Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação

Fundamentos de Banco de Dados

Angelo Brayner
brayner@dc.ufc.br

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Histórico -

- Até final da década de 1960
  - Sistema de arquivos
- Final da década de 1960
  - Nascimento da tecnologia de banco de dados
    - Gerenciamento eficiente de dados
  - Representação de dados através de árvores
    - Modelo hierárquico
  - Representação de dados através de grafos
    - Modelo Redes

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Histórico -

- Década 1970
  - Modelo Relacional
    - Representar dados em um nível conceitual mais elevado
      - Relações matemáticas
      - Tabelas
  - Impulsionou pesquisas na área de banco de dados
    - Processamento de consultas
    - Controle de concorrência
      - monitoramento do acesso concorrente aos dados

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Histórico -

- Década de 1980
  - Modelo orientado a objeto
    - Modelo de dados extensível
      - Tipos de dados definidos pelo usuário
    - Objeto
      - Dados
      - Comportamento (métodos)
    - Sistema de banco de dados OO gerencia
      - Dados
      - Código

UFC CC DC

1. Tecnologia de Banco de Dados
- Histórico -

- Década de 1980
  - Distribuição
    - Sistemas de Bancos de Dados Distribuídos
    - Sistemas Bancos de Dados Cliente-Servidor
    - Sistemas de Bancos de Dados Paralelos
  - Integração
    - Integrar Bancos de Dados
      - Distribuídos
      - Heterogêneos
      - Autônomos

UFC CC DC

1. Tecnologia de Banco de Dados
- Histórico -

- Década de 1990
  - Banco de Dados Objeto-Relacional
- (2000, ...)
  - Integração
  - Mobilidade
  - Fluxo de Dados
  - Sistemas de banco de dados cientes do hardware
    - Máquinas multi-cores
    - Memória de estado sólido

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Motivação -

- Sistemas de arquivos (décadas 1960 e 1970)
  - Cada aplicação
    - Definir e manter seus próprios dados

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Motivação -

- Sistemas de arquivos (anos 60 e 70)
  - Aplicativo C para alterar nome de um estudante, dada a matrícula e o novo nome
    - Programa Altera_Struct
  - Correspondente na linguagem SQL

```
update Estudantes
set nome="Novo Nome"
where matricula=matr
```

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Motivação -

- Propriedades
  - Garantir independência de dados
    - Alteração na organização lógica ou física dos dados não implica na alteração de aplicações
  - Eliminar redundância de dados
    - Eliminar inconsistência de dados
  - Facilitar acesso a dados através de uma linguagem de consulta
  - Evitar inconsistências produzidas pelo acesso concorrente
  - Recuperar banco de dados após falhas

UFC
CC
DC

1. Tecnologia de Bancos de Dados
- Arquitetura de Três Camadas -

UFC
CC
DC

1. Tecnologia de Bancos de Dados
- Arquitetura de Três Camadas -

- Esquema Interno - Camada Interna
  - Descreve como os dados estão fisicamente armazenados
    - Exemplo
      - Organização de arquivo
        - seqüencial-indexado, hash, seqüencial, heap
      - Alocação em disco
      - Estruturas de índices
- Esquema Conceitual - Camada Conceitual
  - Descreve quais dados estão armazenados no banco de dados
- Esquema Externo - Camada Externa
  - Descreve parte do banco de dados
    - Simplificar a visão do usuário
      - "Ver" só o que interessa
    - Segurança

UFC
CC
DC

1. Tecnologia de Bancos de Dados
- Sistemas de Banco de Dados -

- Sistema de Bancos de Dados (SBD)
  - Banco de Dados (BD)
    - Conjunto de dados relacionados
  - Sistema Gerenciador de Bancos de Dados (SGBD)
    - Componente de software
      - Acesso
      - Controle de Concorrência
      - Recuperação
      - Armazenamento

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Arquitetura SBD -

UFC
CC
DC

1. Tecnologia de Banco de Dados
- Componentes da Arquitetura de SBDs -

- SGBD
  - Compilador DML
    - Analisa sintaticamente e semanticamente comandos DML expressos em uma linguagem de consulta (SQL)
    - Traduz estes comandos para uma forma de representação interna (p. ex. álgebra relacional, cálculo relacional)
  - Pré-Compilador DML
    - Traduz comandos DML em chamadas a procedimentos na linguagem hospedeira
  - Interpretador DDL
    - Interpreta comandos DDL e os armazena no catálogo
      - Tabelas contendo meta-dados (Esquema)
  - Máquina de Consultas
    - Otimização de consultas e geração de planos de execução

UFC CC DC

1. Tecnologia de Banco de Dados
- Componentes da Arquitetura de SBDs -

- SGBD
  - Gerenciador de Transações
    - Controle de concorrência
    - Recuperação do banco de dados após falhas
  - Gerenciador de Buffer
    - Responsável por manter na área de buffer (memória principal) páginas mais acessadas
    - Política de alocação de páginas em buffer
      - LRU, MRU, FIFO, ARC
  - Gerenciador de Armazenamento
    - Responsável pelo armazenamento físico em memória secundária

UFC CC DC

1. Tecnologia de Banco de Dados
- Componentes da Arquitetura de SBDs -

- BD
  - Arquivos de dados
    - Armazena os dados
  - Arquivos de Índices
    - Estruturas de índices para os arquivos de dados
      - Índices ordenados
        - Árvores B+, Arquivos grade, árvores kd, Árvores R, Árvores quadrantes
      - Índices não ordenados
        - Indice hash

UFC CC DC

1. Tecnologia de Banco de Dados
- Componentes da Arquitetura de SBDs -

- BD
  - Catálogo
    - Armazena esquema do banco de dados (meta-dados)
      - Nomes das tabelas, atributos de cada tabela, definição de índice para uma tabela, etc...
    - Armazena informações estatísticas
      - Exemplo
        - Cardinalidade de uma tabela
      - Utilizadas na otimização de consultas
  - Fragmentos de código
    - Stored procedures, gatilhos (triggers), funções

UFC CC DC

1. Tecnologia de Banco de Dados
- Modelo De Dados -

- Definido por três componentes
  - Uma coleção de tipos de estrutura de dados
    - blocos de construção do banco de dados
  - Uma coleção de operadores
    - Podem ser aplicados a qualquer instância dos tipos de dados definidos para o modelo
  - Uma coleção de regras de integridade
    - Definem o conjunto de estados consistentes do banco de dados
- Funcionalidade
  - Representar dados do mundo real
    - Capturar semântica e incorporá-la em um banco de dados
      - Através do modelo relacional representar os dados de uma universidade

1. Tecnologia de Banco de Dados
– Classificação de Sistemas de Bancos de Dados –

- Modelo de Dados
  - Sistema de Banco de Dados Relacional
    - Modelo relacional
  - Sistema de Banco de Dados Orientado a Objeto
    - Modelo orientado a objeto
  - Sistema de Banco de Dados Objeto-Relacional
    - Modelo relacional + Modelo OO

1. Tecnologia de Banco de Dados
– Classificação de Sistemas de Bancos de Dados –

- Arquitetura
  - Centralizada
    - Sistema de Banco de Dados Centralizados
      - Os componentes do SBD residem no mesmo host
  - Distribuída
    - Critérios
      - Função, Controle, Dados
    - Sistema de Banco de Dados Cliente-Servidor
      - Distribuição de funções do SGBD entre clientes e servidor
    - Sistema de Banco de Dados Paralelos
      - Distribuição do controle de funções do DBMS entre diversos sistemas computacionais
    - Sistema de Banco de Dados Distribuídos
      - Distribuição de dados através de diversos SBDs homogêneos

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Modelo de dados MER
  - Proposto por P. Chen em 1976
  - Utilizado como modelo conceitual em projeto de BDs
    - Ferramenta para a modelagem de BDs
  - Não é implementado por nenhum SBD
- Princípio básico
  - Representar dados através
    - Entidades
    - Relacionamentos
      - Entre entidades
    - Atributos
      - Propriedades de entidades ou relacionamentos

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Entidade
  - Representação de um objeto do mundo real
    - Exemplos de entidades do mundo real
      - Objeto concreto
        - Um empregado, um carro, um estudante
      - Objeto abstrato
        - Uma empresa, uma conta bancária, uma disciplina

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Atributos de uma entidade
  - Propriedades que caracterizam uma entidade
  - Exemplos
    - Atributos de empregados
      - matrícula, nome, endereço, rg, cpf, data-nasc, salário, lotação, data-admissão
    - Atributos de estudantes
      - matrícula, nome, curso, rg, cpf, data-ingresso
  - A cada atributo de uma entidade deve estar associado um valor

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Conjunto de entidades
  - Entidades que apresentam mesmo conjunto de atributos
    - Estudantes, Empregados, Contas
- Definição formal de Atributo
  - $\forall E_i \in$ Entidades, Atributo: $E_i \longmapsto v_k$, onde $v_k \in$ dom(Atributo)
  - Atributo nome do conjunto de entidades Empregado
    - nome(Empregado_1)=José

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Atributos chave de uma entidade
  - Atributos que identificam univocamente uma entidade
  - Seja f um atributo chave para o conjunto de entidades E
    - Se $x \in E$ e $f(x)=v$, então
      - $\forall e \in E, e \neq x, f(e) \neq v$
  - Exemplo
    - Matrícula é um atributo chave para Estudante
- Tipos de atributos
  - Atributo atômico
    - Atributo indivisível
  - Atributo composto
    - Formado por vários atributos atômicos

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Tipos de atributos
  - Atributo mono-valorado
    - Atributo para o qual está associado um único valor
  - Atributo multi-valorado
    - Atributo para qual podem estar associados vários valores
    - Exemplo
      - Para o atributo telefone podem estar associados vários valores, como telefone residencial, comercial e celular
  - Atributo derivado
    - Atributo cujo valor poder derivado com base no valor de um outro atributo (atributo base)
    - Exemplo
      - Idade pode ser derivado do atributo data de nascimento

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Relacionamento
  - Abstração que representa associação entre entidades de diferentes conjuntos de entidades
  - Exemplo
    - Relacionamento que associa o empregado Bárbara com o departamento "Ciência da Computação"
- Conjunto de relacionamentos (tipo de relacionamento)
  - Grupo de relacionamentos que representam o mesmo tipo de associação
    - Conjunto de relacionamentos lotação
      - Todos relacionamentos entre empregado e departamento

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Conjunto de relacionamentos
  - Seja R um conjunto de relacionamentos, representando associações entre os conjuntos de entidades E_1, E_2, ... E_n, então
  - $R \subseteq E_1 \times E_2 \times \ldots \times E_n$
    - Seja $r \in R$, então $r=(e_1, e_2, \ldots, e_n)$, onde $e_1 \in E_1$, $e_2 \in E_2$, ..., $e_n \in E_n$
  - Exemplo
    - lotação ⊆ Departamento x Empregado
      - (Bárbara, Ciência da Computação) ∈ lotação
- Papel de uma entidade no relacionamento
  - Função de uma entidade no relacionamento
    - Papel de empregado
      - É lotado (Bárbara <u>é lotada em</u> Ciência da Computação)
    - Papel de departamento
      - Lota (Ciência da Computação <u>lota</u> Bárbara)

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Grau de Relacionamento
  - Número de entidades participantes no relacionamento
  - Exemplos
    - Relacionamento binário
      - Associa dois conjuntos de entidades
    - Relacionamento ternário
      - Associa três conjuntos de entidades
        - Exemplo
          - Relacionamento que associa Agência-Conta-Cliente

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Auto-relacionamento (relacionamento recursivo)
  - Relacionamento envolvendo um único conjunto de entidades
  - O conjunto de entidades apresenta diferentes papéis
  - Considere o seguinte cenário:
    - Supervisores são responsáveis por subconjuntos de empregados de um mesmo departamento
      - Relacionamento supervisiona
        - Empregado desempenha dois papéis
          - é-supervisionado-por
          - é-supervisor-de

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Atributos de relacionamento
  - Propriedades que descrevem um relacionamento
    - Considere o atributo data-lotação
      - Representa a data em que um empregado foi lotado em um determinado departamento
      - data-lotação é atributo do relacionamento Lotação
    - Considere o atributo nota
      - Descreve a nota de um aluno em uma disciplina
      - nota é um atributo do relacionamento cursa
        - Relacionamento entre os conjuntos de entidades Estudante e Disciplina

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Restrições estruturais de relacionamentos
  - Cardinalidade de relacionamento
  - Restrição de participação
- Cardinalidade de relacionamento
  - Indica o número de entidades que podem participar de um relacionamento
  - Seja R um relacionamento binário entre os conjuntos de entidades A e B
    - Cardinalidade Um para um (1:1)
      - Uma entidade de A está associada a uma só entidade de B
      - Uma entidade de B está associada a uma só entidade de A

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Cardinalidade de relacionamento
  - Cardinalidade Um para muitos (1:N)
    - Uma entidade de A (B) está associada a qualquer quantidade de entidades de B (A)
    - Uma entidade de B (A) está associada a uma só entidade de A (B)
  - Cardinalidade Muitos para muitos (N:N)
    - Uma entidade de A pode estar associada a qualquer quantidade de entidades de B
    - Uma entidade de B só pode estar associada a qualquer quantidade de entidades de A

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Exemplos Cardinalidade de relacionamentos
  - Cardinalidade do relacionamento lotação entre Departamento e Empregado 1:N
  - Cardinalidade do relacionamento cursa entre Estudante e Disciplina N:N
  - Cardinalidade do auto-relacionamento supervisiona 1:N
  - Cardinalidade do relacionamento gerência entre Departamento e Empregado 1:1

