utilizado por alguns BDs NewSQL

NewSQL - Motivação

- BDRs tradicionais não são uma boa escolha para *Big Data* pela sua **difículdade em escalar horizontalmente**
- BDs NoSQL são uma boa escolha para aplicações *Big Data* que não se preocupam com **consistência eventual** (*read-intensive*)
- Mas e se a minha aplicação *Big Data* precisa de **consistência forte**?
 - BDs NoSQL não são uma boa escolha
 - grande maioria não suporta ACID, consistência de esquema e consultas complexas

NewSQL - Aplicações

- Aplicações com **alto processamento OLTP**
 - proliferação de aplicações Web em muitos tipos de dispositivos (*smartphones, tablets, notebooks, ...*) e utilizadas por um volume muito grande de pessoas
 - exemplos: automação bancária, *e-commerce*, ...
- Aplicações **de tempo real**
 - exigem dados consistentes que devem ser consumidos rapidamente
 - exemplo: bolsa de valores, jogos *online multiplayer*

NewSQL: O Melhor dos 2 Mundos

	BDR	BD NoSQL	BD NewSQL
Full SQL	suportado	não suportado	suportado
Arquitetura	tipicamente centralizada	tipicamente distribuída e na nuvem	tipicamente distribuída e na nuvem
Modelo de dados	relacional	chave-valor, colunar, documento, grafo	relacional
<i>Schemaless</i>	não	sim	não
Propriedades	ACID	BASE	ACID
Escalabilidade Horizontal	não suportado	suportado	suportado
Complexidade das Consultas	alta	baixa	alta
Processamento de dados	registros simples	Big Data	Big Data

NewSQL - Particionamento

- *Sharding*
 - ◆ particionamento horizontal de várias tabelas em vários nodos servidores de dados
 - ◆ SGBD tenta manter os *shards* balanceados
 - preferência pelo armazenamento do dado no(s) nodo(s) onde ele é mais frequentemente acessado (*workload*)
 - *alguns SGBDs NewSQL suportam **live migration***
 - *capacidade de mover dados entre nodos sem interromper o transações e o controle das propriedades ACID quando ocorrem mudanças no workload*
 - ◆ transações que desejam um certo dado irão encontrá-lo em um ou mais *shards* específicos
 - uso de catálogos e índices globais para rápida localização do dado no data center

NewSQL – *Scheduler*

- BDDRs tradicionais usam técnicas de coordenação (1) *centralizada* ou (2) *distribuída* de transações distribuídas
 - BDs NewSQL optam geralmente pela *técnica* (2) que dá mais autonomia aos nodos para gerenciarem suas transações
- Técnicas baseadas em *bloqueio (lock)* são *evitadas* devido à complexidade de gerenciamento de *locks* e *deadlocks* distribuídos
 - prefere-se técnicas que evitam *deadlocks*, como *Timestamp (TS)* e *Multiversão (MVCC)*

NewSQL – MVCC Schedulers

- Técnica mais utilizada pelos BDs NewSQL
 - *uma operação $\text{write}(x)$ de uma transação T_k gera uma nova versão x' e, enquanto T_k está ativa, outras transações podem ler a versão antiga (x) do dado, evitando bloqueios de transações que desejam apenas ler o dado. Se T_k commitar, então $x \leftarrow x'$; senão fica x*
 - vantagem: processa mais rápido que técnicas baseadas em *lock*
 - desvantagens: eventual *garbage collection* para versões antigas e *algoritmos de consenso* (reconciliação)
- BD Clustrix
 - Técnica híbrida 2PL + MVCC
 - define *locks* para atualização de dados, mas gera versões para permitir leitura do dado por outras transações



NewSQL – Replicação

- Replicação garante maior disponibilidade de dados
 - BDs NewSQL suportam consistência forte na gestão de réplicas
 - atualização de um dado X por uma transação T_k deve ser garantida em todas as réplicas que possuem X antes do *commit* de T_k

NewSQL – Replicação

- A maioria dos BDs NewSQL adota a seguinte estratégia
 - um dado X é atualizado inicialmente em um nodo e, posteriormente, o resultado da atualização é propagado para os demais nodos que possuem réplicas de X
 - evita o reproprocessamento da mesma operação sobre X em todas as réplicas

NewSQL – *Recovery*

- *Recovery* em BDDRs tradicionais
 - Se um nodo A falha, um nodo B assume o controle das suas transações distribuídas e as *commita* (ou aborta), caso ele garanta a execução com sucesso (ou não) dessas transações nos demais nodos envolvidos
 - O nodo A, ao “voltar à vida”, verifica a situação das suas transações distribuídas no *log* de B e se recupera
- BDs NewSQL adaptam essa técnica tradicional de *Recovery* distribuído, como por exemplo
 - ZooKeeper
 - serviço *open source* da Apache de alto desempenho para coordenação distribuída e confiável
 - modificações de algoritmos de consenso, como *Paxos* e *Raft*
 - uso de algoritmos paralelos de alto desempenho para apoiar as tarefas do novo gerente de transações B em decorrência de uma falha no nodo A

NewSQL – Principais SGBDs

 **H-Store**

VOLTDB


Clustrix

 **CockroachDB**

 **SingleStore**


nuodb[®]

UFSC-CTC-INE-PPGCC

INE 410131 – Gerência de Dados para Big Data

Aula 6 – Bancos de Dados NewSQL

Parte II - Modelagem

Projeto Lógico de BD NewSQL

- Projeto de BDR tradicional centralizado
 - definir e estruturar dados persistentes relevantes para um domínio de aplicação
 - levantamento de requisitos, modelagem conceitual, modelagem lógica e modelagem física
- Projeto de BD NewSQL → Projeto BDD
 - modelagem lógica
 - definir adicionalmente a **alocação** do esquema lógico (tabelas) nos BDs dos nodos servidores de dados
 - decisão sobre quais dados serão armazenados em quais nodos
 - leva em conta **fragmentação** e **replicação** de dados

