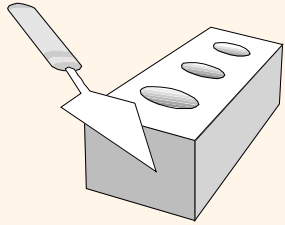
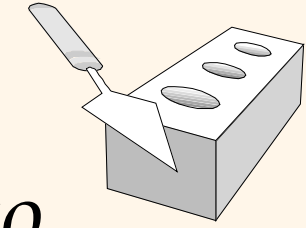


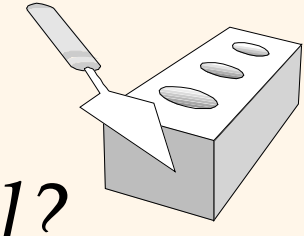
O Modelo Relacional

Capítulo 3



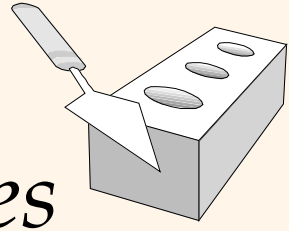
Modelo Relacional - Breve Histórico

- Proposto por Codd em 1970
- Superou os modelos mais antigos: redes e hierárquico
  - representação simples dos dados
  - consultas complexas podem ser feitas de maneira simples
- Meados anos 70: 1os. SGBD relacionais foram desenvolvidos em projetos da IBM e UC-Berkeley
- Desde então vários produtos relacionais têm sido oferecidos, representando um mercado milionário



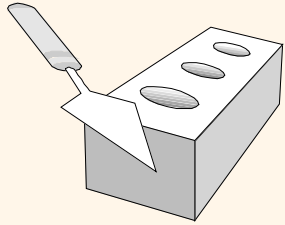
Por que Estudar o Modelo Relacional?

- Porque é o modelo mais usado
  - Fornecedores: IBM, Informix, Microsoft, Oracle, Sybase, etc.
- Ainda existem "Sistemas Legados" nos modelos antigos
  - Exemplo: IBM's IMS (hierárquico); IDS e IDMS (redes)
- Competidor recente: modelo orientado a objetos
  - ObjectStore, Versant, Ontos
  - Uma combinação: *modelo objeto-relacional*
    - Informix Universal Server, UniSQL, O2, Oracle, DB2



Banco de Dados Relacional: Definições

- *Banco de dados relacional:* um conjunto de *relações* com nomes distintos
- *Relação:* uma tabela com linhas e colunas. É composta por 2 partes:
  - *Esquema* : especifica o nome da relação e o nome e o tipo (domínio) de cada coluna (campo ou atributo).
    - Exemplo: Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).
  - *Instância*: é o conjunto de registros que compõem a tabela num dado momento; é variável; deve obedecer o esquema da relação
    - # linhas = *cardinalidade*, # campos = *grau*.
- Requerimento do modelo: uma relação é um conjunto de linhas únicas*(ou tuplas ou registros)* (i.e., **todas as linhas são distintas**).



Exemplo de Instância da Relação

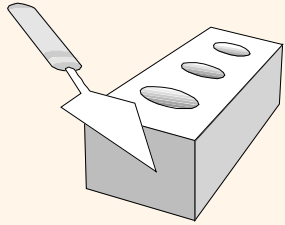
Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

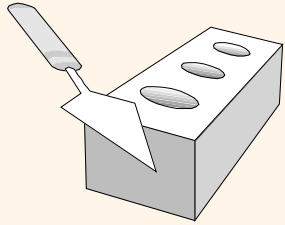
- Cardinalidade = 3; Grau = 5; todas linhas diferentes

- Todas as instâncias de uma coluna da relação têm que ser diferentes?

Linguagens de Consulta Relacionais

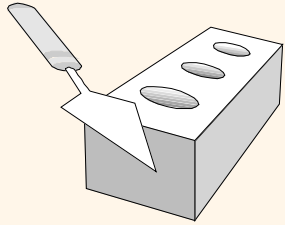


- Facilidade de obter a informação armazenada no banco geralmente determina seu valor
- Uma importante característica do modelo relacional: suporte a consultas de dados simples mas poderosas.
- Consultas podem ser escritas de forma intuitiva e o SGBD é responsável por sua avaliação para execução eficiente.
  - A chave para isso é ter uma semântica precisa para consultas relacionais.
  - Possibilita que o otimizador amplamente reordene operações, assegurando que as respostas não se alterem.
- Linguagens de consulta relacionais: Álgebra relacional; Cálculo relacional; SQL



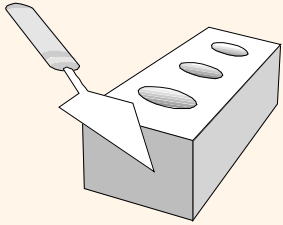
A Linguagem SQL

- SQL - Structured Query Language
- Desenvolvida nos anos 70 pela IBM como linguagem de consulta para o Sistema-R
- Inicialmente para consulta, tornou-se a linguagem mais usada para criação, manipulação e consultas em SGBD relacionais



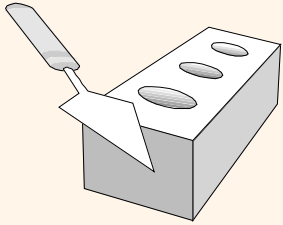
A Linguagem SQL

- Definição de SQL padrão: decorrente da sua utilização por vários fornecedores
  - possibilita avaliar a completitude da SQL oferecida
  - possibilita distinção entre características específicas de produtos
  - portabilidade: produtos que se baseiam em características padrão são mais portáveis
- Padrões:
  - SQL-86 (ANSI)
  - SQL-89 (revisão pouco expressiva)
  - SQL-92 (revisão significativa; ANSI/ ISO)
  - SQL-99 (extensões significativas, padrão atual)



A Linguagem SQL

- Usa a palavra *TABLE* para referenciar a relação
- Linguagem de Definição de Dados (DDL): subconjunto da SQL que suporta a criação, exclusão e modificação de tabelas
- Linguagem de Manipulação de Dados (DML): subconjunto da SQL que suporta a modificação e a recuperação de dados



Consultando dados em SQL

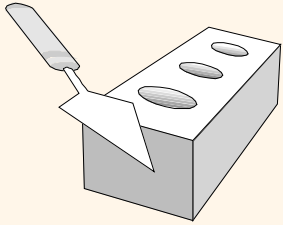
- Comando: SELECT
- Para achar todos os estudantes de 18 anos, podemos escrever:

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- Para obter apenas *name* e *login*, substituir a primeira linha por:

```
SELECT S.name, S.login
```



Consultando várias relações

- Qual é o resultado da consulta abaixo, dada a instância de Enrolled?

```
SELECT  S.name, E.cid
FROM  Students S, Enrolled E
WHERE  S.sid=E.sid AND E.grade="B"
```

Students

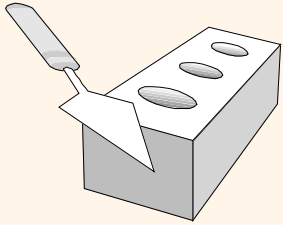
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

obtemos:

S.name	E.cid
Jones	History105



Criando Relações em SQL

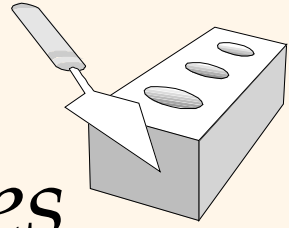
- Comando: CREATE TABLE

```
CREATE TABLE Students
    (sid: CHAR(20),
     name: CHAR(20),
     login: CHAR(10),
     age: INTEGER,
     gpa: REAL)
```

- Cria a relação **Students**. O tipo (domínio) de cada campo é especificado e validado pelo SGBD sempre que as tuplas são adicionadas ou modificadas.

```
CREATE TABLE Enrolled
    (sid: CHAR(20),
     cid: CHAR(20),
     grade: CHAR(2))
```

- Como outro exemplo, a tabela **Enrolled** mantém informação sobre os cursos nos quais os estudantes se matriculam.