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Restrição de participação
  - Especifica a obrigatoriedade ou não de uma entidade e participar de um relacionamento com outra entidade
  - Participação total
    - A participação de um conjunto de entidades A é total em R, se toda entidade de A participa de pelo menos um relacionamento em R
    - Dependência existencial
    - Considere o relacionamento lotação, para o qual todo empregado deve estar lotado em algum departamento
      - A participação de Empregado em lotação é total
  - Participação parcial
    - A participação de um conjunto de entidades A é parcial em R, se apenas um subconjunto de entidades de A participa em R
      - Relacionamento cursa entre Estudante e Disciplina

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Modelagem (parcial) de dados sobre cursos, alunos e disciplinas (Universidade)

UFC CC DC

2. Modelo Entidade-Relacionamento
- Conceitos Básicos -

- Entidade fraca
  - Entidade cuja existência depende de estar associada, via um relacionamento (relacionamento de identificação), com uma outra entidade (entidade forte)
  - Considere o relacionamento dependência entre os conjuntos de entidades Empregado e Dependente (dependentes dos empregados)
    - A existência do dependente Bárbara
      - Condiciona-se existência do empregado Angelo e que Bárbara esteja relacionada a Angelo através do relacionamento dependência
  - Uma entidade fraca é identificada
    - Por estar relacionada com uma entidade forte
      - Pelo atributo chave da entidade forte
    - Atributos da própria entidade fraca
      - Chave parcial

UFC CC DC

2. Modelo Entidade-Relacionamento
- Diagrama ER -

- Ferramenta (gráfica) de projeto
  - Capaz de capturar e representar graficamente toda estrutura lógica de um banco de dados
  - Utilizada para modelagem de BDs
  - Existem ferramentas
    - Fornecem interface gráfica para criar DERs
      - brmodelo, ERwin, visual paradigm
    - A partir do DER especificado, geram o esquema do BD relacional
      - ERwin, visual paradigm

2. Modelo Entidade-Relacionamento
- Diagrama ER -

- Notação

Conjunto de entidades

Conjunto de entidades fracas

Conjunto de relacionamentos

Relacionamento de identificação

Atributo

2. Modelo Entidade-Relacionamento
- Diagrama ER -

- Notação

Atributo chave

Atributo derivado

Atributo multivalorado

Atributo composto

2. Modelo Entidade-Relacionamento
- Diagrama ER -

- Notação

Atributo chave parcial de uma entidade fraca

Cardinalidade 1:N

Participação total de E2 em R

Restrição estrutural de participação e cardinalidade de E_2 em R

2. Modelo Entidade-Relacionamento
- Diagrama ER -

- Exercício
  - Utilize o DER para modelar o BD para o seguinte cenário na empresa X
    - Todo empregado deve estar lotado em um único departamento e é coordenado por um supervisor (empregado)
    - Um departamento possui no mínimo 5 empregados, onde um deles é o gerente do departamento.
    - Os dependentes dos funcionários devem possuir como atributos: nome, data-nasc. A idade limite para ser dependente de um empregado é 18 anos
    - O salário de um empregado é calculado com base nos seus diversos vencimentos e descontos.
    - Para cada tipo de vencimento ou de desconto, existe uma descrição e o valor correspondente

2. Modelo Entidade-Relacionamento
- Diagrama ER -

2. Modelo Entidade-Relacionamento
- Diagrama ER -

UFC CC DC

2. Modelo Entidade-Relacionamento - Propriedades Avançadas -

- Especialização
  - Existem conjuntos de entidades compostos de subgrupos de entidades, onde
    - Cada subgrupo apresenta propriedades específicas
  - Conjunto de entidades Empregado numa universidade
    - Pode-se identificar os seguintes subconjuntos
      - Professores
        - Titulação
        - Universidade de titulação
        - Regime de trabalho (DE, 40h, 20h)
      - Técnicos
        - Grau de instrução
        - Área de atuação (contador, secretária, etc...)
      - Engenheiros
        - Especialidade

UFC CC DC

2. Modelo Entidade-Relacionamento - Propriedades Avançadas -

- Especialização
  - Processo de identificação de subgrupos de entidades dentro de um conjunto de entidades
    - Processo de especialização pode ser recursivo
  - Um único conjunto de entidades pode ser especializado por mais de uma característica de diferenciação (especialização)
    - No exemplo de Empregado em uma universidade
      - Especialização por tipo de empregado
      - Especialização por tipo de contrato
        - RJU
        - Serviços prestados

UFC CC DC

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Especialização
  - Subgrupos identificados em um processo de especialização podem participar de relacionamentos que não se aplicam a todas entidades do conjunto de entidades de origem
  - Herança de propriedades
    - Subgrupos de entidades herdam todas as propriedades do conjunto de entidade de mais alto nível

UFC CC DC

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Especialização
  - Notação MER

UFC CC DC

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Generalização
  - Processo de identificação de conjuntos de entidades que possuem características em comum
    - Mesmos atributos e participam de mesmos relacionamentos
  - Formação de um único conjunto de entidades de mais alto nível
  - É realizada sobre vários conjuntos de entidades
    - Identificar um conjunto de entidades de mais alto nível

UFC CC DC

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Generalização (cont.)
  - Ênfase nas similaridades entre diversos conjuntos de entidades
  - Redução de redundância de representação
    - Atributos compartilhados só serão representados no conjunto de entidades de nível mais alto
      - Não serão repetidos
  - Na prática
    - Generalização é o processo inverso da especialização

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Restrições para generalização e especialização
  - Sejam $E_1, E_2, \ldots, E_n$ subconjuntos de entidades do conjunto de entidades E

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Restrições para generalização e especialização
  - Disjunção
    - $\forall 0<i,j\leq n,\ i\neq j:\ E_i \cap E_j = \emptyset$
    - Nenhuma entidade de E pode pertencer a mais de um subgrupo
  - Completeza
    - Total
      - $E = \bigcup_{i=1}^{n} E_i$
    - Parcial
      - $E \supseteq \bigcup_{i=1}^{n} E_i$

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Agregação
  - Abstração que representa relacionamentos como entidades
    - Utilizado para representar relacionamentos de relacionamentos
  - Exemplo
    - Modelagem de dados em um banco BX, onde clientes estão relacionados a agência e conta
    - Estratégia 1

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Agregação
  - Problemas na Estratégia 1
    - Não representa a estrutura lógica que deve ser modelada
      - Relacionamento de cliente com o relacionamento agência-conta
    - Na prática, para acessar todas as contas de uma agência
      - Terão que ser acessadas todas triplas (agência,conta,cliente)
      - Uma conta pode pertencer a vários clientes
        - Vários acessos redundantes

UFC CC DC

2. Modelo Entidade-Relacionamento
- Propriedades Avançadas -

- Agregação
  - Estratégia 2 (agregação)

UFC CC DC

2. Modelo Entidade-Relacionamento
- Diagrama ER -

- Utilize o DER para modelar o BD com os dados de estoque da empresa X
  - Cada item de estoque possui um código, descrição, quantidade em estoque, preço de compra, preço de venda e o fornecedor.
  - Cada item de estoque só poderá ser fornecido por um único fornecedor. Cada fornecedor terá um código, nome, endereço, telefones, cidade e estado.
  - Para cada item comprado pela empresa, deve ser identificado o código do item comprado, o código do fornecedor vendeu (forneceu) o item, a quantidade adquirida, o preço de compra, a data e hora em que a compra foi efetuada.
  - Existem várias lojas (filiais) vendendo produtos da empresa. Cada loja tem um código identificador, nome, endereço, cidade de localização. Cada filial é administrada por um gerente, que obrigatoriamente deve ser um vendedor da loja
  - Cada vendedor terá uma matrícula, nome, RG, CPF, salário e filial de lotação. Os atributos RG e CPF não podem ter valores repetidos em mais de uma tupla. O valor do salário deve ser sempre maior que R$ 1.200,00.
  - Para cada venda, deve ser identificado o código do item vendido, a matrícula do vendedor que efetuou a venda, a quantidade vendida, o preço de venda, a data e hora em que a venda foi efetuada. O preço de venda não poder ser inferior ao preço de comprar acrescido de 40% (do valor do próprio preço de compra).

UFC
CC
DC

3. Modelo Relacional
- Introdução -

- Proposto em 1970 por Codd
  - IBM
- Consolidou-se como principal modelo de dados para aplicações comerciais
- Modelo de dados para Bancos de Dados Relacionais
- SBDs relacionais
  - DB2 (IBM)
  - Postgres
  - MySQL
  - Oracle
  - SQL Server

UFC
CC
DC

3. Modelo Relacional
- Introdução -

- Um banco de dados relacional consiste
  - Um conjunto de tabelas (relações)
- Tabelas
  - Conjunto de linhas (Tuplas)
  - Cada linha
    - Conjunto de colunas (atributos)

UFC CC DC

3. Modelo Relacional
- Conceitos Básicos -

- Domínio
  - Conjunto de valores permitidos para um atributo
    - Valores são atômicos
      - Indivisíveis
    - Exemplo: Domínio do atributo matrícula
      - Conjunto de todos os valores válidos de matrícula
  - dom(A) denota o domínio do atributo A

UFC CC DC

3. Modelo Relacional
- Conceitos Básicos -

- Esquema de uma relação R
  - Descreve os dados uma relação (meta-dados)
  - Representado por $R(A_1, A_2, \ldots, A_n)$, onde
    - $A_1, A_2, \ldots, A_n$ uma lista de atributos de R
- Instância r(R) de uma Relação R (ou relação R)
  - Dado o esquema $R(A_1, A_2, \ldots, A_n)$
  - $r(R) \subseteq dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$
    - subconjunto do produto cartesiano dos domínios dos atributos que definem R
    - cada tupla de r(R) relaciona valores dos vários domínios
    - r(R) representa um conjunto de tuplas de R

UFC
CC
DC

3. Modelo Relacional
- Conceitos Básicos -

- Informalmente, uma relação (tabela) R, com esquema $R(A_1, A_2, \ldots, A_n)$, consiste da
  - Instância r(R), onde
    - $r(R)=\{t_1, t_2, \ldots, t_k\}$
      - Cada t_i representa uma n-tupla de n valores $<v_1, v_2, \ldots, v_n>$, onde cada $v_j \in dom(A_j)$, $0<j\leq n$
- Observação importante
  - r(R) contém os dados
  - o esquema de R descreve os dados armazenados na relação

UFC
CC
DC

3. Modelo Relacional
- Conceitos Básicos -

- Esquema de um banco de dados relacional
  - Conjunto de esquemas de relação mais um conjunto de restrições de integridade IC
    - $S=\{R_1, R_2, \ldots, R_n\}$ e um conjunto de restrições de integridade IC
- Instância de um banco de dados relacional
  - Seja o esquema S
  - Uma instância DB para o esquema S
    - $DB=\{r_1, r_2, \ldots, r_n\}$, onde
    - Cada r_i é uma instância de relação de R_i e
    - Cada r_i satisfaz as restrições de integridade especificadas em IC

UFC
CC
DC

3. Modelo Relacional
- Restrições de Integridade -

- Restrição de domínio
  - O valor de cada atributo A tem que ser um valor atômico de dom(A)
- Restrição de Chave
  - Uma relação R é definida como um conjunto de tuplas
    - Elementos de conjuntos são distintos entre si
    - Tuplas de R devem ser distintas entre si
      - Duas tuplas de R não podem ter a mesma combinação de valores para seus atributos
  - Geralmente existe um subconjunto SC de atributos em um esquema de relação R
    - Todas as tuplas de qualquer instância r(R) apresentam uma combinação diferente de valores para os atributos de SC
      - $\forall\ t_i, t_j \in r\ 0<i,j\leq n,\ i\neq j : t_i[SC] \neq t_j[SC]$
    - Super chave (superkey)

UFC
CC
DC

3. Modelo Relacional
- Restrições de Integridade -

- Restrição de Chave
  - Super chave pode apresentar atributos que podem se comportar como chave individualmente
    - Empregado(matr,nome,ender,cpf)
      - matr e cpf são atributos da super chave
      - Apenas um dos dois já garante a não existência de tuplas repetidas
  - Chave candidata (candidate key)
    - Atributo da super chave que pode funcionar como chave da relação
    - Para Empregado existem duas chaves candidatas
      - matr ou cpf
  - Chave primária (primary key)
    - Chave candidata escolhida como chave da relação
      - Garante a unicidade de uma tupla na relação

UFC
CC
DC

3. Modelo Relacional
- Restrições de Integridade -

- Restrição de Integridade de Entidade
  - Especifica que nenhuma chave primária pode ter valor nulo
- Restrição de Integridade Referencial
  - Sejam dois esquemas de relação R e S
  - Um conjunto de atributos FK de um esquema de relação R é chave estrangeira (foreign key) se
    - Os atributos em FK têm o mesmo domínio que a chave primária PK da relação S, e
    - Um valor de FK em uma tupla t_i de r(R)
      - Ou ocorre em S como valor da chave primária PK para uma tupla t_k em s(S)
        - $t_i[FK] = t_k[PK]$
      - ou é nulo

UFC
CC
DC

3. Modelo Relacional
- Restrições de Integridade -

- Restrição de Integridade Referencial
  - Exemplo
    - Departamento(cod_depart, nome, ender)
    - Empregado(matr, nome, ender, cpf, lotação)
    - Se lotação for chave estrangeira (referenciando cod_depart), então
      - Deve ser garantido que o valor de lotação ou referencie um valor existente como chave primária em Departamento ou seja nulo
  - Garantida automaticamente pelos SGBDs existente no mercado