Projeto Lógico de BDR

Centralizado

- Compromisso entre
 - evitar um grande número de tabelas
 - evitar um tempo longo de resposta nas consultas e atualizações de dados
 - implica minimizar junções entre tabelas
 - evitar atributos opcionais
 - evitar tabelas subutilizadas
 - implica evitar desperdício de espaço
 - evitar muitos controles de integridade no BD
 - evitar organizações de dados em tabelas que gerem muitos controles de integridade
 - implica evitar muitas dependências entre dados

Processo de Mapeamento

1. Mapeamento preliminar de entidades e seus atributos
2. Mapeamento de especializações
3. Mapeamento de relacionamentos e seus atributos

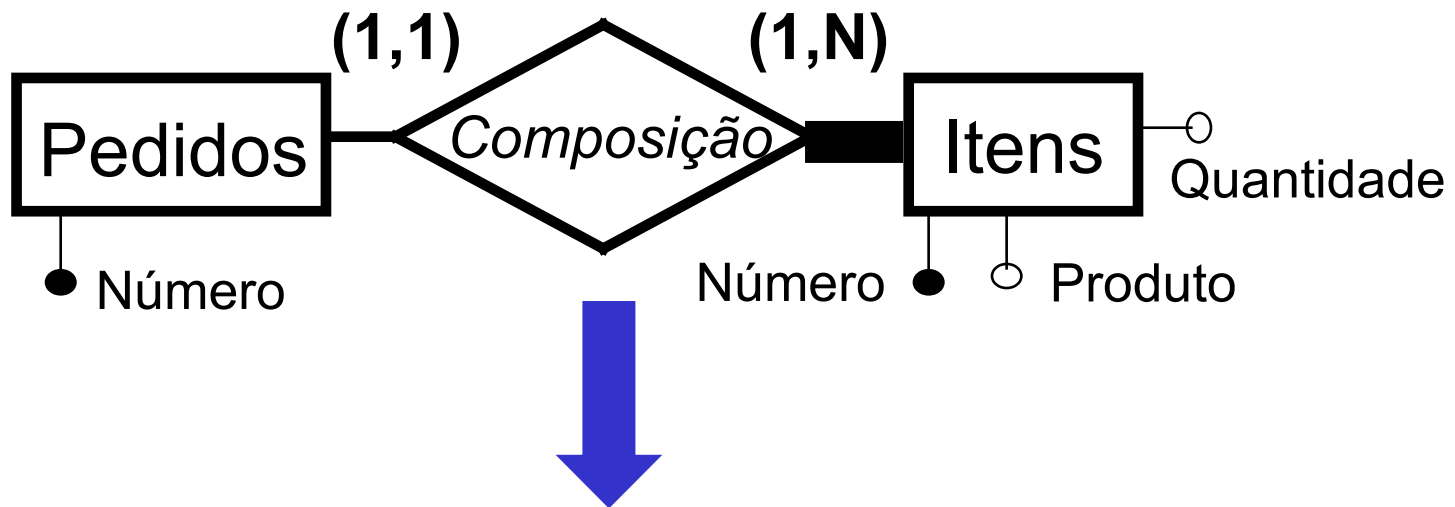
Mapeamento de Entidades



Empregados (CPF, Nome, Idade)

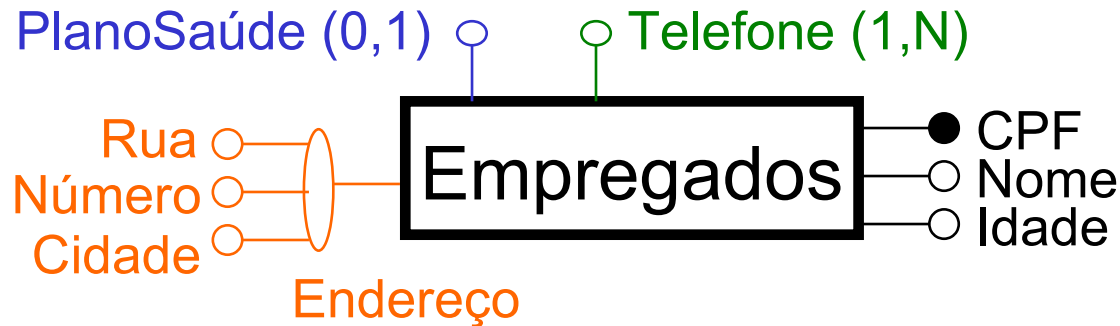
Mapeamento de Entidades Fracas

- Identificador da entidade forte torna-se
 - parte da chave primária na *tabela* correspondente à entidade fraca (*tabelaFraca*)
 - chave estrangeira na *tabelaFraca*



Itens ([NroPedido](#), [NroItem](#), Produto, Quantidade)

Mapeamento de Atributos



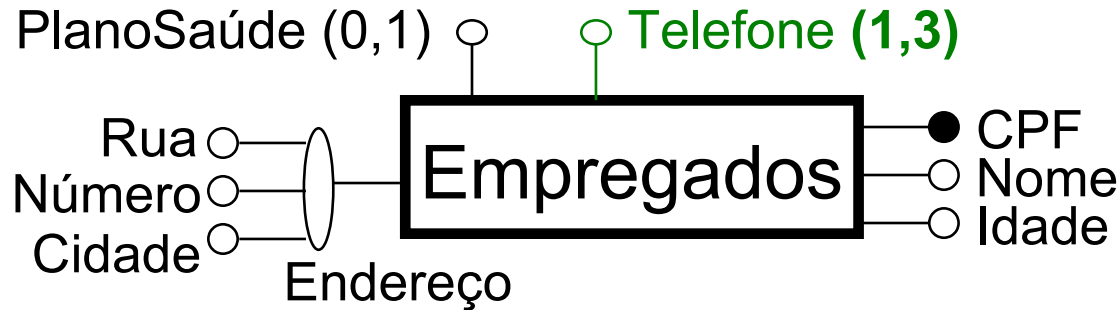
Empregados (CPF, Nome, Idade, **PlanoSaúde**,
Rua, **Número**, **Cidade**)

Telefone(CPF, Número)

ou

Telefone (CPF, Número)

Mapeamento de Atributos



Empregados (CPF, Nome, Idade, PlanoSaúde, Rua, Número, Cidade, FoneRes, FoneCom, Celular)