Destruindo e Alterando Relações

- Destruir: comando DROP TABLE

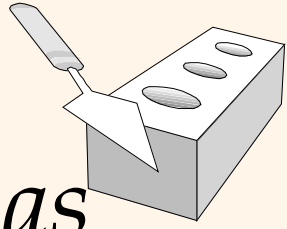
```
DROP TABLE Students
```

- Destrói a relação **Students**. A informação sobre o **esquema** e as **tuplas** são removidas.

- Alterar esquema: comando ALTER TABLE

```
ALTER TABLE Students
        ADD COLUMN firstYear: integer
```

- O esquema de **Students** é alterado ao ser adicionado um novo campo; cada tupla na instância corrente é estendida com um valor null no novo campo.



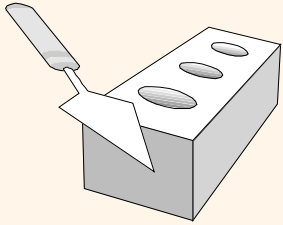
Adicionando e Removendo Tuplas

- Adicionar: Comando INSERT INTO
- Pode-se inserir uma única tupla por meio de:

```
INSERT INTO  Students (sid, name, login, age, gpa)
VALUES  (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- Remover: Comando DELETE FROM
- Pode-se remover todas as tuplas que satisfaçam alguma condição (ex.: name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

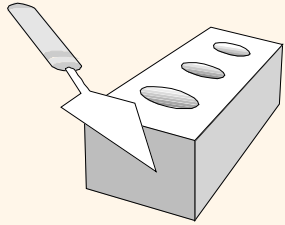


Alterando Tuplas

- Comando UPDATE FROM
- Pode-se modificar os valores das colunas de uma tupla específica ou de várias delas por meio de:

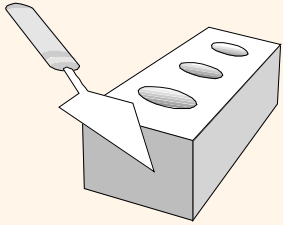
```
UPDATE
FROM Students S
SET S.age = S.age + 1, S.gpa = S.gpa - 1
WHERE S.sid = 53688
```

```
UPDATE
FROM Students S
SET S.gpa = S.gpa + 0.5
WHERE S.gpa >= 3.2
```



Restrições de Integridade (RIs)

- Um bom SGBD deve evitar a entrada de informação incorreta
- RI: condição que deve ser verdadeira para qualquer instância da base de dados; ex.: restrições de domínio.
  - RIs são especificadas quando o esquema é definido.
  - RIs são conferidas quando relações são modificadas.
- Uma instância válida de uma relação é aquela que satisfaz todas as RIs especificadas relacionadas a ela.
  - SGBD não deve permitir instâncias inválidas.
- Se um SGBD confere RIs, os dados armazenados são mais próximos do significado do "mundo real".

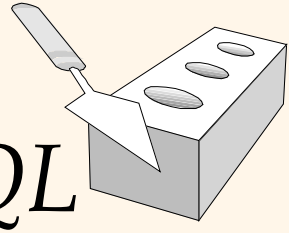


Restrições de Chave

- Toda relação tem no mínimo uma chave
- Um conjunto de campos é uma chave candidata para uma relação se:
  1. Não existem 2 tuplas diferentes com os mesmos valores para estes campos, ou seja, ela identifica unicamente qualquer tupla da relação
  2. Se retirarmos qualquer atributo da chave, ela deixa de identificar unicamente as tuplas. ⇒ isto é falso? então a chave é uma super-chave

  Ex.: *sid* é uma chave para **Students**. (E *name*?)

  O conjunto {*sid, gpa*} é uma super-chave.
- Se há mais de 1 chave para uma relação, uma das chaves é escolhida (pelo DBA) para ser a chave primária (que não pode ter valor null).



Chaves Primárias e Candidatas em SQL

- Possivelmente há várias chaves candidatas (especificadas usando UNIQUE), entre as quais apenas uma é escolhida como chave primária (PRIMARY KEY).

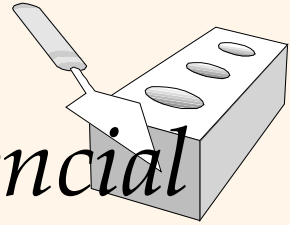
```
CREATE TABLE Enrolled
  (sid CHAR(20)
    cid  CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid,cid) )
```

vs.

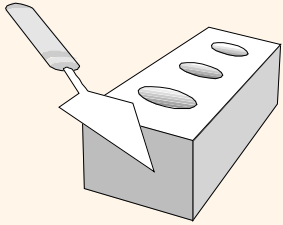
```
CREATE TABLE Enrolled
  (sid CHAR(20)
    cid  CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid),
    UNIQUE (cid, grade) )
```

- "Para um determinado estudante e curso, há uma única nota." vs. "Estudantes podem cursar apenas um curso e receber uma única nota para aquele curso; ainda, dois estudantes em um curso não recebem a mesma nota."
- Usada de forma displicente, uma restrição de integridade pode impedir o armazenamento de instâncias de base de dados que surgem na prática.

Chaves Estrangeiras, Integridade Referencial



- Chave estrangeira:
  - Conjunto de campos em uma relação que é usado para fazer referência a uma tupla em outra relação. (Deve corresponder à chave primária da segunda relação)
  - Funciona como um 'ponteiro lógico'.
- No exemplo *sid* de Enrolled é uma chave estrangeira que se refere à relação Students:
  - Enrolled(*sid*: string, *cid*: string, *grade*: string)
  - Se todas as restrições de chave estrangeira são garantidas, a integridade referencial é alcançada, i.e., não há referência pendente.
- Você pode citar um modelo de dados sem integridade referencial?
  - Links HTML



Chave Estrangeira em SQL

- Apenas estudantes listados na relação Students podem fazer matrícula em cursos.