UFC
CC
DC

3. Modelo Relacional
- Álgebra Relacional -

- Coleção de operadores sobre relações
  - Relação $\xrightarrow{\text{expressão AR}}$ Relação
- Linguagem de consulta procedimental para BDs relacionais
  - Ferramenta <u>Relax</u>
- Operadores básicos
  - Seleção (σ) } Operadores unários
  - Projeção (Π) } Operadores unários
  - União (∪) } Operadores binários
  - Diferença (−) } Operadores binários
  - Produto cartesiano (x) } Operadores binários

UFC
CC
DC

3. Modelo Relacional
- Álgebra Relacional -

- Operação de seleção
  - Seleciona um subconjunto de tuplas de uma relação, para as quais um predicado P é Verdade
    - Filtro de tuplas
  - Notação
    - σ_P (r)
      - r é uma relação
      - P representa um predicado (condição de seleção)
    - P é construído através de átomos
      - $t[A_i]\ \theta\ t[A_k]$, $t \in r$ e A_i e A_k são atributos de r
      - $t[A_i]\ \theta\ C$, onde C é uma constante
      - θ denota um operador de comparação
        - $=, \neq, >, \geq, <, \leq$
    - Átomos podem ser conectados por ∧ (and), ∨ (or), ¬ (not)

3. Modelo Relacional
- Álgebra Relacional -

- Operação de seleção
  - Exemplo
    - Considere a relação Empregado
      - Empregado(matr, nome, ender, cpf, salário, lotação)
    - Listar todos os empregados que ganham salário maior que 5000
      - $\sigma_{salário>5000}$ (Empregado)
    - Listar todos os empregados não lotados no departamento com código igual a 002 e que ganham salários entre 5000 e 10000

3. Modelo Relacional
- Álgebra Relacional -

- Ferramenta Relax
  - Banco de dados Kemper Datenbanksysteme (en)
    - Construa o DER
    - Consulta C1
      - Construa e execute a expressão da AR, que retorna as disciplinas de quatro créditos, lecionadas pelo professor de ID igual a 2125

3. Modelo Relacional
- Álgebra Relacional -

- Operação de seleção (cont.)
  - Propriedades da seleção
    - $\sigma_{P1 \wedge P2}(r) \Leftrightarrow \sigma_{P1}(\sigma_{P2}(r))$
      - Distributividade
    - $\sigma_{P1}(\sigma_{P2}(r)) \Leftrightarrow \sigma_{P2}(\sigma_{P1}(r))$
      - Comutatividade
- Exercício
  - Mostre as propriedades acima, utilizando a consulta C1

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Projeção
  - Seleciona um subconjunto de atributos de uma relação
    - Filtro de colunas
  - Notação
    - $\Pi_{Ai1, Ai2,..., Ain}$ (r)
      - r é uma relação com esquema $R(A_1, A_2, ..., A_n)$
      - $\{A_{i1}, A_{i2}, ..., A_{in}\} \subseteq \{A1, A2, ..., An\}$
  - Exemplo
    - Listar o nome e salário de todos os funcionários
      - $\Pi_{nome, salário}$ (Empregado)
    - Listar nome e salário de todos os empregado que ganham salário maior que 9000

3. Modelo Relacional
- Álgebra Relacional -

- Exercício
  - Construa e execute a expressão da AR, que retorna o nome das disciplinas (lecture.title) de quatro créditos, lecionadas pelo professor de ID igual a 2125

3. Modelo Relacional
- Álgebra Relacional -

- Operação de União
  - Executa a união de duas relações compatíveis
    - Duas relações $R(A_1, A_2, \ldots, A_n)$ e $S(B_1, B_2, \ldots, B_n)$ são compatíveis, se
      - Apresentam o mesmo número de atributos
      - $dom(A_i) = dom(B_i)$, $\forall i$, $0 < i \leq n$
  - Notação
    - $r \cup s$
  - Exemplo
    - Considere as seguintes relações
      - Empregado(matr, nome, ender, dt-nasc, cpf, salário, lotação)
      - Dependente(nome-dep, data-nasc, matr-resp)
    - Listar nome e data de nascimento de todos os funcionários e dependentes na empresa
      - $\Pi_{nome,\ dt\text{-}nasc}$ (Empregado) $\cup$ $\Pi_{nome\text{-}dep,\ data\text{-}nasc}$(Dependente)

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Diferença
  - O resultado da operação r – s
    - relação que contém as tuplas de r que não pertencem a s
    - r e s são relações compatíveis
  - Exemplo
    - Listar matrícula de estudantes não matriculados em disciplinas (Relax)
      - $\Pi_{student_id}$ (student) – $\Pi_{student_id}$ (attends)

3. Modelo Relacional
- Álgebra Relacional -

- Exercícios
  - Listar ID dos professores que não estão lecionando disciplinas
  - Listar o nome de todos os professores e de todos os alunos

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Produto Cartesiano
  - Sejam r e s com esquemas $R(A_1, A_2, \ldots, A_n)$ e $S(B_1, B_2, \ldots, B_m)$, respectivamente
  - Resultado da operação r x s é uma relação T
    - Esquema de T
      - $T(r.A_1, r.A_2, \ldots, r.A_n, s.B_1, s.B_2, \ldots, s.B_m)$
        - (n+m) atributos
    - $t \in T \Leftrightarrow \exists v \in r$ e $\exists u \in s$, tal que $t[A_i]=v[A_i]$, $0<i\leq n$, e $t[B_j]=u[B_j]$, $0<j\leq m$
    - Sejam n_r e n_s as cardinalidades de r e s, respectivamente
      - A cardinalidade de T é $n_r * n_s$

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Produto Cartesiano
  - r x s

r

r.A	r.B
a1	b1
a1	b2
a2	b1

s

s.A	s.B	s.C
a3	b6	c5
a2	b3	c3
a2	b1	c4

r x s

r.A	r.B	s.A	s.B	s.C
a1	b1	a3	b6	c5
a1	b1	a2	b3	c3
a1	b1	a2	b1	c4
a1	b2	a3	b6	c5
a1	b2	a2	b3	c3
a1	b2	a2	b1	c4
a2	b1	a3	b6	c5
a2	b1	a2	b3	c3
a2	b1	a2	b1	c4

3. Modelo Relacional
- Álgebra Relacional -

- Exercício
  - Execute o produto cartesiano entres as relações Professor e Lecture
    - Explique o resultado apresentado

3. Modelo Relacional
- Álgebra Relacional -

- Exercícios (Ferramenta Relax)
  - Escreva as seguintes consultas sobre Banco de dados Kemper Datenbanksysteme (en), utilizando a álgebra relacional
    - Listar disciplinas lecionadas pelo professor com id igual a 2125
    - Mostrar o nome do professor, com sua sala
    - Listar matrícula dos alunos com nota menor que 2
    - Listar nome de disciplinas com créditos menores que 4
    - Apresentar id dos alunos que não estão cursando disciplinas
    - Desafio
      - Mostrar nome do professor e nome das disciplinas ministradas por ele

3. Modelo Relacional
- Álgebra Relacional -

- Operadores derivados
  - Definidos a partir dos operadores básicos
    - Junção ($\bowtie$)
    - Semi-junção ($\ltimes$)
    - Interseção ($\cap$)
    - Divisão ($\div$)

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Junção (theta-join)
  - $r \bowtie_P s = \sigma_P (r \times s)$
    - P é a condição de junção, definido
      - P contém apenas operações de comparação entre atributos de r e s ou
        - Entre atributos de r e s com constantes

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Junção
  - $r \bowtie_{r.B \neq s.B} s$

r

r.A	r.B
a1	b1
a1	b2
a2	b1

s

s.A	s.B	s.C
a3	b6	c5
a2	b3	c3
a2	b1	c4

$r \bowtie_{r.B \neq s.B} s$

r.A	r.B	s.A	s.B	s.C
a1	b1	a3	b6	c5
a1	b1	a2	b3	c3
a1	b2	a3	b6	c5
a1	b2	a2	b3	c3
a1	b2	a2	b1	c4
a2	b1	a3	b6	c5
a2	b1	a2	b3	c3

3. Modelo Relacional
- Álgebra Relacional -

- Exercícios (Ferramenta Relax)
  - Escreva as seguintes consultas sobre Banco de dados Kemper Datenbanksysteme (en), utilizando a álgebra relacional
    - Apresentar nome dos professores que não estão ministrando disciplinas

renomeando a tabela professor por x

$\pi_{name}((\pi_{x.employee_id}(\rho_{x}(professor)) - \pi_{lectured_by}(lecture)) \bowtie_{x.employee_id = professor.employee_id} professor)$

    - Mostrar nome do professor e nome da disciplina ministrada por ele

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Semi-Junção (Semi-join)
  - $r \ltimes_P s = \Pi_R (r \bowtie_P s)$
  - Utilizada para otimizar o processamento de consultas em SBDs Distribuídos
- Operação de Interseção
  - $r \cap s = r - (r - s)$
  - r e s devem ser relações compatíveis

3. Modelo Relacional
- Álgebra Relacional -

- Desafio
  - Retornar o nome dos alunos matriculados em disciplinas, com o nome destas disciplinas e o nome do professor destas disciplinas
    - No resultado deve aparecer nome do aluno, nome da disciplina e nome do professor da disciplina
      - 10 min

$\pi_{\text{student.name, title, professor.name}}$ (((student $\bowtie_{\text{attends.student_id=student.student_id}}$ attends) $\bowtie_{\text{attends.lecture_id=lecture.lecture_id}}$ lecture) $\bowtie_{\text{lectured_by=employee_id}}$ professor)

3. Modelo Relacional
- Álgebra Relacional -

- Operação de divisão
  - Sejam r e s com esquemas
    - $R(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ e $S(B_1, B_2, \ldots, B_m)$
  - Resultado da operação $r \div s$
    - É uma relação $T(A_1, A_2, \ldots, A_n)$, onde
      - Para uma tupla v pertencer a T, todos os valores de v precisam aparecer em R em associação com todas as tuplas de S

3. Modelo Relacional
- Álgebra Relacional -

- Operação de divisão
  - Exemplo

r

r.A	r.B	r.C
a1	b6	c3
a2	b3	c3
a2	b3	c1
a3	b3	c2
a2	b3	c2

s

s.C
c3
c1
c2

r÷s

r.A	r.B
a2	b3

3. Modelo Relacional
- Álgebra Relacional -

- Operação de divisão
  - Sejam r e s com esquemas
    - $R(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ e $S(B_1, B_2, \ldots, B_m)$
      - Conjunto de atributos de S é um subconjunto dos atributos de R (R-Div)
  - $r \div s = \Pi_{R-(R \cap S)}(r) - \Pi_{R-(R \cap S)}((\Pi_{R-(R \cap S)}(r) \times s) - \Pi_{R-(R \cap S),S}(r))$

retorna tuplas de r com falsas associações com s

3. Modelo Relacional
- Álgebra Relacional -

- Desafio
  - Considere as tabelas
    - Emp-Desc(matr,cod-desc) e
    - Desconto(cod-desc,valor,descrição)
  - Mostrar a construção do resultado da consulta que retorna os empregados que têm descontados todos os descontos existentes na empresa

Emp_desc ← R

cod-desc	matr
3	11
99	5
7	11
7	21
99	11

Desconto ← Garantir R-Div $S = \Pi_{cod\text{-}desc}$ (Desconto)

cod-desc	valor	descrição
3	30	IR
99	8.5	INSS
7	10	Seguro

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Atribuição
  - Às vezes, é importante escrever uma expressão da álgebra relacional em diferentes partes
    - Atribuir resultados das partes a relações temporárias
  - Notação
    - := ou ←
  - Exemplo
    - rel1 ← $\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$
    - rel2 ← $\Pi_{R-S}(r)$
    - resultado:= rel1 - rel2

3. Modelo Relacional
- Álgebra Relacional -

- Funções Agregadas
  - Funções aplicadas sobre uma coleção de valores do banco de dados
  - sum
    - Retorna o somatório dos valores de uma coleção
  - avg
    - Retorna a média dos valores de uma coleção
  - max
    - Retorna o maior valor de uma coleção de valores
  - min
    - Retorna o menor valor de uma coleção

3. Modelo Relacional
- Álgebra Relacional -

- Funções Agregadas (cont)
  - count
    - Retorna o número de elementos de uma coleção
  - count-distinct
    - Algumas vezes, torna-se necessário eliminar repetições para o cálculo das funções agregadas

3. Modelo Relacional
- Álgebra Relacional -

- Funções Agregadas (cont.)
  - Exemplos
    - Considere a relação
      - Empregado(matr, nome, ender, salário, cpf, lotação)
        - matr é a chave primária
    - Encontre o número de empregados lotados no departamento 001
      - $count(\Pi_{matr}(\sigma_{lotação=001}(Empregado)))$
    - Encontre o maior salário da empresa
      - $max(\Pi_{salário}(Empregado))$

3. Modelo Relacional
- Álgebra Relacional -

- Funções Agregadas (cont.)
  - Exemplos
    - Encontre o salário médio da empresa
      - $avg(\Pi_{salário}(Empregado))$
    - Encontre a quantidade de salários distintos no departamento 001
      - $count\text{-}distinct(\Pi_{salário}(\sigma_{lotação=001}(Empregado)))$
- Desafio
  - Encontre o primeiro e segundo maiores salários da empresa