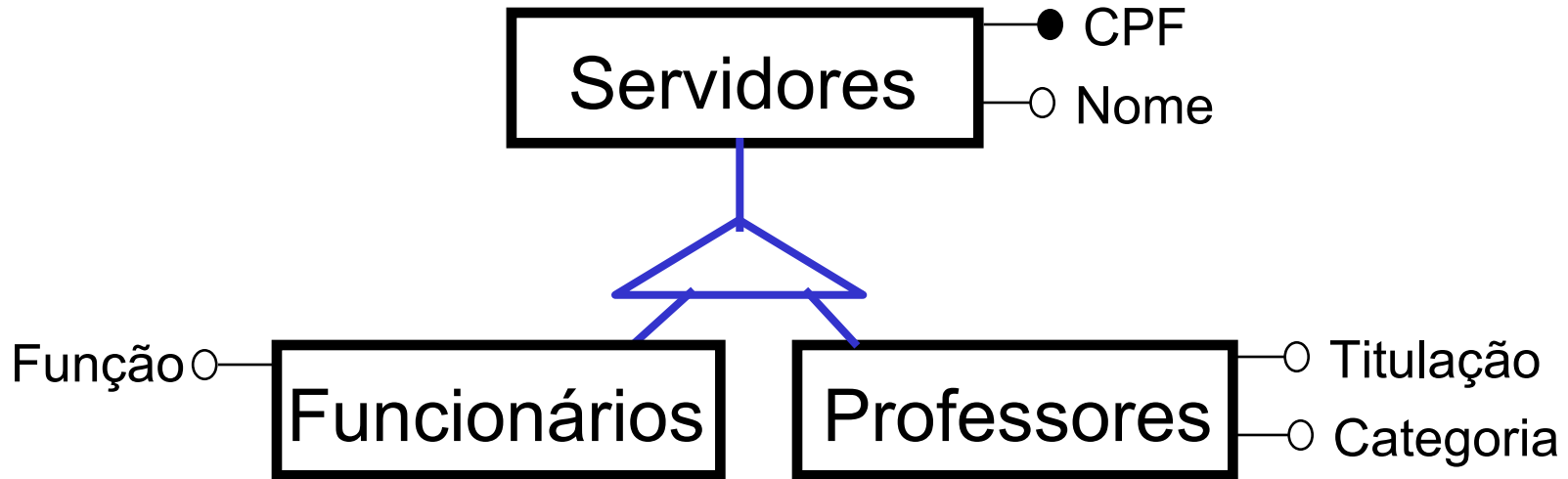
Processo de Mapeamento

1. Mapeamento preliminar de entidades e seus atributos
2. Mapeamento de especializações
3. Mapeamento de relacionamentos e seus atributos

Mapeamento de Especializações

- Três alternativas são geralmente adotadas
 1. **tabela única** para entidade genérica e suas especializações
 2. tabelas para a **entidade genérica** e as **entidades especializadas**
 3. tabelas apenas para as **entidades especializadas**

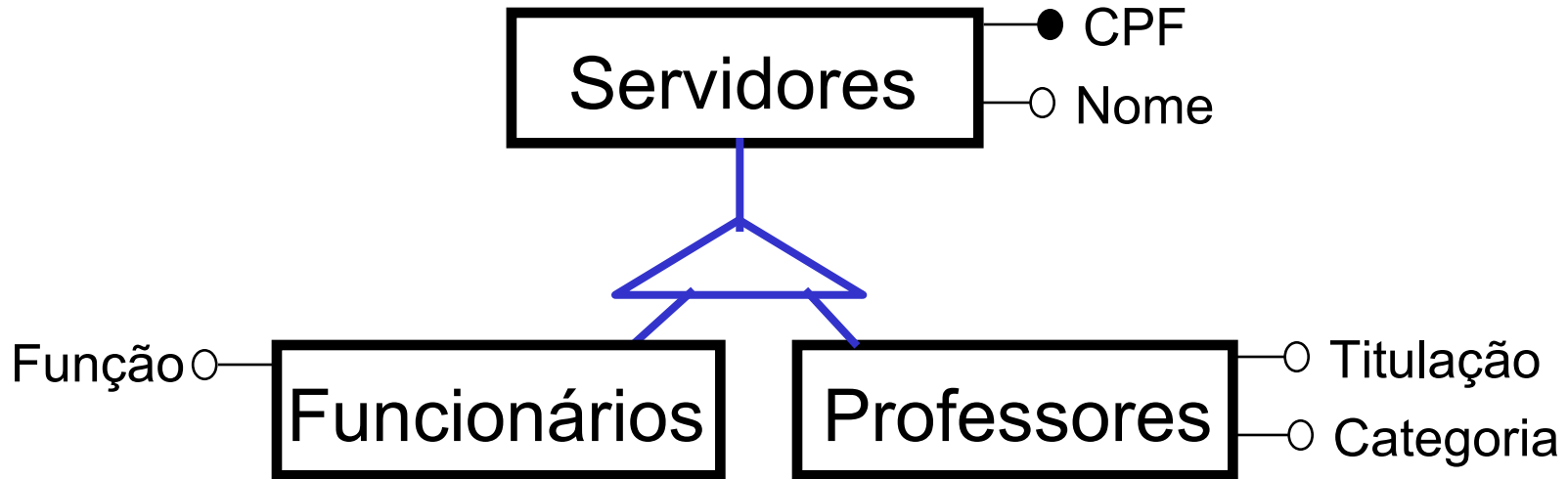
Alternativa 1



Servidores (CPF, Nome, Tipo, Função, Titulação, Categoria)