```
CREATE TABLE Enrolled
  (sid CHAR(20),  cid CHAR(20),  grade CHAR(2),
    PRIMARY KEY  (sid,cid),
    FOREIGN KEY (sid) REFERENCES Students )
```

chave estrangeira

Enrolled

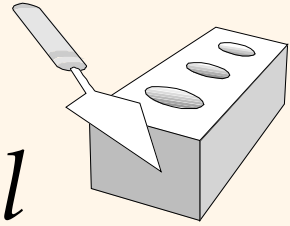
sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

chave primária

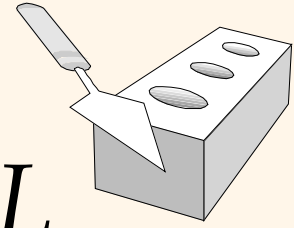
Students

sid	name	login	age	gpa
53666	Jones	jones@ cs	18	3.4
53688	Smith	smith@ eecs	18	3.2
53650	Smith	smith@ math	19	3.8

Garantindo a Integridade Referencial



- Considere Students e Enrolled; sid em Enrolled é uma chave estrangeira que faz referência a Students.
- O que deveria ser feito se uma tupla de Enrolled com um id de estudante não existente fosse inserida? (Rejeitá-la!)
- O que deveria ser feito se uma tupla de Student fosse removida?
  - Remover também todas as tuplas de Enrolled que a referenciam.
  - Rejeitar a remoção de uma tupla Students que é referenciada por outra relação.
  - Atribuir um sid padrão ao sid das tuplas de Enrolled que a referenciam.
  - Atribuir o valor null, denotando 'desconhecido' ou 'não aplicável', ao sid das tuplas de Enrolled que se referem a ela.
- Similar se a chave primária de uma tupla de Students é atualizada.

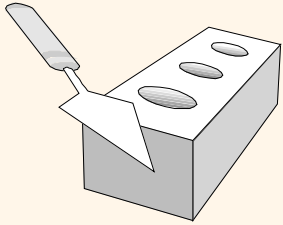


Integridade Referencial em SQL

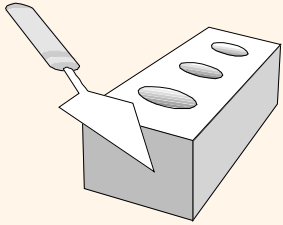
- SQL/92 e SQL:1999 contém as 4 opções para remover e atualizar relações mantendo a integridade referencial:
  - O padrão é NO ACTION (delete/update é rejeitado)
  - CASCADE (também remove todas as tuplas que fazem referência à tupla removida)
  - SET NULL / SET DEFAULT (atribui valor à chave estrangeira da tupla referenciada)

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
          ON DELETE CASCADE
          ON UPDATE SET DEFAULT )
```

De onde vêm as RIs?



- RIs estão baseadas em semânticas do mundo real que são descritas nas relações da base de dados.
- Podemos verificar uma instância da base de dados para ver se uma RI é violada, mas NUNCA podemos inferir que uma RI é verdadeira ao olharmos uma instância.
  - Uma RI é uma declaração a respeito de todas as instâncias possíveis.
- RIs de chave, de chave estrangeira e de domínio são mais comuns e as mais suportadas também.

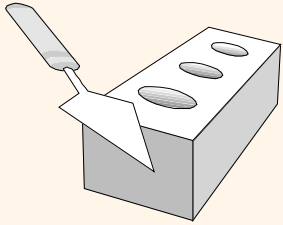


Visões

- Uma visão é apenas uma relação, mas armazenamos uma definição, ao invés de um conjunto de tuplas.

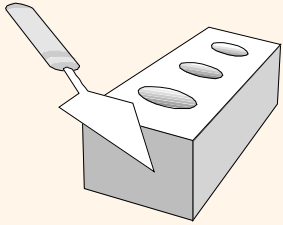
```
CREATE VIEW YoungActiveStudents (name, grade)
        AS SELECT S.name, E.grade
        FROM Students S, Enrolled E
        WHERE S.sid = E.sid and S.age<21
```

- Visões podem ser excluídas usando o comando DROP VIEW.
  - Como tratar DROP TABLE se há uma visão da tabela?
    - O comando DROP TABLE tem opções que permitem que o usuário especifique isto.



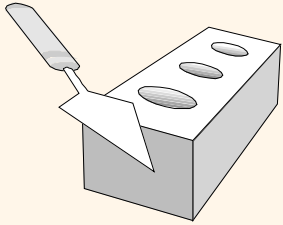
Visões e Segurança

- Visões podem ser utilizadas para apresentar informações (ou resumos) necessárias, ao mesmo tempo em que escondem detalhes a respeito da(s) relação(ões).
  - Dado YoungStudents, mas não Students ou Enrolled, podemos encontrar estudantes que estão matriculados, mas não os cids dos cursos em que estão matriculados.



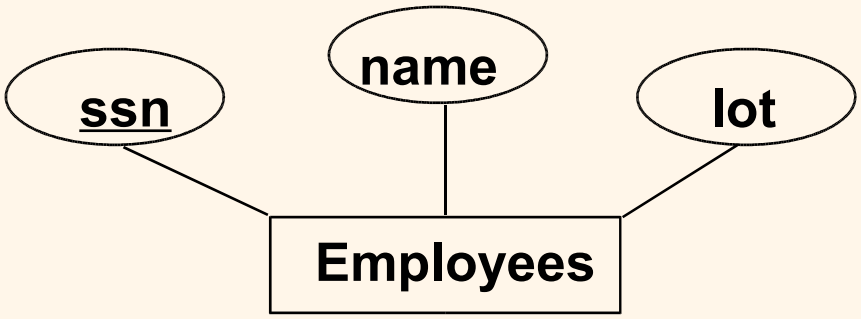
Projeto Lógico do BD: ER para Relacional

- ER: conveniente para representar um esquema inicial e de alto nível do banco de dados
- Existe um método para *transformar* um esquema ER em modelo relacional?



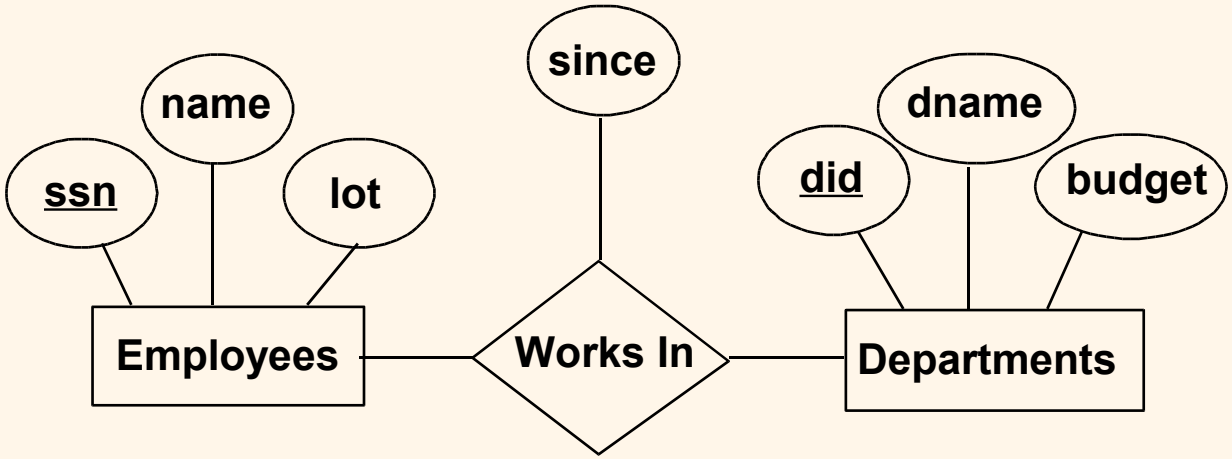
Transformando Conjuntos-entidade em tabelas