3. Modelo Relacional
- Álgebra Relacional -

- Desafio
  - Execute a seguinte consulta no Relax (Datenbansystem)
    - Retornar o nome de todos alunos com as disciplinas que estão cursando
      - Resultado no formato:

Nome	Nome Disc
André	Lógica
André	FBD
André	Cálculo III
Bárbara	SGBD
João	Sistemas distribuídos

3. Modelo Relacional
- Álgebra Relacional -

- Considere as seguintes relações
  - Vendedor(matr, nome, ender, salário, cpf)
  - Vendas(matr_vend,cod_item,qtde, pr_venda)
- Listar histórico de vendas de cada vendedor. O resultado deve apresentar o seguinte esquema:
  - Res(nome, cod-item, qtde, pr-venda)

$\Pi_{Res}(Vendedor \bowtie_{matr=matr_vend} Vendas)$

Consulta com perda de informação
- Não aparecerão, no resultado, Vendedores, que não efetuaram vendas

3. Modelo Relacional
- Álgebra Relacional -

- Outer join
  - Adiciona tuplas que não satisfazem a condição de junção ao resultado
  - Tipos
    - Junção externa à esquerda (left outer join)
    - Junção externa à direita (right outer join)
    - Junção externa completa (full outer join)

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Left Outer join
  - $r ⟕ s$
  - Calcula o resultado da junção de r com s
    - Adiciona ao resultado da junção
      - Tuplas da relação à esquerda (r) que não satisfazem à condição de junção
      - Atribui valores nulos aos atributos não definidos para estas tuplas

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Right Outer join
  - $r ⟖ s$
  - Calcula o resultado da junção de r com s
    - Adiciona ao resultado da junção
      - Tuplas da relação à direita (s) que não satisfazem à condição de junção
      - Atribui valores nulos aos atributos não definidos para estas tuplas

3. Modelo Relacional
- Álgebra Relacional -

- Operação de Full Outer join
  - r ⟗ s
  - Calcula o resultado da junção de r com s
    - Adiciona ao resultado da junção
      - Tuplas das relações operando que não satisfazem à condição de junção
      - Atribui valores nulos aos atributos não definidos para estas tuplas

3. Modelo Relacional
- Álgebra Relacional -

- Operação de outer-join
  - left outer join
    - Considere as seguintes relações
      - Vendedor(matr, nome, salário)
      - Vendas(matr,cod-item,qtde, pr-venda)
    - Listar o histórico de vendas de cada vendedor

Matr	nome	salário
9	Bárbara	4.000,00
3	Andre	3.500,00
11	Yami	1.500,00
5	Flor	1.000,00
7	Caio	2.500,00

Matr	cod-item	qtde	pr-venda
11	553	1	25,00
3	72	3	100,00
7	553	9	25,00

3. Modelo Relacional
- Álgebra Relacional -

- Operação de outer-join
  - left outer join (cont.)
    - Listar o histórico de vendas de cada vendedor
      - Vendedor ⟕ Vendas

Matr	nome	salário	matr	cod-item	qtde	pr-venda
9	Bárbara	4.000,00	NULL	NULL	NULL	NULL
3	Andre	3.500,00	3	72	3	100.00
11	Yami	1.500,00	11	553	1	25,00
5	Flor	1.000,00	NULL	NULL	NULL	NULL
7	Caio	2.500,00	7	553	9	25,00

3. Modelo Relacional
- Álgebra Relacional -

- Execute as seguintes consultas no Relax (Datenbansystem)
  - Retornar o nome de <u>todos</u> alunos com as disciplinas que estão cursando (slide 95)
  - Retornar o nome de <u>todos</u> professores com as disciplinas que estão lecionando
  - Retornar o código de todas disciplinas e o código de seus pré-requisitos (tabela requires)
- Desafio
  - Retornar o nome de todas disciplinas e o nome de seus pré-requisitos

3. Modelo Relacional
- Cálculo Relacional -

- Desenvolvido por Codd
- Linguagem para especificar propriedades que o resultado da consulta deve atender
  - Não há a especificação de um algoritmos para a obtenção do resultado da consulta
  - Linguagem não procedimental
- Consulta em Cálculo Relacional
  - $\{t \mid P(t)\}$
    - O conjunto de todas tuplas t, tal que o predicado (fórmula) P é verdade para t

3. Modelo Relacional
- Cálculo Relacional -

- Fórmulas são construídas através de átomos
- Átomos
  - $t \in r$, onde t é variável do tipo tupla e r uma relação
    - não é permitida a utilização do operador $\notin$
  - $s[A_i]\ \theta\ t[B_k]$, onde s e t são variáveis do tipo tupla. A_i e B_k representam atributos de s e t, respectivamente
    - θ denota um operador de comparação ($=, \neq, >, \geq, <, \leq$)
      - Os atributos devem possuir domínios cujos elementos possam ser comparados por θ
  - $s[A_i]\ \theta\ K$, onde K é uma constante
  - Átomos podem ser conectados por $\wedge$ (and), $\vee$ (or) e $\neg$ (not)

3. Modelo Relacional
- Cálculo Relacional -

- Regras para construção de fórmulas
  - Todo átomo é uma fórmula
  - Se P_i e P_k são fórmulas, então $P_i \wedge P_k$, $P_i \vee P_k$, $\neg P_i$, $\neg P_k$ e $P_i \Rightarrow P_k$ também são fórmulas
  - Se P é uma fórmula e r uma relação, então
    - $(\exists t \in r)\ (P(t))$ é uma fórmula
      - Quantificador existencial
    - $(\forall t \in r)\ (P(t))$ é uma fórmula
      - Quantificador universal

3. Modelo Relacional
- Cálculo Relacional -

- Considere as seguintes relações
  - Departamento(cod_depart, nome, ender)
  - Empregado(matr, nome, ender, cpf, salario, lotacao)
- Consulta C1:
  - Nome e salário dos empregados com salário maior que 5000
  - C1: $\{e[nome], e[salário] \mid e \in Empregado \wedge e[salário] > 5000\}$
  - Consulta na Álgebra Relacional:
    - $\Pi_{nome,\ salário}\ (\sigma_{salário>5000}\ (Empregado))$

3. Modelo Relacional
- Cálculo Relacional -

- Consulta C2:
  - Nome e salário dos empregados que trabalham no departamento 'Informática'

    C2: {e[nome],e[salário] | e ∈ Empregado ∧ ((∃d)(d ∈ Departamento ∧ d[nome]='Informática' ∧ d[cod_dep]=e[lotacao]))}
- Consulta C3:
  - Nome dos empregados que não possuem dependentes

    C3: {e[nome] | e ∈ Empregado ∧ ((∀p) ((p ∈ Dependente) ∧ ¬(e.matr=p.matr))}

3. Modelo Relacional
- Cálculo Relacional -

- Expressões seguras
  - C_k: {e | ¬(e ∈ Empregado)}
    - Retorna um conjunto infinito de tuplas
  - C_k não é segura
- Domínio de uma expressão
  - Valores que aparecem como constantes ou que existem em alguma tupla referenciada na expressão
  - Dom(e ∈ Empregado ∧ e[salário]>5000)
    - Constante 5000 e os valores de tuplas pertencentes a tabela Empregado
- A <u>expressão é segura</u> se os valores que aparecem no seu resultado pertencem ao domínio da expressão

UFC
CC
DC

3. Modelo Relacional
- Cálculo Relacional -

- Exemplo
  - Dom(¬(e ∈ Empregado))
    - Valores de tuplas pertencentes a Empregado
  - {e | ¬(e ∈ Empregado)}
    - Tem como resultado tuplas que não pertencem a Empregado
  - (¬(e ∈ Empregado)) não é segura
- Cálculo Relacional e Álgebra Relacional são padrões para expressividade relacional
  - completeza relacional

UFC
CC
DC

4. Projeto de Bancos de Dados
- Fases -

UFC
CC
DC

4. Projeto de Bancos de Dados
- Fases -

- Análise (especificação) de requisitos
  - Projetista de banco de dados deve realizar entrevistas com usuários prospectivos do banco de dados
    - Requisitos de dados
- Projeto conceitual
  - Com base nos requisitos de dados, criar um esquema conceitual para o banco de dados
    - Modelo de dados conceitual (MER)
      - Construir DER
- Projeto lógico
  - Com base no DER definido na fase anterior
    - Criar um diagrama relacional
      - Representação gráfica de um esquema relacional

UFC
CC
DC

4. Projeto de Bancos de Dados
- Fases -

- Projeto Físico
  - Definir estruturas de armazenamento
    - Como e onde devem ser armazenadas as tabelas
      - Uma tabela em um arquivo ou várias tabelas em um único arquivo
      - Distribuir uma tabela em vários discos
  - Definir caminhos de acesso
    - Definir índices
      - Ordenado
        - Primário
        - Secundário
      - Hash
    - Visões materializadas

UFC CC DC

4. Projeto de Bancos de Dados
- Fases -

- Implementação
  - Com base no DR definido e as estruturas de armazenamento e caminhos de acesso definidos
    - Criar o banco de dados
      - Executar expressões DDL
    - Carregar os dados no BD

UFC CC DC

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Diagrama relacional (DR)
  - Ferramenta gráfica utilizada para representar um esquema de banco de dados relacional
- Passo 1
  - Para cada conjunto de entidades E, criar uma tabela com todos os atributos de E
    - Escolher uma chave candidata para ser a chave primária da tabela
    - Apenas os componentes atômicos de atributos compostos devem ser incluídos
  - Notação de tabela no DR

UFC CC DC

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Passo 2
  - Para cada relacionamento binário 1:1 entre os conjuntos de entidades E1 e E2
    - Escolher uma das tabelas, por exemplo E2, e incluir como chave estrangeira em E2 a chave primária PK da outra tabela (E1)
      - Critério de escolha
        - Entidade com participação total no relacionamento
    - Atributos de relacionamentos devem ser incluídos na tabela com chave estrangeira
  - Notação

UFC CC DC

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Passo 3
  - Para cada relacionamento binário 1:N entre os conjuntos de entidades E1 e E2
    - Identificar o conjunto de entidades que participa do lado N (suponha que seja E2)
    - Incluir como chave estrangeira na tabela E2 a chave primária da outra tabela (E1)
    - Atributos de relacionamentos devem ser incluídos na tabela com chave estrangeira
  - Notação

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Passo 4
  - Para cada relacionamento binário N:N entre os conjuntos de entidades E1 e E2
    - Criar uma nova tabela auxiliar tab-aux para representar o relacionamento
    - Incluir como chaves estrangeiras na tabela tab-aux as chaves primárias de E1 e E2
      - Estes dois atributos comporão a chave primária de tab-aux (chave primária composta)
    - Atributos de relacionamentos devem ser incluídos na tabela tab-aux
  - Notação

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Passo 5
  - Para relacionamento de grau maior que 2
    - Criar uma nova tabela auxiliar tab-aux para representar o relacionamento
    - Incluir como chaves estrangeiras na tabela tab-aux as chaves primárias das tabelas que participam do relacionamento
      - Estes atributos comporão a chave primária de tab-aux
- Passo 6
  - Para cada conjunto de entidades fracas F
    - Cria uma tabela TFr com todos os atributos de F
    - Incluir como chave estrangeira de TF a chave primária da tabela correspondentes ao conjunto de entidades fortes R
    - A chave primária de TFr será composta pela chave parcial de F e chave primária de R

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Passo 7
  - Para cada atributo multi-valorado A de um conjunto de entidades E1
    - Criar uma tabela T com o atributo A
    - Incluir como chave estrangeira em T a chave primária de E1
    - A chave primária de T será composta do atributo A mais a chave primária de E1

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Construa o DR para o DER do slide 42

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Exercício
  - Construa o DR para o seguinte DER

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- DR

UFC
CC
DC

4. Projeto de Bancos de Dados
- Construção do Diagrama Relacional -

- Construa o DR para o DER do slide 44

UFC
CC
DC

5. SQL
- Histórico -

- Structured Query Language - SQL
  - Desenvolvida pela IBM
    - Structured English Query Language - Sequel
    - Linguagem de consulta para o sistema R
- Dois conjuntos de operações
  - Data Description Language (DDL)
    - Operações para a definição do banco de dados
  - Data Manipulation Language (DML)
    - Operações para o acesso e manipulação dos dados

UFC
CC
DC

5. SQL
- Histórico -

- Padrão
  - ANSI
    - SQL 86
    - SQL 89
    - SQL 92
    - SQL 99
      - Propriedade de SBDs objeto-relacional

UFC
CC
DC

5. SQL
- Operações da DDL -

- Esquema de um banco de dados relacional
  - Conjunto de esquemas de relação mais um conjunto de restrições de integridade IC
- Expressões DDL do SQL permitem especificar
  - Esquema de relações (tabelas)
  - Domínio de valores associados a cada atributo
  - Restrições de integridade
  - Conjunto de índices a serem mantidos para cada relação
  - Estrutura de armazenamento físico de cada relação em disco
  - Autorização de acesso
    - Tabelas, visões
      - Acesso e/ou manipulação

UFC
CC
DC

5. SQL
- Operações da DDL -

- Criando um banco de dados

  CREATE DATABASE nome-BD

- Abrindo um banco de dados (SQL Server)