- **Tipo** pode assumir mais de um valor se a especialização é não-exclusiva

Alternativa 2

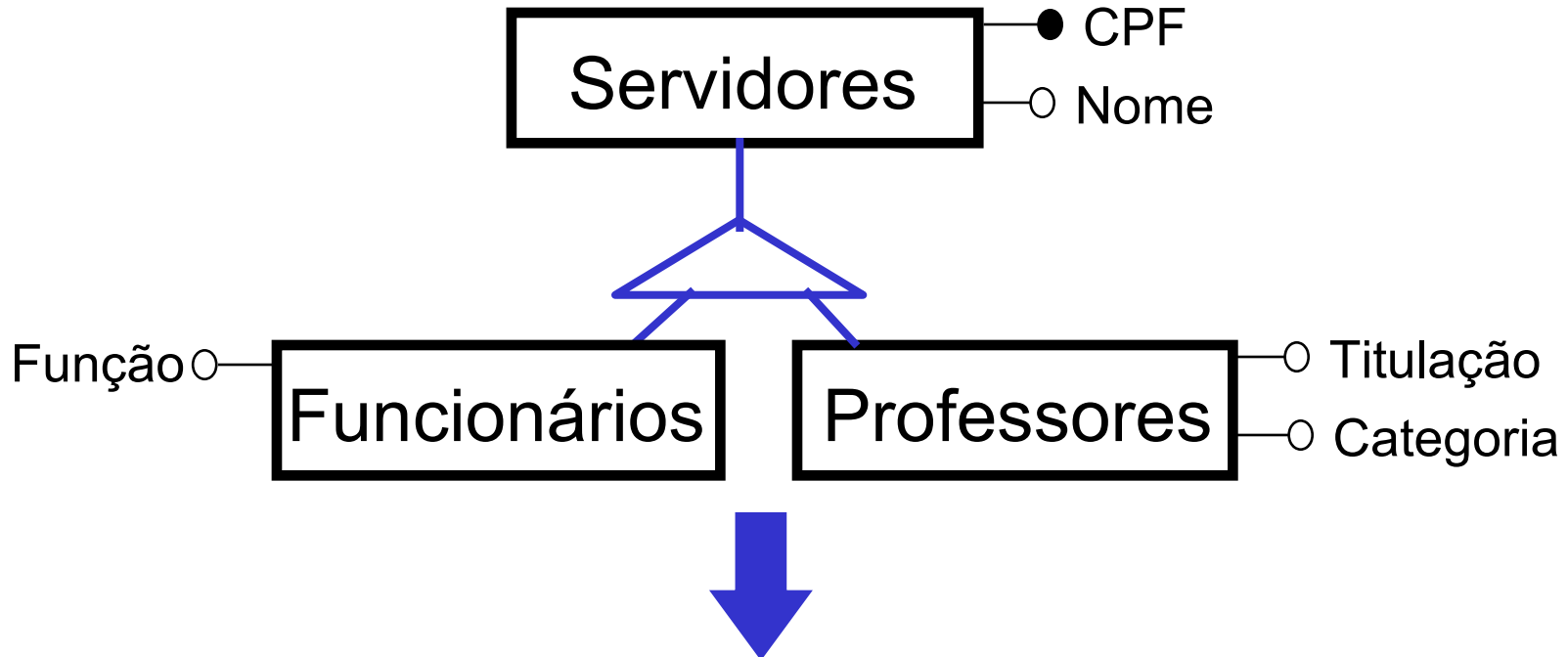


Servidores (CPF, Nome)

Funcionários (CPF, Função)

Professores (CPF, Titulação, Categoria)

Alternativa 3



Funcionários (CPF, Nome, Função)

Professores (CPF, Nome, Titulação, Categoria)

- Não se aplica a especializações parciais

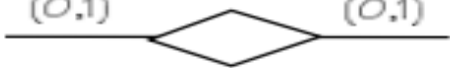
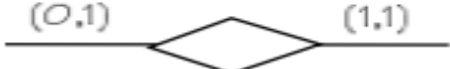

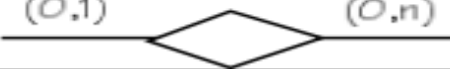
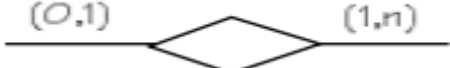

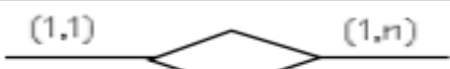
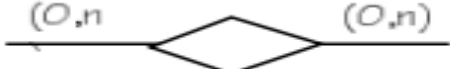
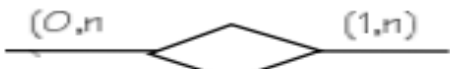
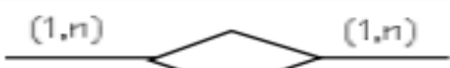
Processo de Mapeamento

1. Mapeamento preliminar de entidades e seus atributos
2. Mapeamento de especializações
3. Mapeamento de relacionamentos e seus atributos

Mapeamento de Relacionamentos

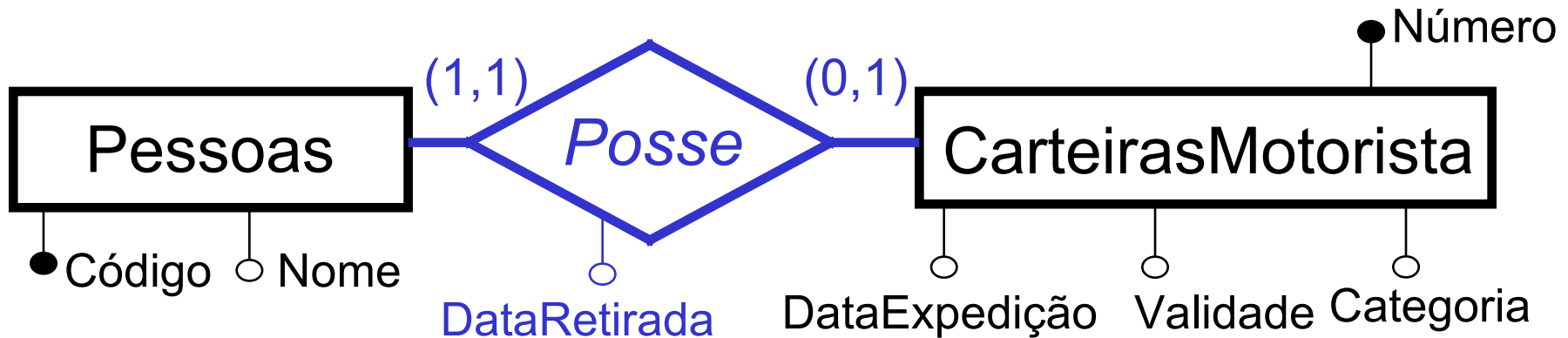
- Recomendações de mapeamento baseiam-se na **análise da cardinalidade** dos relacionamentos
 - com base nesta análise, algumas alternativas de mapeamento podem ser adotadas
 1. **entidades relacionadas** podem ser **fundidas** em uma única tabela (**fusão**)
 2. **tabelas** podem ser criadas para o relacionamento
 3. **chaves estrangeiras** podem ser criadas em tabelas a fim de representar adequadamente o relacionamento