- Como um conjunto-entidade é representado em uma relação?
  - Num mapeamento direto, onde os atributos da entidade tornam-se atributos da tabela e a chave torna-se chave primária

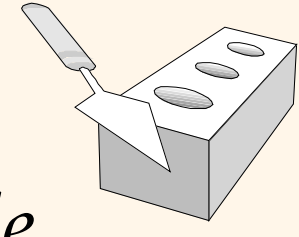



```
CREATE TABLE Employees
      (ssn CHAR(11),
       name CHAR(20),
       lot  INTEGER,
       PRIMARY KEY  (ssn))
```

Transformando Conjuntos-relacionamento sem restrição de chave em tabelas



- since – Data de admissão do funcionário
- did – Identificação do departamento (chave primária)



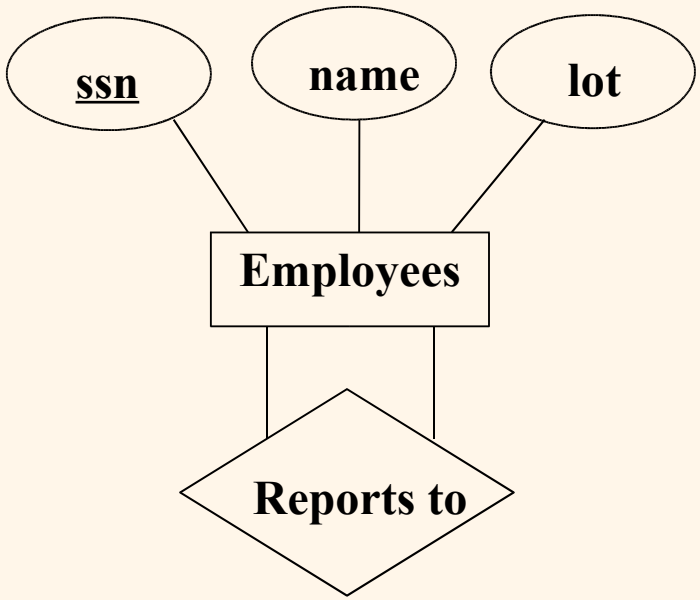
Transformando Conjuntos-relacionamento sem restrição de chave em tabelas

- Ao transformar um conjunto-relacionamento em uma tabela, devemos ter os seguintes atributos:
  - A chave primária de cada conjunto-entidade da relação (como chaves estrangeiras).
    - Este conjunto de atributos forma uma super-chave para a relação.
  - Todos atributos descritivos.

```
CREATE TABLE Works_In(
  ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
     REFERENCES Employees,
  FOREIGN KEY (did)
     REFERENCES Departments)
```

Transformando Conjuntos-relacionamento sem restrição de chave em tabelas

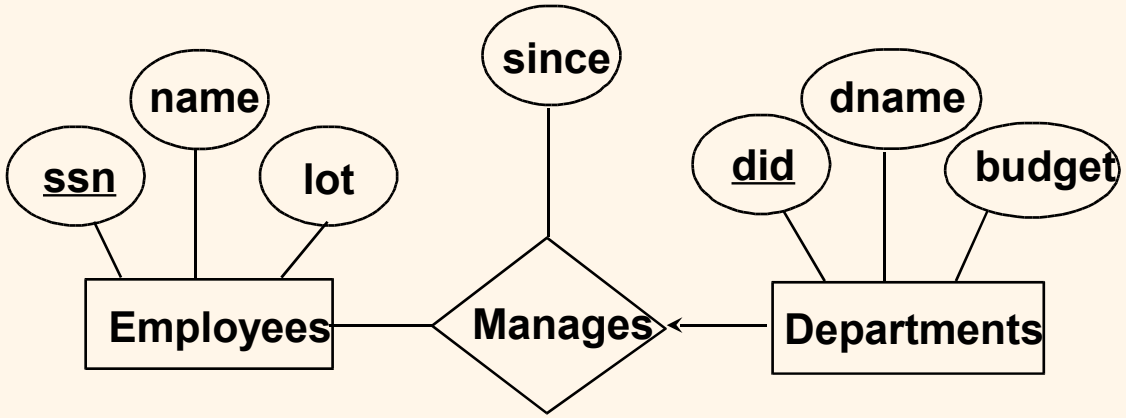
- Auto-Relacionamento

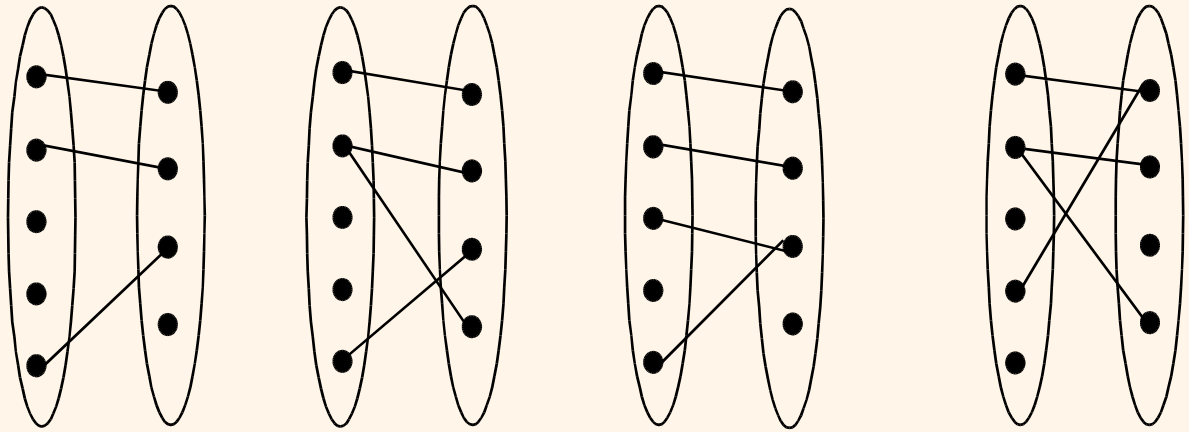



```
CREATE TABLE Reports_To(
  supervisor_ssn  CHAR(11),
  subordinate_ssn  CHAR(11),
PRIMARY KEY (supervisor_ssn,
                    subordinate_ssn),
FOREIGN KEY (supervisor_ssn)
          REFERENCES Employees
(ssn),
FOREIGN KEY (subordinate_ssn)
          REFERENCES Employees
(ssn) )
```

Revisão: Restrições de Chave

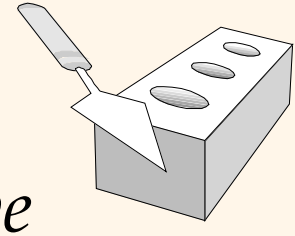
- Cada departamento tem no máximo **um** gerente, de acordo com as restrições de chave em *Manages*.





1-para-1 1-para-muitos muitos-para-1 muitos-para-muitos

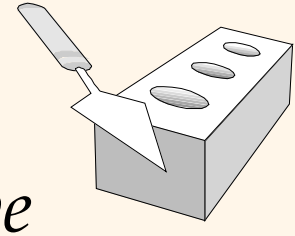
Como passamos isto para o modelo relacional?