  USE nome-BD

- Definindo espaço de armazenamento de um banco de dados
  - Slide 295

UFC
CC
DC

5. SQL
- Operações da DDL -

- Criando tabelas

```
CREATE TABLE nome-tabela
    (nome-coluna tipo-dado [not null] DEFAULT <valor-default>,
    [nome-coluna tipo-dado [not null] ... ],
    [CONSTRAINT nome-restrição]
      UNIQUE nome-coluna
      | PRIMARY KEY(nome-coluna {, nome-coluna})
      | FOREIGN KEY (nome-coluna {, nome-coluna})
        REFERENCES nome-tabela
            [ON DELETE CASCADE |
                        SET NULL | NO ACTION ],
            [ON UPDATE CASCADE],
      | CHECK (predicado)
    )
```

Definição de restrições

Especifica chaves candidatas

UFC
CC
DC

5. SQL
- Operações da DDL -

- Criando tabelas
  - Alguns tipos de dados suportados pelo SQL 99
    - char(n)
      - string de caracteres de tamanho fixo n
    - varchar(n)
      - string de caracteres de tamanho variável (máximo n)
    - bigint (8 bytes)
      - -2^{63} (-9,223,372,036,854,775,808) a 22^{63} -1 (9,223,372,036,854,775,807)
    - int (4 bytes)
      - -2^{31} (-2,147,483,648) a -2^{31} -1 (2,147,483,647)
    - smallint (2 bytes)
      - -2^{15} (-32,768) a -2^{15} -1 (32,767)
    - tinyint (1 byte)
      - 0 a 255

UFC
CC
DC

5. SQL
- Operações da DDL -

- Criando tabelas
  - Alguns tipos de dados suportados pelo SQL 99
    - decimal(p,d)
      - numérico com p dígitos
      - Dos p dígitos, d dígitos representam casas decimais após a vírgula
    - float e real
      - numérico ponto flutuante date
      - data de calendário
    - datetime
      - data e hora
        - vários formatos
          - YYYY-MM-DD HH:MM:SS
      - Função getdate()
        - Retorna data e hora corrente

UFC
CC
DC

5. SQL
- Operações da DDL -

- Removendo tabelas
  - Estrutura básica
    - DROP TABLE nome-tabela [CASCADE | RESTRICT]
    - Remove as tuplas da tabela e sua definição do catálogo
    - CASCADE remove as restrições do tipo foreign key das tabelas que referenciam a tabela removida
      - Oracle, Postgres, DB2
  - SQL Server
    - DROP TABLE nome-tabela

UFC
CC
DC

5. SQL
- Operações da DDL -

- Alterando tabelas

ALTER TABLE nome-tabela

[ADD nome-coluna tipo-dado [not null]]

[ALTER COLUMN nome-coluna tipo-dado [not null]]

[DROP column nome-coluna]

[ADD CONSTRAINT nome-restrição]

[DROP CONSTRAINT nome-restrição]

[DROP PRIMARY KEY]

[repetir ADD ou DROP em qualquer ordem]

5. SQL
- Operações da DDL -

- Alterando tabelas (SQL Server)

ALTER TABLE nome-tabela ADD
[nome-coluna tipo-dado [not null]]
[CONSTRAINT nome-restrição]
ALTER TABLE nome-tabela DROP COLUMN
[nome-coluna1, nome-coluna2, ...]

5. SQL
- Operações da DDL -

- Criar o BD de lojas, com as seguintes restrições
  - O salário de cada vendedor deve ser maior que 1000
  - Não podem existir valores de cpf e RG repetidos em Vendedores
  - Não se pode permitir a remoção de filiais para os quais ainda existam vendedores lotados
  - Não se pode remover itens de estoque e nem vendedores, para os quais existam vendas na tabela Histórico

UFC
CC
DC

5. SQL
- Operações da DDL -

- Alterar o BD lojas, com as seguintes restrições
  - O salário de cada vendedor deve ser maior que 1200
  - Ao alterar o código de uma filial, o valor do atributo codfil para todos empregados da filial deve ser automaticamente alterado
  - Incluir a coluna cnpj em Fornecedores e Filiais
    - Não podem existir valores de cnpj repetidos
    - cnpj não pode ser nulo

UFC
CC
DC

5. SQL
- Operações da DML -

- Incluindo tuplas em uma tabela
  - Cláusula Insert
    - Sintaxe

```
INSERT [INTO] nome_tabela
[(lista_de_colunas))]
{VALUES ( {DEFAULT | NULL | expr }[,...n] )
 | subquery }
```

UFC
CC
DC

5. SQL
- Operações da DML -

- Exercícios
  - Inserir uma tupla em Fornecedor com código igual a 1
    1. Adicionar o atributo NomeContato à tabela Fornecedor
       a. Não pode ser nulo
       b. Analise o que aconteceu
    2. Remova a tupla inserida e execute o passo 1
  - Inserir novamente uma tupla em Fornecedor com código igual a 1
    - Insira uma tupla em Fornecedor com valores diferentes para os atributos da tupla já inserida, exceto para o atributo código, que deve ser igual a 1
      - Analise o resultado

UFC
CC
DC

5. SQL
- Operações da DML -

- Exercícios
  - Inserir uma tupla em Filiais com código igual a 1
    - Inserir uma tupla em Vendedores com código de filial igual a 2
      - Analise o resultado
    - Inserir a tupla do passo anterior com código de filial válido
      - Analise o resultado
  - Remover a tupla de filial
      - Analise o resultado
  - Inserir uma tupla em estoque com todos valores do item com código igual a 1, exceto a chave primária
    - Utilize a opção de sub-consulta
  - Povoar o BD Lojas

5. SQL
- Operações da DML -

- Atualizando tuplas de uma tabela
  - Cláusula Update
    - Sintaxe

```
UPDATE nome_tabela
SET nome_coluna = {expr | NULL | (subquery) }
  {, nome_coluna = {expr | NULL | (subquery) }}
WHERE predicado
```

- Removendo tuplas de uma tabela
  - Cláusula Delete
    - Sintaxe

```
DELETE FROM nome_tabela
  WHERE predicado
```


5. SQL
- Operações da DML -

- Exercício
  - Aumentar o preço de venda dos produtos do fornecedor de código 1 em 10%

UFC
CC
DC

5. SQL
- Operações da DML -

- Consultas simples sobre o banco de dados

SELECT [ALL | DISTINCT] {* |
expr [[AS] c_alias]
{, expr [[AS] c_alias] ... }}
FROM nome-tabela [[AS] alias]
{, nome-tabela [[AS] alias] ...}
WHERE predicado

UFC
CC
DC

5. SQL
- Operações da DML -

- Consultas simples sobre o banco de dados
  - SELECT
    - Representa a operação de projeção da álgebra relacional
      - ALL (default)
        - Retorna todas as tuplas, inclusive repetidas
      - DISTINCT
        - Retorna apenas tuplas não repetidas
      - *
        - Retorna todos os atributos da(s) tabela(s)
      - expr
        - Atributo ou expressão matemática envolvendo atributos das tabelas
          - salario
          - salario*1.40

5. SQL
- Operações da DML -

- Consultas simples sobre o banco de dados
  - FROM
    - Representa a operação de produto cartesiano da álgebra relacional
      - Tabelas referenciadas
  - WHERE
    - Corresponde a operação de seleção da álgebra relacional
      - Predicado
        - Condição da seleção

5. SQL
- Operações da DML -

- Exemplo
  - listar o nome e o salário de todos os vendedores, lotados na filial 01. No resultado, os nomes da colunas devem ser Nome do Vendedor e Salário Líquido

```
Select nome as 'Nome do Vendedor', salario
as 'Salário Líquido'
from vendedores
where codfil=01
```

  - listar o código dos itens que tiveram vendas

```
Select distinct coditem
from historico
```

5. SQL
- Operações da DML -

- Predicados com operações sobre strings
  - Operador
    - like
  - Identificação de padrão
    - %
      - Casa com qualquer substring
    - _
      - Casa com qualquer caracter

5. SQL
- Operações da DML -

- Predicados com operações sobre strings
  - Exemplos (na cláusula where)
    - nome like 'inf%'
      - Retorna strings que iniciam pelo substring inf
    - nome like '%si_'
      - Retorna strings que contenham 'si' como substring e terminem com um caracter qualquer após 'si'
    - Listar todos vendedores com sobrenome 'brayner'

```
Select nome from vendedores
where nome like '%brayner%'
```

5. SQL
- Operações da DML -

- Consultas simples sobre o banco de dados
  - Retornar o nome dos vendedores e o nome de suas filiais de lotação
    - Junção

```
select v.nome as 'Nome Vendedor',
       f.nome as 'Nome Filial'
```
projeção

```
from vendedores v, filiais f
```
produto cartesiano

```
where codfil=cod
```
seleção

5. SQL
- Operações da DML -

- Exercícios
  - Listar vendedores com salários maior que 1000 e menor que 2000
  - Listar nome dos vendedores com o nome de sua filial de lotação e uma simulação de seu salário com um aumento de 15%
  - Listar nome do vendedor e o nome de sua filial de lotação, para todos vendedores que ganham mais que 4500

UFC
CC
DC

5. SQL
- Operações da DML -

- Exercícios (Desafio)
  - Listar nome dos fornecedores que tiveram itens vendidos por vendedores que ganham mais que 4500
  - Listar nome das filiais que venderam itens de fornecedores localizados na cidade de São Paulo

UFC
CC
DC

5. SQL
- Operações da DML -

- Consultas com o operador de união
  - Tabelas compatíveis
  - UNION
    - União de duas tabelas
      - Resultados de duas consultas
    - Sem repetições
  - UNION ALL
    - União de duas relações, com repetições
  - Exemplo
    - Liste o nome e cpf de todos os vendedores e dependentes existentes na empresa

```
select nome,cpf from vendedores UNION
select nome,cpf from Dependente
```

5. SQL
- Operações da DML -

- Consultas com o operador de interseção
  - Tabelas compatíveis
  - INTERSECT
    - Interseção entre duas tabelas
      - Sem repetições
- Consultas com o operador de diferença
  - Tabelas compatíveis
  - EXCEPT
    - Diferença entre duas tabelas
      - Sem repetições

5. SQL
- Operações da DML -

- Exercícios
  - Listar matrícula dos vendedores que não possuem vendas

```
Select matr From vendedores
EXCEPT
Select matrvend From historico
```

  - Listar matrícula dos vendedores que tiveram vendas

```
Select matr From vendedores
INTERSECT
Select matrvend From historico
```

5. SQL
- Operações da DML -

- Exercícios (Desafio)
  - Listar código dos fornecedores que não tiveram vendas
  - Listar matrícula de vendedores que não venderam produtos de fornecedores de São Paulo

5. SQL
- Operações da DML -

- Consultas ordenadas
  - ORDER BY coluna-resultado [ASC | DESC]
    {, coluna-resultado [ASC | DESC] ...}
  - Listar vendedores ordenados por salário na ordem decrescente e por nome na ordem crescente

```
Select *
from vendedores
order by salario desc, nome
```

UFC
CC
DC

5. SQL
- Operações da DML -

- Funções Agregadas
  - Funções aplicadas sobre uma coleção de valores (colunas) do banco de dados, retornando um escalar
  - sum
    - Retorna o somatório dos valores de uma coleção
  - avg
    - Retorna a média dos valores de uma coleção
  - max
    - Retorna o maior valor de uma coleção de valores

UFC
CC
DC

5. SQL
- Operações da DML -

- Funções Agregadas
  - min
    - Retorna o menor valor de uma coleção
  - count
    - Retorna o número de elementos de uma coleção
  - Sintaxe
    - nome-da-função (ALL | DISTINCT nome-coluna) | count(*)
    - Não podem ser utilizados na cláusula WHERE
    - Não é permitido o uso de função composta

5. SQL
- Operações da DML -

- Exercícios
  - Encontre o número de vendedores lotados na filial Recife

```
select count(*) from vendedores e, filiais d
where e.codfil=d.cod and d.nome like '_ec_fe'
```

  - Encontre o montante da folha de pagamento da empresa

```
select sum(salario) from vendedores
```

  - Encontre o salário médio pago pela empresa

```
select cast (avg(distinct salario) as decimal(6,2))
from vendedores
```


5. SQL
- Operações da DML -

- Exercícios (Desafio)
  - Encontre o número de vendedores que venderam itens de fornecedores localizados na cidade de Sobral
  - Encontre a quantidade de itens vendidos, que são fornecidos por fornecedores localizados na cidade de Sobral

5. SQL
- Operações da DML -

- Cláusula GROUP BY
  - Agrupar tuplas por valores de atributos
    - um ou mais atributos
  - Aplicar funções agregadas a diferentes grupos de tuplas
  - Exemplo
    - Quantidade de vendedores que cada filial tem
      - Agrupar tuplas de vendedores por código da filial
        - Um grupo para cada valor de codfil
      - Aplicar a função count a cada grupo

5. SQL
- Operações da DML -

- Exemplo Group by
  - Listar a quantidade de vendedores por filial

```
select codfil as 'Filial', count(*) as 'Número de
Vendedores'
from vendedores
group by codfil
```

A função count é aplicada para o conjunto de tuplas de cada grupo, no caso, codfil

Filial	Número de Vendedores
1	4
2	2

5. SQL
- Operações da DML -

- Atenção
  - Todas colunas que aparecem na cláusula select têm que aparecer na cláusula group by
    - Exceto os argumentos da funções agregadas
  - Exemplo de sintaxe incorreta

```
select codfil, matr, count(*) from vendedores
group by codfil
```


5. SQL
- Operações da DML -

- Exercício
  - Listar maiores e menores salários de cada filial. No resultado deve aparecer o nome da filial, o maior salário e o menor salário.

```
select f.nome, max(e.salario) as 'Maior Salario',
min(e.salario) as 'Menor_Salario'
from filial f, vendedores v
where cod=codfil
group by codfil, f.nome
```