Mapeamento de Relacionamentos

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
Relacionamentos 1:1			
	±	✓	×
	×	±	✓
	×	±	✓
Relacionamentos 1:n			
	±	✓	×
	±	✓	×
	×	✓	×
	×	✓	×
Relacionamentos n:n			
	✓	×	×
	✓	×	×
	✓	×	×
✓ Alternativa preferida ± Pode ser usada × Não usar			

Relacionamento 1-1 - Exemplo

- Opcional em um dos sentidos

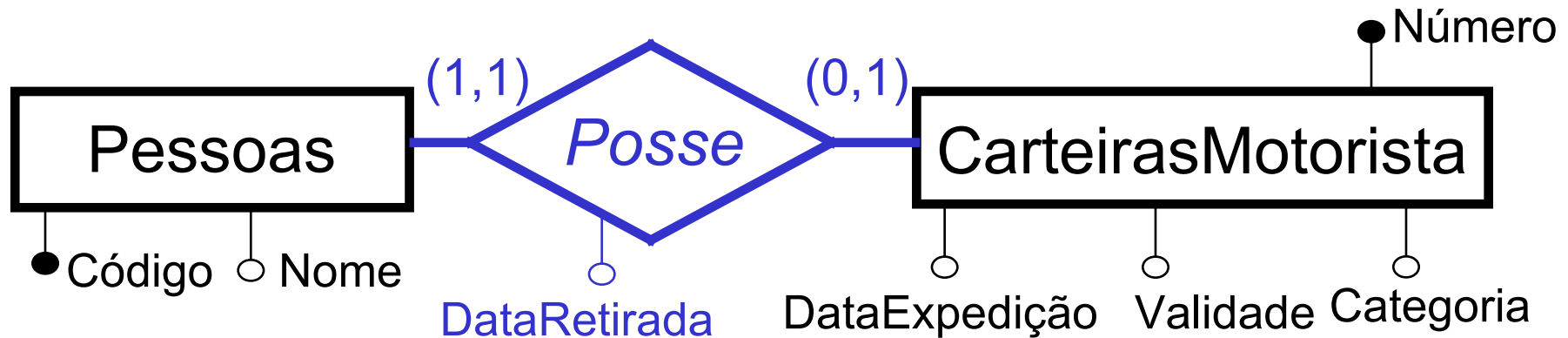


alternativa 1

Pessoas (Código, Nome, NúmeroCarteiraMotorista, DataExpedição, Validade, Categoria, DataRetirada)

Relacionamento 1-1 - Exemplo

- Opcional em um dos sentidos



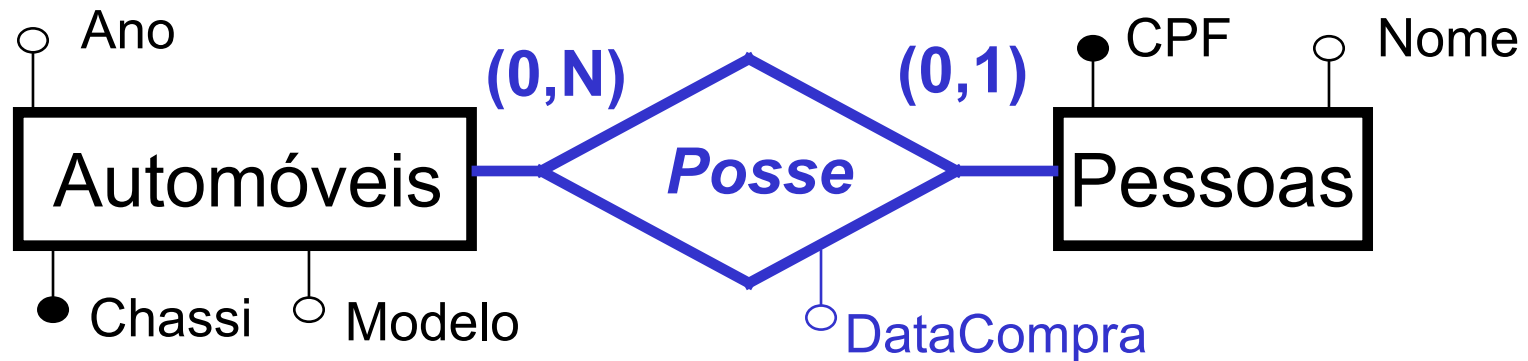
alternativa 2

Pessoas (Código, Nome)

CarteirasMotorista (Número, DataExpedição, Validade, Categoria, Código, DataRetirada)

Relacionamento 1-N - Exemplo

- Opcional no “lado 1”



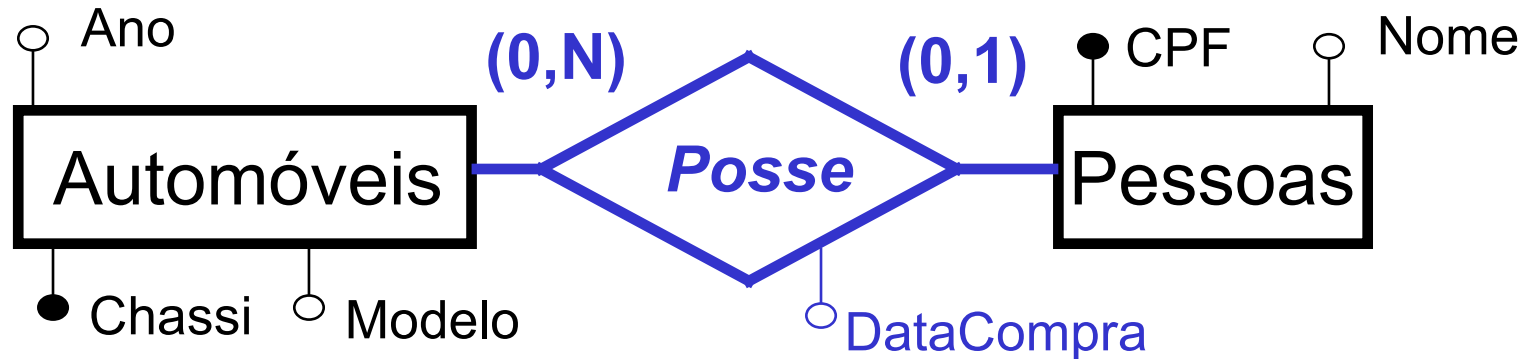
Pessoas (CPF, Nome)