Transformando
Conjuntos-relacionamento com restrição de chave em tabelas

- Relacionamento transforma-se em uma tabela
  - Cada departamento pode ser gerenciado por apenas uma pessoa, logo <u>did</u> será a chave primária

```
CREATE TABLE  Manages(
   ssn  CHAR(11),
   did  INTEGER,
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY (ssn)
            REFERENCES Employees,
   FOREIGN KEY (did)
            REFERENCES
                Departments)
```



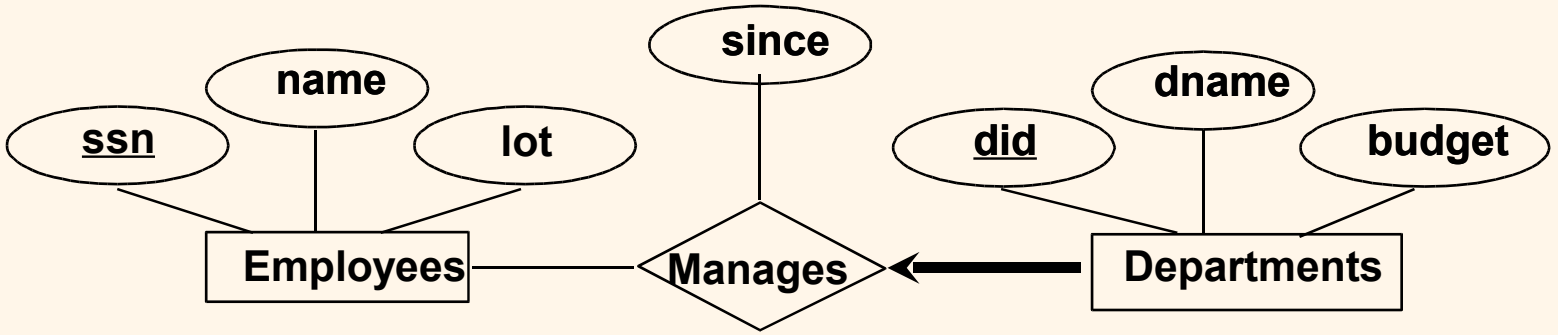
Transformando
Conjuntos-relacionamento com restrição de chave em tabelas

- As informações sobre o relacionamento são colocadas na tabela referente à entidade que carrega a chave da relação.

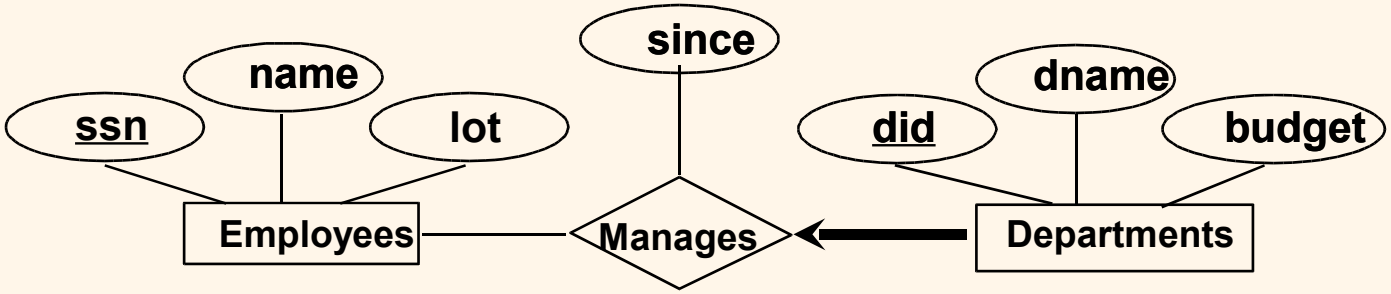
```
CREATE TABLE  Dept_Mgr(
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   ssn  CHAR(11),
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY (ssn)
      REFERENCES Employees)
```

Revisão: Restrições de Participação

- Todo departamento tem um gerente?
  - Se sim, esta é uma restrição de participação: a participação de Departments em Manages é denominada total (vs. parcial).
    - Cada valor de did na tabela Departments deve aparecer em uma linha da tabela Manages (com um valor não-null para ssn)

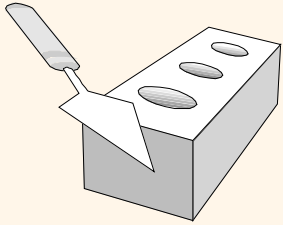


Restrições de Participação em SQL



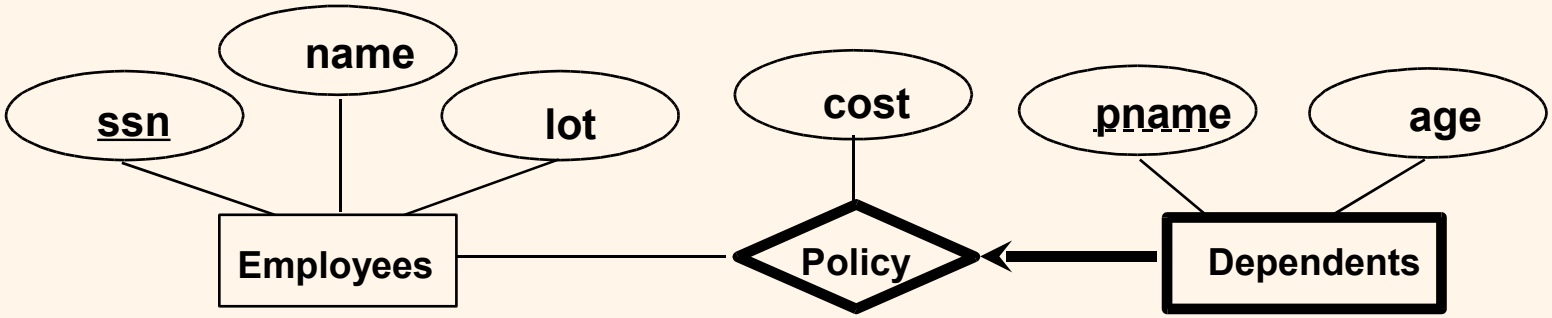

```
CREATE TABLE  Dept_Mgr(
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   ssn  CHAR(11) NOT NULL,
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY  (ssn)
            REFERENCES Employees,
            ON DELETE NO ACTION)
```