5. SQL
- Operações da DML -

- Exercício
  - Mostrar a quantidade de itens vendidos e o número de vendas por vendedor e por item. Resultado deve estar ordenado pelo nome do vendedor.

```
select v.nome as 'Nome Vendedor', e.ref as 'Item',
sum(h.qtde) as 'Unidades Vendidas', count(*)
as 'Número de Vendas'
from estoque e, vendedores v, historico h
where e.cod=h.coditem and v.matr=h.matrvend
group by matrvend, v.nome, coditem, e.ref
order by v.nome
```


5. SQL
- Operações da DML -

- Cláusula having
  - Filtro de grupos
    - Selecionar grupos
  - Exemplo
    - Listar nome das filiais com média salarial maior que 2000, em ordem decrescente de média salarial

```
select f.nome, avg(salario)
from filial f, vendedores v
where cod=codfil
group by f.cod, f.nome
      having avg(v.salario)>2000
order by avg(salario) desc /* order by 2 desc */
```

5. SQL
- Operações da DML -

- Atenção
  - Consulta com where e having
    - Predicado da cláusula where é avaliado primeiramente
      - Só as tuplas, que satisfazem o predicado da seleção, são agrupadas pelo group by
      - Filtro de tuplas
    - Predicado da cláusula having é avaliado
      - Grupos, que satisfazem o predicado da cláusula having, aparecem no resultado
      - Filtro de grupos

5. SQL
- Operações da DML -

- Exercícios
  - Listar nome e média salarial das filiais que possuem mais de 4 vendedores

```
select d.nome, avg(e.salario) as 'Média Salarial'
from filial d, vendedores e
where d.cod=e.codfilial
group by d.cod, d.nome
   having count(matr)>=5
```

5. SQL
- Operações da DML -

- Exercícios
  - Listar nome da filial e quantidade de vendedores das filiais cuja média salarial é maior que 4000

```
select f.nome, count(*) as 'Número Vendedores'
from filial f, vendedores v
where cod=codfilial
group by f.cod, f.nome
    having avg(salario)>=4000
```


5. SQL
- Operações da DML -

- Exercícios
  - Gerar relatório com nome do vendedor, que efetuou vendas, a quantidade de vendas efetuadas por ele e a quantidade de itens vendidos, em ordem decrescente de quantidade de vendas. Só devem aparecer no relatório vendedores com volume de vendas superior a 1,000.00

```
select v.nome, count(*) as 'Total de Vendas'
from Vendedores v, historico h
where v.matr=h.matrvend
group by matr, v.nome
  having sum(h.qtde*h.prven) > 10000
order by 2
```

5. SQL
- Operações da DML -

- Exercícios (Desafio)
  - Gerar seguinte relatório: Para cada item vendido, mostrar referência do item, quantidade vendida, valor total das vendas, número de vendas e nome do fornecedor. Só devem aparecer no relatório itens vendidos por mais de um vendedor. Resultado deve estar ordenado em ordem decrescente por valor total de vendas e ordem crescente por referência.
  - Listar nome dos fornecedores, cujos itens foram vendidos em mais de duas filiais.

5. SQL
- Operações da DML -

- Checando valores nulos
  - Predicado IS NULL
    - Exemplo
      - select * from Empregado
      - where dt-nasc is null
- Atribuindo um valor a atributos com conteúdo NULL, em consultas
  - ISNULL(nome_atributo,0)

5. SQL
- Operações da DML -

- Consulta SQL aninhada (sub-consulta)
  - Consulta SQL embutida dentro de outra consulta SQL
  - Comporta-se de forma semelhante a uma sub-rotina (função ou método) dentro de um programa

5. SQL
- Operações da DML -

- Consulta SQL aninhada
  - Exemplo (BD lojas)
    - Listar vendedores com salário maior que a média salarial da empresa

```
select v.nome
from vendedores v
where salário > (select avg(salário) from
vendedores)
```

subconsulta retorna um conjunto de valores para a consulta mais externa (executada primeiro)

5. SQL
- Operações da DML -

- Consulta SQL aninhada
  - Listar o primeiro e segundo maiores salários da empresa

```
select max(salário) from Vendedores
union all
select max(salario) from Vendedores
where salário <> (select max(salário)
from Vendedores)
```

```
select max(salário) as 'Maior salário',
(select max(salario) from Vendedores
where salário <> (select max(salário)
from Vendedores)) as 'Segundo Maior
salário' from Vendedores
```


5. SQL
- Operações da DML -

- Consulta SQL aninhada
  - Listar nome das filiais com média salarial maior que a média salarial da empresa

```
select f.nome
from filiais f, vendedores v
 where f.cod=v.codfil
group by f.nome
 having avg(v.salário) > (select avg(salário)
from vendedores)
```

5. SQL
- Operações da DML -

- Predicado IN
  - Verifica a pertinência de elementos em um conjunto
  - Sintaxe
    - expr [NOT] IN (subconsulta) | expr [NOT] IN (val [,val ...])
  - Exemplo

```
select nome
from vendedores
where matr in (1,5,8,9)
```


5. SQL
- Operações da DML -

- Listar os vendedores lotados nas filiais localizadas na cidade de Fortaleza

```
select nome from vendedores
where codfil in (select cod from filiais
where cidade='Fortaleza' )
```

5. SQL
- Operações da DML -

- Consulta correlacionada
  - Listar empregados que possuem salário maior que a média salarial de suas filiais

```
select nome from vendedores
where salário > (select avg(salário)
from vendedores where codfil=????? )
```

A subconsulta precisa do valor do atributo codfil de cada tupla da consulta mais externa, como parâmetro de entrada

5. SQL
- Operações da DML -

- Consulta correlacionada
  - Listar empregados que possuem salário maior que a média salarial de suas filiais

```
select nome from vendedores v
where v.salário > (select avg(salário)
from vendedores vend
where vend.codfil = v.codfil )
```

Variável de correlação
Variável da consulta mais externa utilizada pela consulta mais interna

Executada para cada tupla da consulta mais externa

5. SQL
- Operações da DML -

- Predicados SOME, ANY e ALL
  - Implementam os quantificadores existencial e universal
    - Listar vendedores que ganham salários maior ou igual a média salarial de pelo menos uma filial
  - Sintaxe
    - expr θ {SOME | ANY | ALL} (subconsulta)
    - θ ∈ {<, <=,>, >=, =, <>}

5. SQL
- Operações da DML -

- Predicados SOME, ANY e ALL (cont.)
  - θSOME (subconsulta) e θANY (subconsulta)
    - Retornam verdade se e somente se
      - Para pelo menos um elemento s retornado pela subconsulta, <expr θ s> é verdade
    - Implementam o quantificador existencial
      - São equivalentes
    - Listar empregados com salários maior ou igual a média salarial de pelo menos uma filial

```
select nome from vendedores
where salário >=some (select avg(salário) from
vendedores group by codfil)
```

5. SQL
- Operações da DML -

- Predicados SOME, ANY e ALL (cont.)
  - θALL (subconsulta)
    - Retorna verdade se e somente se,
      - Para todo elemento s retornado pela subconsulta, <expr θ s> é verdade
    - Implementa o quantificador universal
    - Listar vendedores com salários maior ou igual a média salarial de cada filial

```
select nome from vendedores
where salário >=all (select avg(salário) from
vendedores group by lotação)
```


5. SQL
- Operações da DML -

- Predicados SOME, ANY e ALL (cont.)
  - Listar filial com maior média salarial
    - Não é permitido função agregada composta

```
select f.nome
from vendedores v, filiais f where f.cod=v.codfil
group by cod,f.nome
  having avg(salário) >=all (select avg(salário)
from vendedores group by codfil)
```

5. SQL
- Operações da DML -

- Exercícios (Desafio)
  - Listar nome da filial com maior volume financeiro de vendas
  - Listar os vendedores com maior volume financeiro de vendas para cada filial. No relatório deve aparecer o nome do vendedor e o nome da filial

5. SQL
- Operações da DML -

- Predicado EXISTS
  - Sintaxe
    - [NOT] EXISTS (subconsulta)
  - EXISTS (subconsulta)
    - Retorna verdade se e somente se
      - O conjunto retornado por subconsulta não é vazio
  - NOT EXISTS (subconsulta)
    - Retorna verdade se e somente se
      - O conjunto retornado por subconsulta é vazio

5. SQL
- Operações da DML -

- Predicado EXISTS (cont.)
  - Listar nome da filial com empregados ganhando duas vezes mais que a média salarial da filial

```
select f.nome
from filial f
where exists (select * from vendedores v
where v.codfil=f.cod and
salário > (2*(select avg(salário) from vendedores
where codfil=v.codfil)))
```


5. SQL
- Operações da DML -

- Predicado Between
  - Sintaxe
    - expr1 [NOT] BETWEEN expr2 and expr3
  - Exemplo
    - matr between 2 and 10 ⇔

      matr >= 2 and matr<=10

5. SQL
- Operações da DML -

- Formas de Junção em SQL
  - Listar nome dos departamentos com empregados com salários maior que 7000

```
select d.nome
from Empregado e, Departamento d
where d.cod_depart=e.lotação
      and salario>7000
```

projeção

produto cartesiano

seleção

$$\sigma_{d.cod_depart=e.lotação}(Departamento \times Empregado)$$

$$\Updownarrow$$

$$Departamento \bowtie_{d.cod_depart=e.lotação} Empregado$$

5. SQL
- Operações da DML -

- Sintaxe da cláusula FROM

```
[ FROM {<tabela_fonte>} [,...n] ]
<table_fonte> ::=
   nome_tabela [ [AS] qualificador ]  [ WITH ( <table_hint>
[,...n]) ]
   | nome_tabela [ [AS] qualificador ]  [ (column_alias [,...n] ) ]
   | (subquery) [AS] qualificador  [ (column_alias [,...n] ) ]
     | nome_visão [AS] qualificador  [ (column_alias [,...n] ) ]
   | <tabela_fonte> <tipo_junção> <tabela_fonte> ON
<condição_junção>


 <tipo_junção> ::=
   [ INNER | { { LEFT | RIGHT | FULL } [OUTER] } ] [<join_hint>]
    JOIN
```

5. SQL
- Operações da DML -

- Formas de Junção em SQL (cont.)
  - Sintaxe da cláusula FROM (cont.)
    - WITH (<table_hint> [,...n])])
      - Especifica estratégias (dicas) para o otimizador de consultas
        - Índices
        - tipo e granularidade de bloqueio (lock)
    - (column_alias [,...n])
      - Especifica alias para colunas retornadas de
        - uma tabela ou subconsulta
      - Um alias para cada coluna especificada na lista do select da subconsulta
    - join_hint
      - Indica qual o algoritmo de junção deve ser executado
        - (Nested) loop join, merge join ou hash join (SQL Server 2005)

5. SQL
- Operações da DML -

- Formas de Junção em SQL (cont.)
  - Sintaxe da cláusula FROM (cont.)
    - Tipos de junção
      - Junção theta
        - INNER JOIN
      - Junção externa à esquerda
        - LEFT OUTER JOIN
      - Junção externa à direita
        - RIGHT OUTER JOIN
      - Junção externa completa
        - FULL OUTER JOIN

5. SQL
- Operações da DML -

- Junção theta
  - Listar nome dos vendedores com o nome do respectivo filial

```
select e.nome, d.nome
from vendedores e inner join filiais d
on e.codfil=d.cod
```


5. SQL
- Operações da DML -

- Junção theta (INNER JOIN)
  - Para os vendedores que têm salário maior que 900, listar nome com o nome da respectiva filial

```
select empregado, filial
from filiais d inner join
(select nome, codfil from vendedores where
salário>900) e (empregado,filial)
on cod=filial
```

alias para colunas nome e lotação (resultado da subconsulta)

5. SQL
- Operações da DML -

- LEFT OUTER JOIN
  - Calcula o resultado da junção
    - Adiciona ao resultado tuplas da relação à esquerda, que não satisfazem a condição de junção
      - Atribui valores nulos aos atributos não definidos para estas duplas
  - Listar o histórico de vendas de cada vendedor (BD lojas)

```
select v.nome, e.referência,
d.qtde,d.qtde*d.pr_venda
from Vendedores v left outer join (historico
h inner join Estoque e on h.cod_item=e.e.cod
on v.matr=d.matr
```


5. SQL
- Operações da DML -

- Formas de Junção em SQL (cont.)
  - RIGHT OUTER JOIN
    - Para cada empregado, listar nome dele, nome do departamento de lotação e nome dos dependentes

```
select e.nome, d.nome, p.nome
from Departamento d inner join
(Dependente p right outer join Empregado e on
p.matr_resp=e.matr) on d.cod_dep=e.lotação
```

5. SQL
- Operações da DML -

- Formas de Junção em SQL (cont.)
  - FULL OUTER JOIN
    - Calcula o resultado da junção
      - Adiciona ao resultado da junção
        - Tuplas das relações envolvidas na junção que não satisfazem a condição de junção
        - Atribui valores nulos aos atributos não definidos para estas tuplas

5. SQL
- Visões -

- Acesso a um banco de dados
  - Requer conhecimento do esquema
    - Indesejável
      - Para usuários inexperientes
      - Desenvolvedores de aplicativos que acessam o BD
  - Por questões de segurança e privacidade
    - Grupos de usuários devem ter acesso a dados de interesse
      - O acesso a todo o banco de dados é perigoso