Automóveis (Chassi, Modelo, Ano)

Posse (CPF, Chassi, DataCompra)

Relacionamento 1-N

- Opcional no “lado 1”



alternativa 2

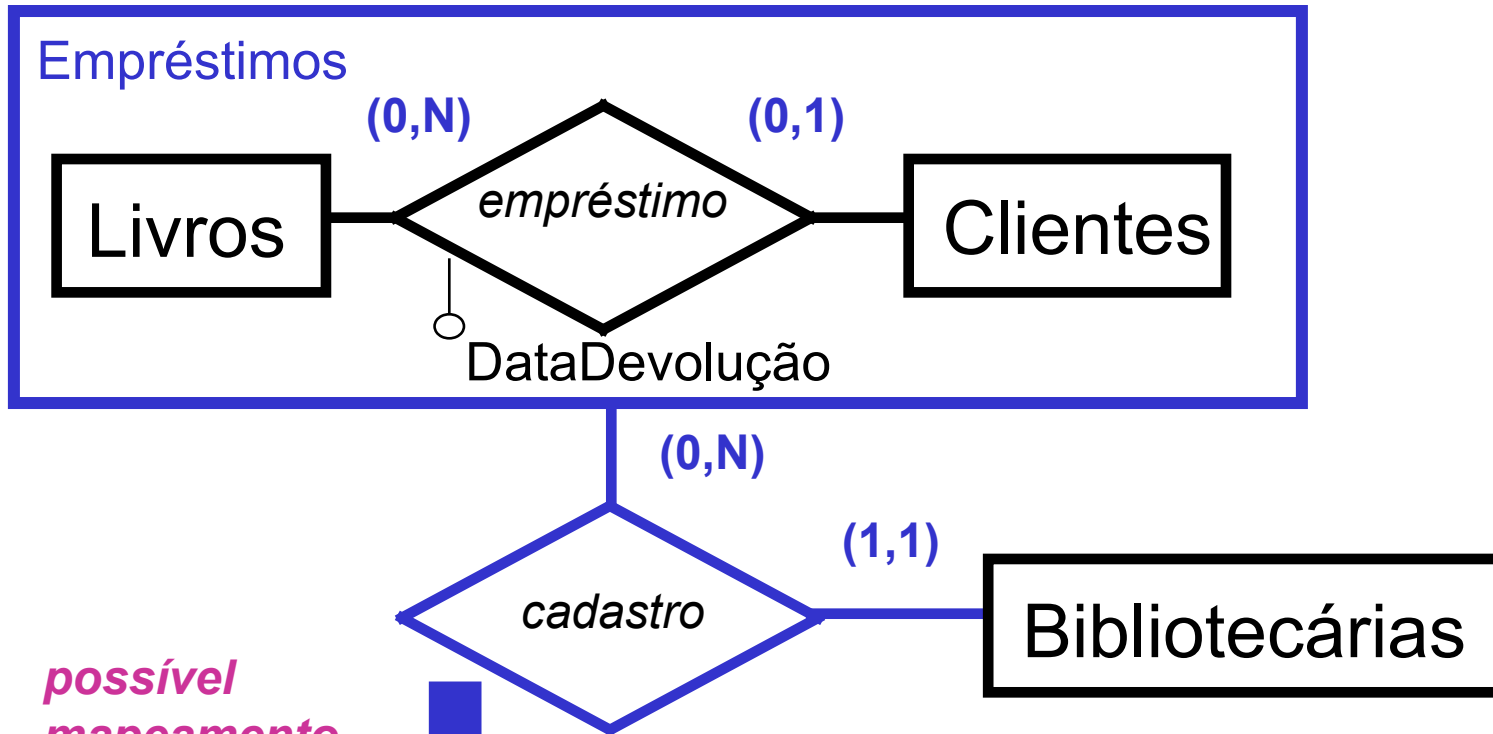
Pessoas (CPF, Nome)

Automóveis (Chassi, Modelo, Ano, CPF, DataCompra)

Relacionamentos com Entidades Associativas

1º) mapear relacionamento interno

2º) mapear relacionamento(s) externo(s)



Livros (Código, ..., CPFcli, DataDevolução, CPFbibl)

Clientes (CPFcli, ...)

Bibliotecárias(CPFbibl, ...)

Projeto BD Distribuído (BDD)

- Estratégia para **alocação** dos dados
- Duas questões a tratar
 - Projeto de **Fragmentação**
 - Projeto de **Replicação**

Projeto BDD

(Esquema de Alocação)

- Considera basicamente
 - metas de **desempenho** no acesso ao BDD
 - trade-off: rapidez nas atualizações (baixa distribuição) X disponibilidade (alta distribuição), ...
 - **freqüência de transações** em cada nodo (análise do *workload*)
 - pode ser o gargalo do BDD, se distribuição foi mal definida (exemplo: muitos dados concentrados em um nodo ou a maioria dos dados que eu geralmente preciso estão em outros nodos)

Projeto BDD - Fragmentação

- Separação dos dados de uma tabela para armazenamento em mais de um nodo
 - definição de um esquema de fragmentação
- Tipos de fragmentação
 - horizontal, vertical e mista

Fragmentação Horizontal (FH)

- Separação de uma tabela T em nível de tupla
- Cada **fragmento horizontal** fh_i de T ($fh_i(T)$) é definido através de uma seleção
 - $fh_i(T) = \sigma_c(T)$
- T é obtida através da **união** de todos os seus fragmentos
 - $T = fh_1(T) \cup fh_2(T) \cup \dots \cup fh_n(T)$
- FH com **fragmentação derivada**
 - tuplas de uma tabela secundária T' (com referência à T) são também fragmentadas