Revisão: Entidades Fracas



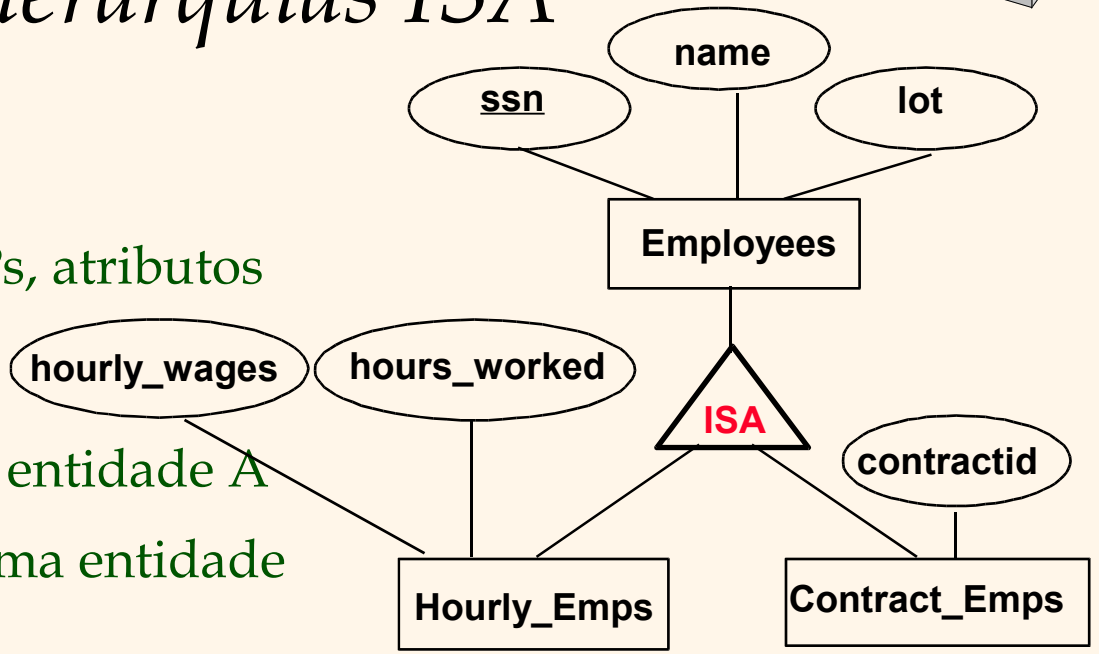
- Uma entidade fraca pode ser identificada univocamente considerando apenas a chave primária de outra entidade (que a possui).
  - O conjunto-entidade "dominante" e o conjunto-entidade fraca devem participar em um conjunto-relacionamento um-para-muitos (1 dominante, muitas entidades fracas).
  - O conjunto-entidade fraca deve ter participação total neste conjunto-relacionamento.
- Conjunto-entidade fraca e conjunto-relacionamento identificador são transformados em uma única tabela.
  - Quando a entidade "dominante" é removida, todas as entidades fracas também devem ser removidas.

Traduzindo Conjuntos-Entidade Fracas

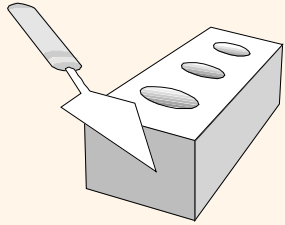



```
CREATE TABLE  Dep_Policy (
   pname  CHAR(20),
   age  INTEGER,
   cost  REAL,
   ssn  CHAR(11) NOT NULL,
   PRIMARY KEY  (pname, ssn),
   FOREIGN KEY  (ssn) REFERENCES Employees,
     ON DELETE CASCADE)
```

Revisão: Hierarquias ISA



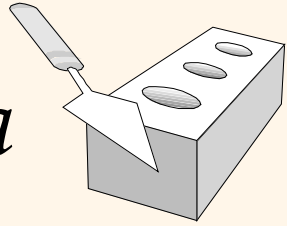
- Como em C++, ou outras LPs, atributos são herdados.
- Se declaramos A **ISA** B, cada entidade A também é considerada como uma entidade B.
- Sobreposição de restrições: Joe pode ser uma entidade Hourly_Emps e uma entidade Contract_Emps **ao mesmo tempo**?
- Restrições de cobertura: Cada entidade Employees tem que ser **necessariamente** uma entidade Hourly_Emps ou uma entidade Contract_Emps?



Traduzindo Hierarquias ISA para Relações

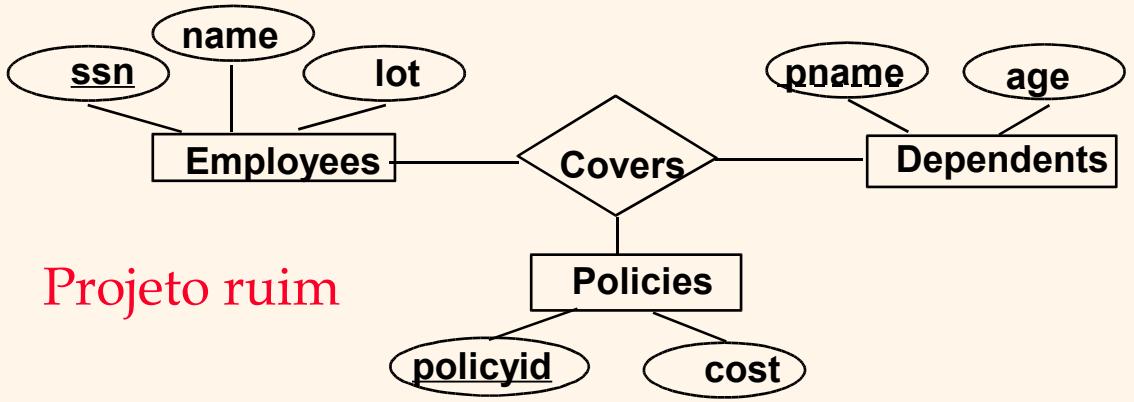
- Abordagem geral:
  - 3 relações: Employees, Hourly_Emps e Contract_Emps.
    - Hourly_Emps: Cada empregado é armazenado em Employees. Para empregados que trabalham por hora (hourly emps), informações adicionais são armazenadas em Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn*); deve-se remover tupla em Hourly_Emps se a tupla do empregado referenciado em Employees for removido.
    - Contract_Emps: da mesma forma que a anterior, teremos os atributos (*contractid*, *ssn*); também deve-se remover tupla em Contract_Emps se a tupla do empregado referenciado em Employees for removido.

Traduzindo Hierarquias ISA para Relações



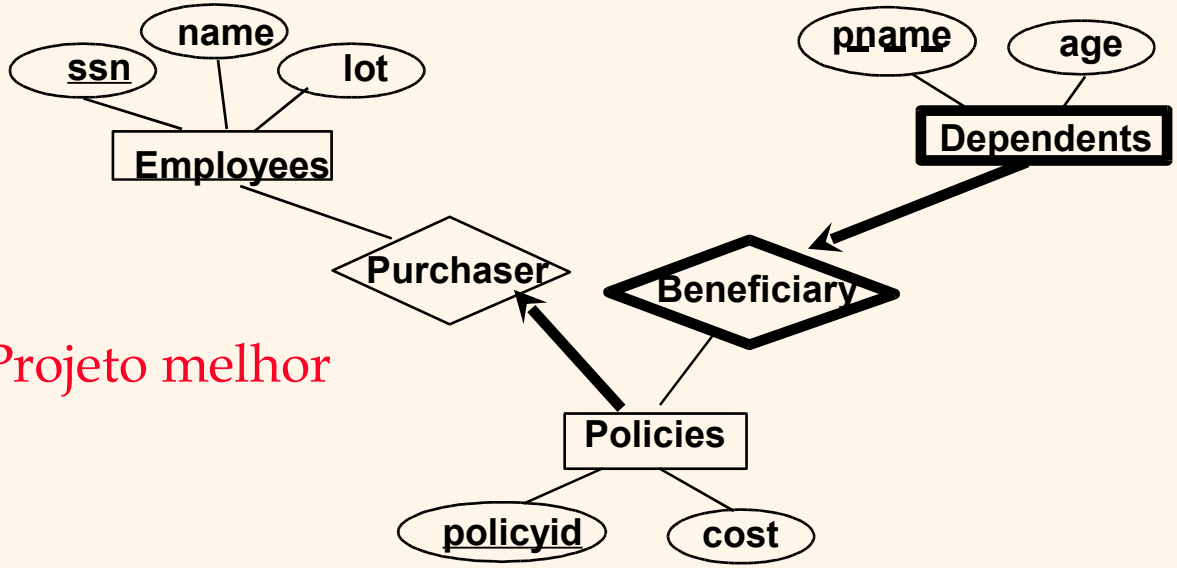
- Alternativas:
  - Apenas Hourly_Emps e Contract_Emps.
    - Hourly_Emps: *ssn, name, lot, hourly_wages, hours_worked*.
    - Contract_Emps: *ssn, name, lot, contractid*.
    - Cada empregado deve estar somente em uma destas duas subclasses.
  - Utilizando somente uma relação:
    - Employees: *ssn, name, lot, hourly_wages, hours_worked, contractid*.
    - Os atributos *hourly_wages, hours_worked, contractid* são declarados como null, pois *hourly_wages e hours_worked* não podem aparecer ao mesmo tempo que *contractid*, os dois primeiros vindo da relação Hourly_Emps e o seguinte de Contract_Emps.