UFC
CC
DC

5. SQL
- Visões -

- Janelas sobre o banco de dados
  - Cada janela mostra parte do banco de dados
    - Diferentes visões sobre o banco de dados
- Visões (views)
  - Definidas sobre tabelas do banco de dados
    - Tabelas base
- Tipos de visões
  - Virtual
  - Materializada

UFC
CC
DC

5. SQL
- Visões -

- Visão virtual
  - A definição da visão é armazenada
    - Dados da visão não são persistentes
  - Sempre que referenciada
    - Os dados são materializados
      - Custo praticamente igual a cada materialização
  - Quanto ao acesso
    - Somente leitura
      - Visões que só permitem acesso de leitura
    - Permitem atualização
      - Visões que permitem atualizações nas tabelas base

UFC
CC
DC

5. SQL
- Visões -

- Visão materializada
  - Dados e definição são persistentes
  - Problema de atualização dos dados da visão
    - Sempre que há uma atualização nas tabelas base da visão
    - Recalculada
    - Atualizada
      - Com intervenção humana
      - Automática
  - Reduz custos de materialização de resultado
  - Visões somente para leitura
  - Aplicações
    - Implementação Data Warehouse
    - Integração de fontes de dados heterogêneas

UFC
CC
DC

5. SQL
- Visões -

- Definição de visões (SQL Server)

```
CREATE [ALTER] VIEW nome_da_visão
[(nome_coluna {, nome_coluna ...})]
[WITH <propriedade_da_visão> [ ,...n ] ]
AS expressão_SQL [WITH CHECK OPTION]
   <propriedade_da_visão> ::= {[SCHEMABINDING]
    [ENCRYPTION ] [ VIEW_METADATA ]  }
```

- WITH CHECK OPTION
  - Atualizações (INSERT ou UPDATE) na tabela base através da visão só serão permitidas se
    - Resultam em tuplas visíveis para a visão

UFC
CC
DC

5. SQL
- Visões -

- SCHEMABINDING
  - Associa a visão ao esquema das tabelas base da visão
    - Não permite que as tabelas base sejam alteradas (ou removidas), caso a definição da visão seja afetada
      - Remover coluna da tabela base que também aparece na definição da visão
    - Para permitir alteração das tabelas base
      - Alterar definição da visão ou remover a visão
  - Obrigatório para visões materializadas
  - Na expressão SQL da definição da visão
    - Nomes de tabelas, visões, funções têm que ser na forma <schema.objeto>
      - dbo.empregado

UFC
CC
DC

5. SQL
- Visões -

- ENCRYPTION
  - Criptografa a definição da visão em sys.syscomments
- VIEW_METADATA
  - Não permite a visualização do nome das tabelas base da visão

5. SQL
- Visões -

- Exemplos
  - Definir uma visão, que retorna o nome da filial em que o vendedor está lotado e o nome do vendedor

```
create view V1 (nome_filial, nome_vendedor)
as
select f.nome,v.nome
from filiais f inner join vendedores v
on cod=codfil

select * from V1
```


5. SQL
- Visões -

- Exemplos
  - Definir uma visão, que retorna o nome da filial e a quantidade de vendedores lotados na filial

```
create view V2 (nome_filial, num_vend)
as
select f.nome,
(select count(*) from vendedores where
f.cod=codfil)
from filiais f

select * from V2
```

5. SQL
- Visões -

- Exemplos
  - Definir visão que retorna matrícula, código da filial e salário dos vendedores que ganham menos que 2000

```
create view V3 (matrícula, filial, salário)
as
select nome,codfil,salario from vendedores
where salário<2000 with check option
```

```
select * from V3
```


5. SQL
- Visões -

- Acessando o banco de dados através da visão V3

matrícula	filial	salário
13	2	1500
99	3	1100

- Atualizando o banco de dados através de visões
  - update v3 set salário=salário+400
    - select * from V3
  - update v3 set salário=salário+2

    Erro!!

    tuplas a serem alteradas vão deixar de ser visíveis para V3

5. SQL
- Visões -

- Visões que permitem atualizações geralmente apresentam as seguintes restrições na subconsulta
  - Não estão especificadas as cláusulas group by e having
  - A palavra reservada distinct não está especificada
  - A cláusula where não contém subconsulta que referencia qualquer tabela na cláusula from diretamente ou indiretamente (via visões)
  - Todas as colunas da subconsulta são colunas simples
    - Não são permitidas colunas do tipo avg(salário) ou expressões aritméticas

5. SQL
- Visões -

- Visão materializada - SQL Server
  - Definir uma visão materializada, com o nome da filial e a quantidade de vendedores lotados na filial

```
Create view V4 (filial, num_vend)
with schemabinding
as
  select f.nome, count_big(*)
  from dbo.vendedores inner join dbo.filiais f
  on cod=codfil
  group by f.nome

Create unique clustered index I_V4
on V4 (filial)
```

5. SQL
- Visões -

- Exercícios
  - Insira um novo vendedor na filial de código igual a 2
    - Execute uma consulta sobre a visão materializada V4
  - Construa uma visão materializada que retorna a quantidade de itens no estoque por fornecedor

5. SQL
- Visões -

- Visão materializada - Postgres

  CREATE MATERIALIZED VIEW nome_da_visão
  [(nome_coluna {, nome_coluna ...})]
  [WITH <propriedade_da_visão> [,...n]]
  AS expressão_SQL [WITH CHECK OPTION]
- Atualização visão materializada - Postgres

  REFRESH MATERIALIZED VIEW [CONCURRENTLY]
  nome_da_visão [WITH [NO] DATA]

5. SQL
- Stored Procedure -

- Pedaços de códigos
  - Armazenados no banco de dados
  - Execução gerenciada pelo SGBD
  - Pode ter parâmetros de entrada e saída

5. SQL
- Stored Procedure -

- SQL Server

```
CREATE [ OR ALTER ] { PROC | PROCEDURE }
    [schema_name.] procedure_name [ ; number ]
    [ { @parameter [ type_schema_name. ] data_type }
     [VARYING] [ = default ] [OUT|OUTPUT}
[READONLY]m] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { [ BEGIN ] corpo_do_procedimento [;] [ ...n ]
[ END ] }
[;]
```

UFC
CC
DC

5. SQL
- Stored Procedure -

- SQL Server
  - Nome da procedure
    - Evite começar com sp_
  - Number
    - serve para agrupar stored procedures de mesmo nome
      - No caso do drop procedure, basta um comando

UFC
CC
DC

5. SQL
- Stored Procedure -

- SQL Server
  - @parameter
    - varying
      - Aplicado para parâmetros de saída, que são cursores
      - Especifica que o parâmetro de saída é um conjunto de tuplas e que o resultado pode variar a cada execução da procedure
    - = default
      - Especifica um valor default, para um parâmetro de entrada
        - Permite que o parâmetro de entrada não seja fornecido, quando o procedimento é disparado
    - Output
      - Especifica parâmetro de saída
      - Utilizado por outro procedimento ou aplicação que chama o procedimento, com parâmetro de saída
    - readonly
      - Especifica que o conteúdo do parâmetro não pode ser modificado dentro do código do procedimento

UFC
CC
DC

5. SQL
- Stored Procedure -

- SQL Server
  - Procedure options
    - Encryption
      - Definição do procedimento é criptografado
    - Recompile
      - Sempre que o procedimento é disparado
        - Um plano de execução é gerado
        - Não há armazenamento no buffer do plano de execução
    - Set Clause as
      - Especifica parâmetros de segurança para executar o procedimento
        - Quem pode disparar, ambiente de execução, etc

UFC
CC
DC

5. SQL
- Stored Procedure -

- SQL Server
  - For replication
    - Procedimento só pode ser disparado durante o processo de replicação
  - Corpo_do_procedimento
    - Transact-SQL
    - Common Language Runtime (CLR)
      - LPs do .net

5. SQL
- Stored Procedure -

- Exemplo
  - Procedimento que imprime a média salarial da filial, passado como parâmetro de entrada o código da filial

```
create procedure cal_med_sal_fil
  @codfil smallint
  as
    declare @media dec(10,2)
    select @media=avg(salario) from vendedores v
     where v.codfil=@codfil
    print 'Média Salarial da Filial ' +
    cast(@codfil as varchar(4)) + 'é ' +
    cast(@media as varchar(10))
```


5. SQL
- Stored Procedure -

- Desafio
  - Crie um procedimento P que passa, como parâmetro de saída, a média salarial da filial. O parâmetro de entrada de P é o código da filial
  - Crie um procedimento P' que tem como parâmetro de entrada o código FIL de uma filial. P' chama P, para que seja calculada a média salarial de FIL, e P' retorna como parâmetro de saída a média salarial calculada

UFC
CC
DC

5. SQL
- Funções -

- Pedaços de códigos
  - Armazenados no banco de dados
  - Execução gerenciada pelo SGBD
  - Devem possuir parâmetro de retorno
- SQL Server
  - Função que retorna escalar
  - Função que retorna uma tabela

UFC
CC
DC

5. SQL
- Funções -

- SQL Server
  - Função que retorna escalar

```
CREATE [ALTER] FUNCTION [ schema_name. ]
function_name
( [ { @parameter_name [ AS ][ type_schema_name. ]
data_type
    [ = default ] [ READONLY ] }   [ ,...n ]  ]  )
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN scalar_expression
    END
```

UFC
CC
DC

5. SQL
- Funções -

- SQL Server
  - Função que retorna uma tabela

```
CREATE [ALTER] FUNCTION [ schema_name. ]
function_name
( [ { @parameter_name [ AS ][ type_schema_name. ]
data_type
    [ = default ] [ READONLY ] }   [ ,...n ]  ]  )
RETURNS TABLE
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
        function_body
        RETURN (consulta_SQL)
    END
```


UFC
CC
DC

5. SQL
- Funções -

- SQL Server
  - Function options
    - schemabinding
    - encryption
    - returns null on null input
    - execute as clause

UFC
CC
DC

5. SQL
- Funções -

- Desafios
  - Especificar uma stored procedure no SQL Server, que gera um relatório com a premiação dada a vendedores por vendas efetuadas
    - Prêmio de R$ 100 por venda efetuada. O total do prêmio não pode ser maior que o salário do vendedor
    - Deve aparecer o nome do vendedor e da filial de lotação e o valor da premiação
  - Especificar uma função que, dado o código do item e a matrícula do vendedor, retorna o nome do item, o nome do vendedor e a quantidade vendida por ele do item fornecido.
    - Especifique uma consulta que utiliza a função acima, para retornar seu resultado

UFC
CC
DC

6. Acesso a BD via Aplicativos
- SQL Embutido -

- SQL é uma linguagem de consulta poderosa, contudo
  - Não é orientada a usuário inexperientes
    - Exige conhecimento da sintaxe e semântica de cláusulas e predicados
  - Não expressa ações imperativas (não declarativas)
    - Imprimir relatórios
    - Interação com o usuário
    - Mostrar resultado da consulta em uma interface amigável
- Embutir SQL em linguagens de programação
  - Viabilizar o acesso a BD através de programas aplicativos
    - Cobol, Fortran, Pascal, C e C++, Java, Python

UFC CC DC

6. Acesso a BD via Programas Aplicativos - SQL Embutido -

- SQL embutido (embedded SQL)
  - Código SQL executado dentro de um programa
  - Linguagem hospedeira (host language)
    - Linguagem de programação que hospeda SQL embutido
  - Processamento de consultas SQL embutido
    - Consultas são executadas pelo sistema de banco de dados
    - O resultado é disponibilizado para o programa
- Aplicativos com SQL embutido não podem ser compilados diretamente
  - Pré-compilação
    - Traduz comandos DML em chamadas a procedimentos (rotinas) na linguagem hospedeira

UFC CC DC

6. Acesso a BD via Programas Aplicativos - SQL Embutido -

- Estrutura de aplicativos com SQL embutido
  - Marcador de SQL embutido para o pré-compilador
    - Cláusula EXEC SQL
      - EXEC SQL <sql embutido> {END-EXEC | ;}
  - Cláusula INCLUDE
    - Identifica onde deve ser incluída uma estrutura de memória
      - SQLCA
        - SQL Communication Area
      - Variáveis especiais
      - Utilizada para a comunicação entre o programa e o sistema de banco de dados
      - EXEC SQL INCLUDE sqlca;

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- // SQL Communication Area - SQLCA

```
typedef struct SQLCA
{
unsigned char  sqlcaid[EYECATCH_LEN]; // string = 'SQLCA    '
long sqlcabc;                                          // tamanho de SQLCA
long sqlcode;                                          // SQL return code
short sqlerrml;                                  // Comprimento de SQLERRMC
unsigned char sqlerrmc[SQLERRMC_SIZ];   // Mensagem de erro
unsigned char sqlerrp[8];            // Reservado - Diagnostic Information
long sqlerrd[6];                    // array de seis inteiros de códigos de
status
unsigned char sqlwarn[8];                          // Warning flags
unsigned char sqlext[3];                           // Reservado
unsigned char sqlstate[5];                         // código de erros em
run-time
} SQLCA;
```


6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Estrutura de aplicativos com SQL embutido
  - Seção de declaração (declare section)
    - Parte do código onde são declaradas variáveis da linguagem hospedeira
      - Váriáveis hospedeiras
      - Variáveis utilizadas em cláusulas de SQL embutido
    - EXEC SQL BEGIN DECLARE SECTION
      - <variáveis hospedeiras>
    - EXEC SQL END DECLARE SECTION

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

```
#include <stdio.h>
#include "prompt.h"
exec sql include sqlca;
char cod_item[ ]="Entre com o código do item: ";
int main()
{ EXEC SQL BEGIN DECLARE SECTION;
    char referencia_saida[30], servidor_bd[20], usuario[20];
    int estoque,cod_itemint;
    float pr_venda;
  EXEC SQL END DECLARE SECTION;
  EXEC SQL CONNECT TO :servidor_bd USER :usuário;
while((scanf("%d",&cod_itemint))>=0)
  {EXEC SQL SELECT referencia,qtde,pr_venda
      INTO : referencia_saida,:estoque, :pr_venda
          FROM Estoque WHERE cod_item=:cod_itemint;
    printf("Referência: %s Quantidade: %5d  Preço: %5,2f",
    referencia_saida,estoque, pr_venda); }
     EXEC SQL DISCONNECT ALL;
    return (0); }
```


UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Resultado de uma consulta SQL
  - Conjunto de tuplas
- Variáveis hospedeiras
  - Utilizadas para recuperar colunas de uma tupla
  - Linguagens de programação só podem processar uma linha por vez
- Como uma linguagem de programação pode processar um resultado de uma consulta SQL?