FH - Exemplo

Filiais

número	cidade	endereço
1	Fpolis	rua X, 10
2	Blumenau	rua H, 55
3	Blumenau	rua E, 18
4	Fpolis	rua K, 87
5	Fpolis	rua Q, 52

Funcionários

código	nome	endereço	DN	cargo	salário	filial
1	João	rua X, 10	11/11/70	vendedor	1500,00	1
2	Maria	rua H, 55	12/04/71	caixa	1200,00	1
3	Paulo	rua E, 18	13/08/72	vendedor	1300,00	1
4	Carlos	rua K, 87	14/01/73	caixa	1000,00	2
5	Ana	rua Q, 52	15/05/74	vendedor	1300,00	2
...						

fh_1

$\sigma_{\text{cidade} = \text{'Blumenau'}}$ (Filiais)

número	cidade	endereço
2	Blumenau	rua H, 55
3	Blumenau	rua E, 18

fh_2 (derivada para Funcionários)

$\sigma_{\text{cidade} = \text{'Fpolis'}}$ (Filiais)

número	cidade	endereço
1	Fpolis	rua X, 10
4	Fpolis	rua K, 87
5	Fpolis	rua Q, 52

código	nome	endereço	...	filial
1	João	rua X, 10		1
2	Maria	rua H, 55		1
3	Paulo	rua E, 18		1
...				

Fragmentação Vertical (FV)

- Separação de T em nível de atributo
- Cada **fragmento vertical** fv_i de T ($fv_i(T)$) é definido através de uma projeção
 - $fv_i(T) = \pi_{a_1, \dots, a_j}(T)$
- T é obtida através da **junção** de todos os seus fragmentos
 - $T = fv_1(T) \bowtie fv_2(T) \bowtie \dots \bowtie fv_n(T)$
 - requer a mesma **chave candidata** em todos os fragmentos

FV - Exemplo

código	nome	endereço	DN	cargo	salário	filial
1	João	rua X, 10	11/11/70	vendedor	1500,00	1
2	Maria	rua H, 55	12/04/71	caixa	1200,00	1
3	Paulo	rua E, 18	13/08/72	vendedor	1300,00	1
4	Carlos	rua K, 87	14/01/73	caixa	1000,00	2
5	Ana	rua Q, 52	15/05/74	vendedor	1300,00	2
...						

Funcionários

fv_1 (dados pessoais)

$\pi_{\text{código, nome, endereço, DN}}$ (Funcionários)

código	nome	endereço	DN
1	João	rua X, 10	11/11/70
2	Maria	rua H, 55	12/04/71
3	Paulo	rua E, 18	13/08/72
4	Carlos	rua K, 87	14/01/73
5	Ana	rua Q, 52	15/05/74
...			

fv_2 (dados profissionais)

$\pi_{\text{código, cargo, salário, filial}}$ (Funcionários)

código	cargo	salário	filial
1	vendedor	1500,00	1
2	caixa	1200,00	1
3	vendedor	1300,00	1
4	caixa	1000,00	2
5	vendedor	1300,00	2
...			

Fragmentação Mista (FM)

- Separação de T em nível de tupla e atributo
- T é obtida através da execução de operações de reconstrução de fragmentos horizontais e verticais
 - exemplo
 1. Funcionários são separados por filial
 2. para cada filial, separar dados pessoais e profissionais de funcionários
 - ordem de reconstrução:
 1. junção dos FVs de funcionários em cada filial
 2. união de dados de funcionários por filial

Projeto BDD - Replicação

- Definição dos nodos onde serão armazenados os fragmentos (ou tabelas completas)
 - definição de associações fragmento – nodo
- Se associação é Fragmento $[1,N]$ – 1 Nodo
 - não há replicação
- Se associação é Fragmento $N - M$ Nodo
 - há replicação
- Possibilidades de replicação
 - total, nula ou parcial

Possibilidades de Replicação

- Total
 - desempenho bom para consultas
 - não há necessidade de acesso remoto
 - muita redundância de dados e desempenho ruim para atualizações
 - manutenção de cópias consistentes
 - *scheduler* e *recovery* mais complexos
 - bloqueios em todos os nodos
 - UNDO e REDO em todos os nodos
- Nula
 - inverte-se as vantagens e desvantagens
- Parcial
 - meio termo entre as opções anteriores