Revisão: Relacionamentos Binário vs. Ternário



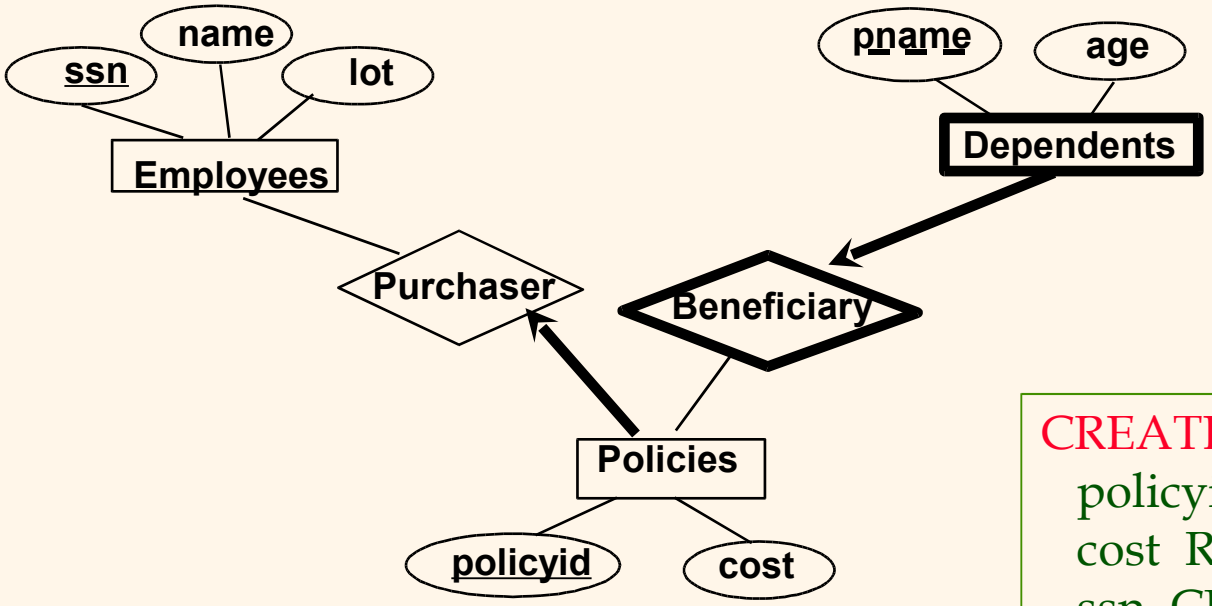
Projeto ruim

- Quais são as restrições adicionais no 2º diagrama?