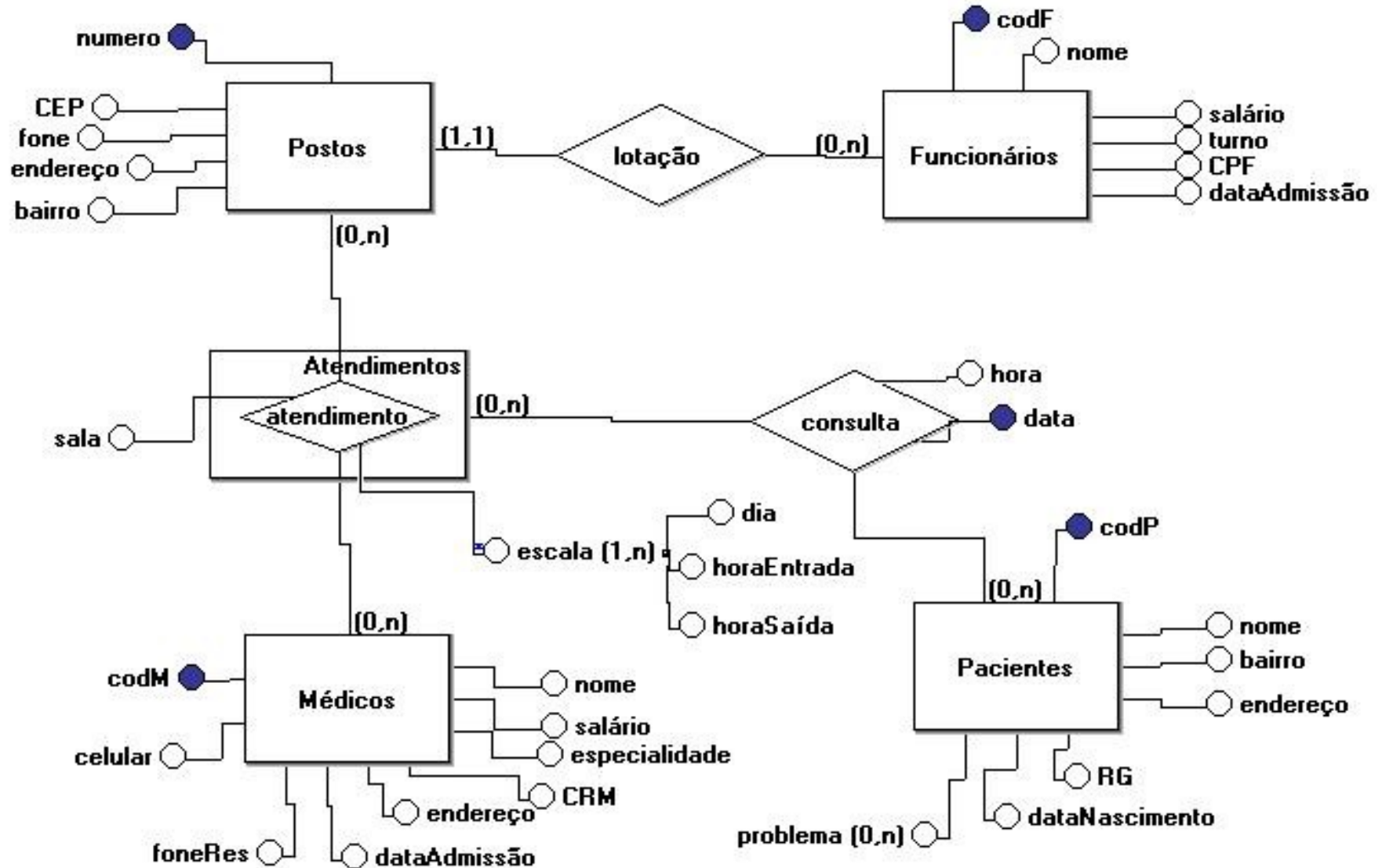
Projeto do Esquema de Replicação

- Considerações sobre replicação
 - deseja-se alta disponibilidade; transações desejam dados que podem estar em qq nodo; grande parte das transações é de leitura
 - replicação (próxima da) total
 - transações que acessam determinados dados partem geralmente dos mesmos nodos
 - replicação parcial destes dados nestes nodos
 - atualizações ocorrem em dados cadastrados localmente
 - replicação nula (apenas fragmentação dos dados de interesse local)

Estudo de Caso de Projeto BDD: Clínica

“Uma **clínica** de uma cidade possui um posto matriz no centro e outros postos em bairros. No posto matriz fica o departamento pessoal. Cada posto tem um *código, rua, número, bairro, CEP e fone*. A clínica emprega médicos e funcionários e presta serviço a pacientes através de consultas com médicos (consultas marcadas devem ser mantidas no BD). Um funcionário trabalha em um posto e possui um *código, nome, CPF, salário, função, data de admissão e turno de trabalho*. Médicos dão atendimento em um certo subconjunto de postos (com uma escala semanal de horários predefinida em cada posto, atendendo em uma sala do posto). Um médico tem *especialidade, código, CRM, nome, salário, endereço, fone residencial e celular para contato e data de admissão*. Os postos oferecem atendimento para todas as especialidades que a clínica suporta. Apenas pacientes que residem na cidade tem direito a consultar nos postos, devendo se dirigir ao posto do seu bairro. Para todo paciente cadastra-se um *código, nome, rua, número, bairro, RG, data de nascimento e eventual(is) problema(s)*.”

Estudo de Caso - Esquema ER



Estudo de Caso - Esquema BDR

Postos (nroPosto, CEP, fone, endereço, bairro)

Funcionários (codF, nome, CPF, salário, função, admissão, turno, *nroPosto*)

Pacientes (codP, nome, rua, número, bairro, RG, DN)

Problemas (codP, descrição)

Médicos (codM, CRM, nome, especialidade, admissão, salário, endereço, foneRes, celular)

Atendimentos (codM, nroPosto, sala)

Escalas (codM, nroPosto, dia, hora-Início, hora-Término)

Consultas (codM, nroPosto, codP, data, hora)

Estudo de Caso – Projeto de Alocação

- 1) Dados de postos, médicos e seus atendimentos e escalas, funcionários, consultas, pacientes e seus problemas: **fragmentados por posto** (supõe-se um nodo por posto)
- 2) Apenas alguns dados de Médicos são necessários em cada posto: **nome, codM, especialidade, endereço, celular, foneRes** (idem para Funcionários: **codF, nome, função, turno**)
- 3) Dados de médicos **replicados nos postos** em que trabalham
- 4) **Posto matriz mantém dados completos de médicos, funcionários e postos**



Para cada Posto “X”:

FragPostoX $\leftarrow \sigma_{\text{bairro} = \text{“X”}}$ (Postos)

FragPacX $\leftarrow \sigma_{\text{bairro} = \text{“X”}}$ (Pacientes)

FragProbPacX \leftarrow Problemas $\bowtie \pi_{\text{codP}}$ (FragPacX)

FragFuncX $\leftarrow \pi_{\text{codf}, \dots, \text{turno}, \text{nroPosto}}$ (Funcionários) $\bowtie \pi_{\text{nroPosto}}$ (FragPostoX)

FragAtendX \leftarrow Atendimentos $\bowtie \pi_{\text{nroPosto}}$ (FragPostoX)

FragEscalasX \leftarrow Escalas $\bowtie \pi_{\text{nroPosto}}$ (FragPostoX)

FragConsultasX \leftarrow Consultas $\bowtie \pi_{\text{nroPosto}}$ (FragPostoX)

FragMedX $\leftarrow \pi_{\text{nome}, \text{codM}, \dots, \text{foneRes}}$ (Médicos) $\bowtie \pi_{\text{codM}}$ (FragAtendX)

Atividade 5 – Projeto BD NewSQL

Proponha um **projeto lógico relacional centralizado** para o domínio de **Jogos Online** mostrado abaixo, seguido de um **projeto de alocação**