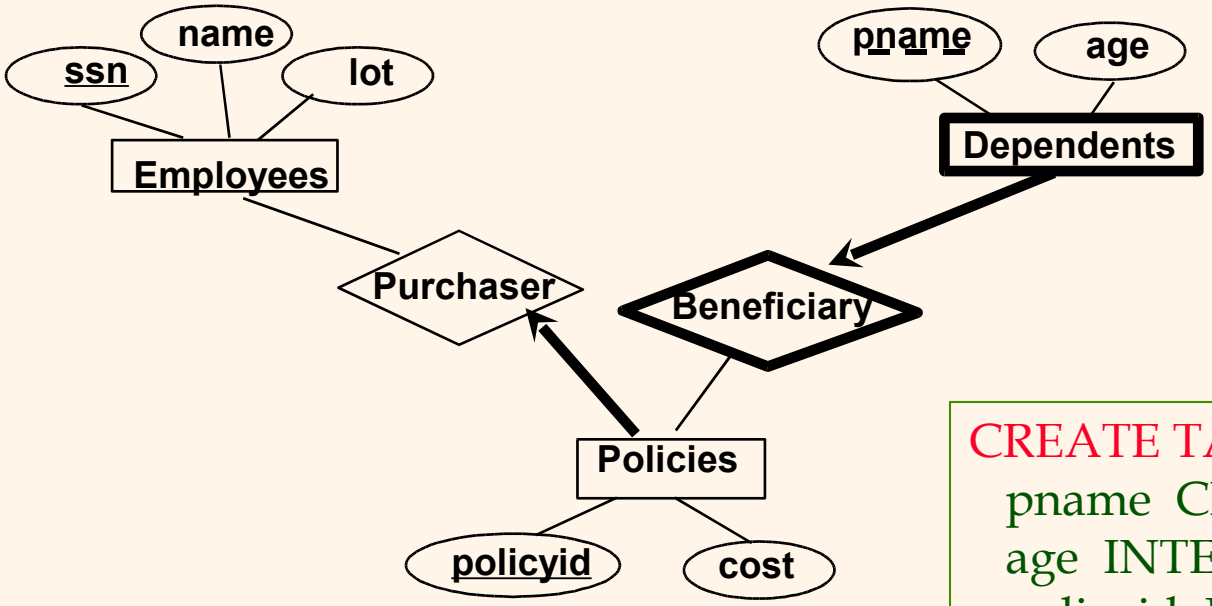
Projeto melhor

Relacionamentos Binário vs. Ternário



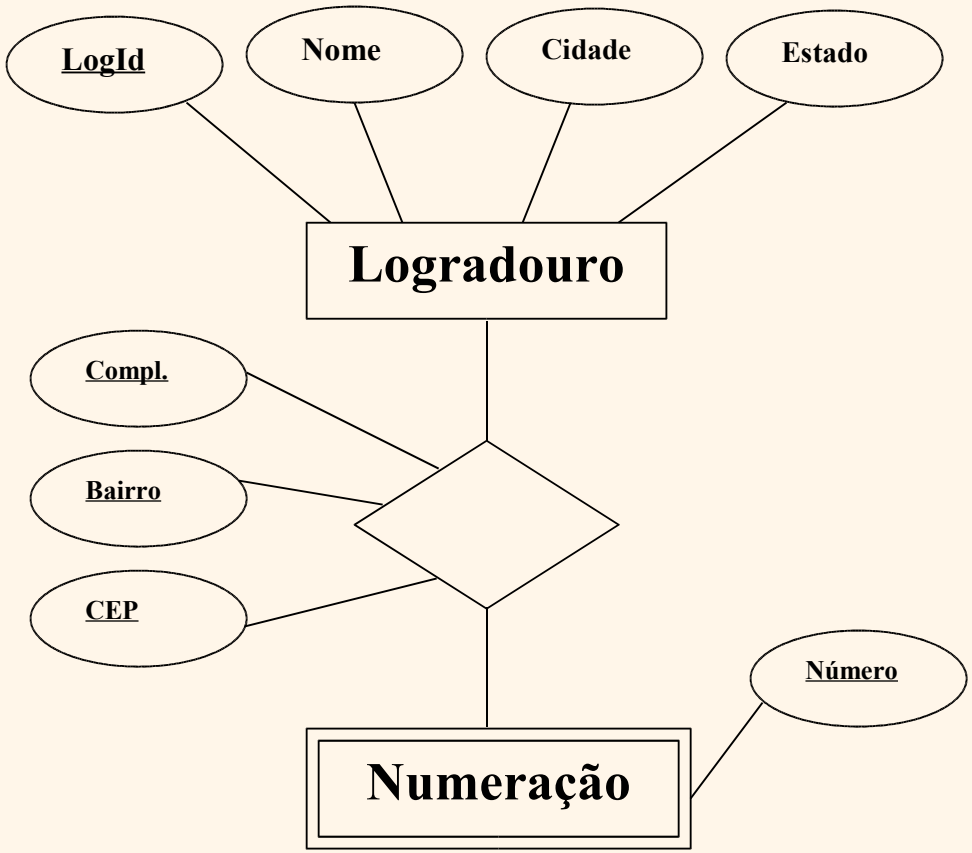

```
CREATE TABLE  Policies (
  policyid  INTEGER,
  cost  REAL,
  ssn  CHAR(11)  NOT NULL,
  PRIMARY KEY (policyid).
  FOREIGN KEY (ssn)
    REFERENCES Employees,
  ON DELETE CASCADE)
```

Relacionamentos Binário vs. Ternário

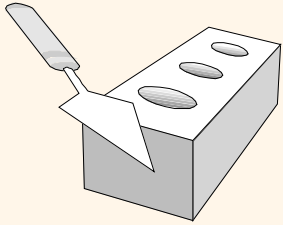



```
CREATE TABLE Dependents (
  pname  CHAR(20),
  age  INTEGER,
  policyid  INTEGER,
  PRIMARY KEY (pname, policyid),
  FOREIGN KEY (policyid)
    REFERENCES Policies,
  ON DELETE CASCADE)
```

Exemplo adicional - Táxi

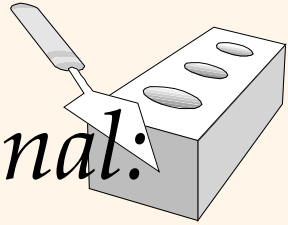



```
CREATE TABLE Endereço (
  numero  INTEGER,
  complemento  CHAR(15)
  bairro  CHAR(30),
  cep  CHAR(8),
  logid  INTEGER
PRIMARY KEY (numero, logid)
FOREIGN KEY (logid)
  REFERENCES Logradouro
ON DELETE CASCADE )
```



Modelo Relacional: Resumo

- Uma representação tabular de dados.
- Simples e intuitivo, sendo o mais utilizado atualmente.
- Restrições de integridade podem ser especificadas pelo DBA, baseadas na semântica da aplicação. O SGBD deve verificar as violações, tratando-as.
  - Duas RIs importantes: chaves primária e chave estrangeira.
  - Ainda, sempre temos restrições de domínio.
- Existem linguagens de consultas "poderosas" e naturais.
- Regras para traduzir ER para o modelo relacional.



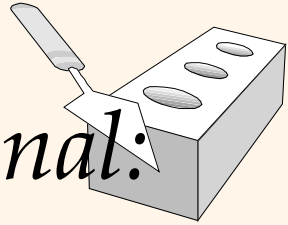
Tradução de ER para Modelo Relacional: Resumo

- Entidade

M. Rel.	Atrib.	Chave
relação	mesmos	mesma

- Relacionamentos 1:1

Mod. Relacional
Embutir em qualquer das relações; se uma delas é fraca, embutir nela; usar chave estrangeira que referencia a outra entidade



Tradução de ER para Modelo Relacional: Resumo

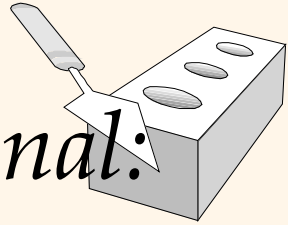
❖ Relacionamentos N:1*

MER	Mod. Relacional	Atributos	Chave
poucos atributos	embutir numa das relações (em geral a do lado N) como um atributo que será chave estrangeira da outra entidade do lado 1	Também vai para a relação	Mesma
vários atributos	Criar relação auxiliar	Mesmos mais as chaves estrangeiras das entidades envolvidas	Composta pela chaves estrangeiras referenciando entidades envolvidas

* a escolha da opção a seguir deve levar em conta também a restrição de participação, quando houver

❖ Relacionamentos N:N

Mod. Relacional	Atributos	Chave
Criar relação auxiliar	Mesmos mais as chaves estrangeiras referenciado as entidades envolvidas	Composta pela chaves estrangeiras referenciando entidades envolvidas



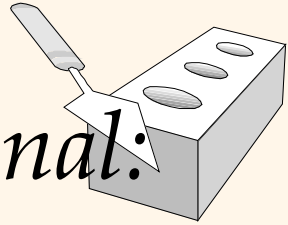
Tradução de ER para Modelo Relacional: Resumo

❖ Auto-relacionamentos

MER	Mod. Relacional	Atributos	Chave
N:1	usar chave estrangeira para a própria relação		
N:N	Criar relação auxiliar	Mesmos mais as chaves estrangeiras das entidades envolvidas	Composta pela chaves estrangeiras referenciando entidades envolvidas

❖ Relacionamentos ternários

MER	Mod. Rel.	Atributos	Chave
1:N:N	relação	Mesmos mais as chaves estrangeiras das entidades envolvidas (lado N) e chave da entidade lado 1	Composta pela chaves estrangeiras referenciando entidades envolvidas (lado N)
N:N:N	relação	Mesmos mais as chaves estrangeiras das entidades envolvidas (lado N)	Composta pela chaves estrangeiras referenciando entidades envolvidas (lado N)



Tradução de ER para Modelo Relacional: Resumo

- Relacionamentos com agregações

Mod. Relacional	Atributos	Chave
Relação auxiliar	Mesmos mais as chaves estrangeiras das entidades envolvidas	Composta pela chaves estrangeiras referenciando entidades envolvidas; note que uma destas chaves é composta

- Especializações

M. Rel.	Atrib.	Chave
Uma relação para cada especialização com chave estrangeira que referencia a super entidade	mesmos	É a chave estrangeira