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Problema conhecido como
  - Impedance Mismatching
- Solução
  - Mecanismo de cursor
    - Permite que o resultado de uma consulta SQL embutido seja obtido tupla a tupla
    - Compatibiliza a orientação de uma-registro-por-vez de LPs com a orientação a conjuntos do SQL embutido

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Declaração de uma variável do tipo cursor
  - EXEC SQL DECLARE nome_cursor CURSOR FOR subconsulta;
  - Resultado de subconsulta será associado ao "arquivo" temporário nome-cursor
- Manipulando uma váriável do tipo cursor
  - Abrindo um cursor (OPEN)
    - EXEC SQL OPEN nome_cursor;
    - Quando a cláusula OPEN é executada
      - A subconsulta especificada na declaração do cursor é processada
      - Posiciona cursor em uma posição anterior à primeira tupla
  - Recuperando uma tupla por vez
    - EXEC SQL FETCH nome_cursor INTO variáveis hospedeira {variável_hospedeira};
  - Fechando uma variável do tipo cursor
    - EXEC SQL CLOSE nome_cursor;

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Cláusula DECLARE CURSOR

  EXEC SQL DECLARE nome_cursor [INSENSITIVE] [SCROLL] CURSOR

  FOR subconsulta {UNION subconsulta}

  ORDER BY coluna [ASC | DESC]

  {, coluna [ASC | DESC] ... }

  [FOR READ ONLY | FOR UPDATE [OF nome_coluna {, nome_coluna ...}]

  - INSENSITIVE
    - Na hora do OPEN CURSOR é armazenado o resultado da subconsulta
      - Resultado estático
    - Alterações nas tabelas bases (update, delete, insert) não são refletidas em tuplas recuperadas com FETCH

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Cláusula DECLARE CURSOR (cont.)
  - SCROLL
    - Permite navegação em um cursor
      - Próximo
      - Anterior
      - Primeiro
      - Último, etc...
- Cláusula FETCH
    - EXEC SQL FETCH
    - [NEXT | PRIOR | FIRST | LAST | ABSOLUTE posição
    - | RELATIVE posição
    - nome_cursor INTO variáveis hospedeira {variável_hospedeira};

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Cláusula FETCH (cont.)
  - Exemplos
    - EXEC SQL FETCH PRIOR C1;
      - Retorna a tupla anterior a atual em C1
    - EXEC SQL FETCH ABSOLUTE 2 C1;
      - Retorna a segunda tupla em C1
    - EXEC SQL FETCH ABSOLUTE -2 C1;
      - Retorna a penúltima tupla em C1
    - EXEC SQL FETCH RELATIVE 5 C1;
      - Retorna a quinta tupla após a tupla corrente em C1
    - EXEC SQL FETCH RELATIVE -3 C1;
      - Retorna a terceira tupla antes da tupla corrente em C1

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

```
#include <stdio.h>
exec sql include sqlca;
int main()
{   EXEC SQL BEGIN DECLARE SECTION;
    char usuario[20], servidor_bd[20]; int cod; float pr_venda_c;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL CONNECT TO :servidor.bd USER :usuário;
    EXEC SQL DECLARE C1 CURSOR FOR SELECT cod_item,pr_venda
        FROM Estoque WHERE pr_venda < 500 FOR UPDATE pr_venda;
    EXEC SQL OPEN C1;
    EXEC SQL BEGIN TRANSACTION
    EXEC SQL SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
    EXEC SQL FETCH NEXT C1 INTO :cod,:pr_venda_c;
 while(SQLCODE==0)
    {pr_venda_c=pr_venda_c*1.15;
     EXEC SQL UPDATE estoque set pr_venda=:pr_venda_c where cod_item=:cod;
          EXEC SQL FETCH NEXT C1 INTO :cod,:pr_venda_c; }
 EXEC SQL CLOSE C1;
     EXEC SQL COMMIT TRANSACTION;
     EXEC SQL DISCONNECT CURRENT;
 return (0); }
```

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Embutido -

- Cláusula UPDATE
  - Update pesquisado
    - EXEC SQL UPDATE nome_tabela
      SET nome_coluna = {expr | NULL | (subquery) }
      {, nome_coluna = {expr | NULL | (subquery) }}
      WHERE predicado
  - Update posicionado
    - EXEC SQL UPDATE nome_tabela
      SET nome_coluna = {expr | NULL | (subquery) }
      {, nome_coluna = {expr | NULL | (subquery) }}
      WHERE CURRENT OF nome_cursor
    - Altera a tupla corrente (mais recentemente acessada)
    - Cursor precisa estar aberto
    - Nome da tabela da cláusula UPDATE e nome da tabela em DECLARE CURSOR devem ser o mesmo

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Dinâmico -

- SQL embutido
  - Consultas ao banco de dados são construídas em tempo de desenvolvimento
    - Já estão definidas em tempo de compilação
- Construir consultas em tempo de execução
  - Construir uma consulta SQL na forma de string
  - Montar uma consulta
    - Através da concatenação de vários substrings
  - SQL dinâmico

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- SQL Dinâmico -

```
#include <stdio.h>
#include "prompt.h"
exec sql include sqlca;
char cod_item[]="Entre com o código do item: ";
char
consulta[]="SELECT referencia,qtde,pr_venda FROM Estoque WHERE cod_item=?";
int main()
{   EXEC SQL BEGIN DECLARE SECTION;
    char referencia_saida[30], servidor_bd[20], usuario[20];
    int estoque,cod_itemint; float pr_venda;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL CONNECT TO :servidor_bd USER :usuário;
    EXEC SQL PREPARE exec_consulta FROM :consulta;
  while((prompt(cod_item,1,cod_item_str,9))>=0)
        {sscanf(cod_itemstr,"%d",&cod_itemint);
     EXEC SQL EXECUTE exec_consulta USING :cod_itemint
     printf("Referência: %s Quantidade: %5d Preço: %5,2f", referencia_saida,estoque, pr_venda); }
    EXEC SQL DISCONNECT ALL;
    return (0); }
```


UFC CC DC

6. Acesso a BD via Programas Aplicativos
- Cursor em Transact-SQL -

```
DECLARE cursorempregado CURSOR SCROLL FOR
SELECT matr,  (select sum(valor) from vencimento tv inner join
venc_emp vp on tv.cod_venc=vp.cod_venc group by matr having
matr=e.matr) as sal_bruto FROM empregado e
DECLARE @matricula smallint, @sal_bruto dec (13,2)
OPEN cursorempregado
FETCH NEXT FROM cursorempregado INTO @matricula, @sal_bruto
WHILE (@@fetch_status = 0)
BEGIN
PRINT cast(@matricula as nvarchar(10))
print cast(@sal_bruto as nvarchar(20))
FETCH NEXT FROM cursorempregado INTO @matricula, @sal_bruto
END
DEALLOCATE cursorempregado
```

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- Cursor em Transact-SQL -

- Escreva uma stored procedure, com cursor, com o seguinte requisito
  - Dada a consulta que retorna o número de vendas de cada vendedor em ordem crescente
  - Mostre este resultado em ordem decrescente, sem alterar a consulta

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- Call Level Interface -

- SQL embutido
  - Sentenças SQL são embutidas em um código fonte
    - Pré-compilação
    - Compilação
  - Um código fonte para cada BD a ser acessado
    - Todo código precisa ser pré-compilado e compilado
- Call Level Interface - CLI
  - Acesso a banco de dados através de chamadas a funções
  - Funções
    - Conectar e desconectar
    - Executar sentenças SQL
      - Sentença SQL é passada como parâmetro de uma função CLI

UFC CC DC

6. Acesso a BD via Programas Aplicativos
- Call Level Interface -

- Funções CLI são específicas por SGBD
  - Para cada SGBD
    - Um conjunto de funções específicas
      - Driver nativo de cada SGBD
- Exemplo de código C com função CLI

```
...
dbcmd(dbproc,"select cod_item from estoque");
resultado=dbsqlexec(dbproc);
dbresults(dbproc);
/* associando colunas do resultado à varáveis do programa
dbbind(dbproc, 1, INTBIND, (DBINT) 0, (BYTE *) &variável_1);
while (dbnextrow(dbproc) != NO_MORE_ROWS)
{ // Código fonte para processar tuplas }
```


UFC CC DC

6. Acesso a BD via Programas Aplicativos
- ODBC -

- Funções CLI são específicas para cada SGBD
- Open Database Connectivity
  - Padrão de APIs para garantir a conectividade entre clientes e servidores de BDs
  - Permite que clientes gerem comandos SQL para serem enviados para execução nos servidores
  - Aplicações de BDs fazem chamadas a funções ODBC
    - Funções
      - Conectar e desconectar
      - Executar sentenças SQL
        - Sentença SQL é passada como parâmetro de uma função ODBC
  - Qualquer cliente que possua uma interface ODBC pode se conectar a qualquer servidor que suporte a interface

6. Acesso a BD via Programas Aplicativos
- ODBC -

6. Acesso a BD via Programas Aplicativos
- JDBC -

- Tipos

DIRETO

ODBC-BRIDGE

UFC
CC
DC

6. Acesso a BD via Programas Aplicativos
- JDBC -

```
import java.sql.*;            //JDBC
import sun.jdbc.odbc.*;  // Bridge ODBC
public class Acesso_BD {
 public static void main (String[] args) {
  try
  {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Driver d = new JdbcOdbcDriver();
    java.sql.DriverManager.registerDriver(d);
  }
  catch (ClassNotFoundException e)
  {
    System.out.println("Driver não encontrado");
     return;
  }
```


UFC
CC
DC

6. Acesso a BD via Programas Aplicativos
- JDBC -

```
import java.sql.*;     //JDBC
public class Acesso_BD {
 public static void main (String[] args)
 { try
  {Class.forName("net.sourceforge.jtds.jdbc.Driver"); }

  catch (ClassNotFoundException e)
  {
    System.out.println("Driver não encontrado");
     return;
  }
```

Método que Carrega a classe JDBC Driver. Não é necessária para JDBC 4.0

6. Acesso a BD via Programas Aplicativos
- JDBC -

```
try
{
String url="jdbc:jtds:sqlserver://localhost:1433/RH;TDS=8.0";
Connection c= DriverManager.getConnection(url, "brayner",
"password");
  ...}
catch (SQLException e) { ... }
```

- Formato url do jTDS para SQL Server
  - jdbc:jtds:sqlserver://<server_name>[:<port>][/<database_name>] [;<property>=<value>[;...]]
    - <port> porta que o servidor está ouvindo
      - Default para o SQL Server é 1433
  - TDS (Tabular Data Stream)
    - Protocolo utilizado pelo SQL Server para se comunicar com clientes SQL Server (Default é 8.0)

6. Acesso a BD via Programas Aplicativos
- JDBC -

```
Statement st = con.createStatement
(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
ResultSet rs = stmt.executeQuery(
  "select e.nome,d.departamento,e.salario from Empregado where
salario>1000");
while (rs.next())
{
  String nome_emp = rs.getString(1);
  Char nome_dep[] = rs.getString(2).toCharArray();
  BigDecimal sal = rs.getBigDecimal("salario",2);
  System.out.println(nome_emp+" "+nome_dep+" "+sal);
};
stmt.close();
con.close();
```

Características do resultset

6. Acesso a BD via Programas Aplicativos
- JDBC -

- Resultset
  - Interface java
    - Contém o resultado de uma consulta SQL
  - Funcionalidade semelhante a de cursor
  - Métodos
    - https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html

6. Acesso a BD via Programas Aplicativos
- JDBC -

- Postgres
  - Tutorial interessante
    - http://www.postgresqltutorial.com/postgresql-jdbc/
  - Documentação
    - https://jdbc.postgresql.org/documentation/head/intro.html

UFC
CC
DC

7. Restrições de Integridade
- Conceitos Básicos -

- Restrições de Integridade em um BD Relacional
  - Restrição de domínio
  - Restrição de entidade
  - Restrição de chave
  - Restrição de integridade referencial
- Especificação de restrições de integridade
  - Não procedimental
  - Procedimental
- Especificação não procedimental de restrições
  - Através de expressões da DDL SQL
  - Impedem qualquer alteração no BD que violem as restrições

UFC
CC
DC

7. Restrições de Integridade
- Restrição de Domínio -

- Especificação não procedimental de restrições (cont.)
  - Restrição de domínio
    - O valor de cada atributo A
      - Tem que ser um valor atômico de dom(A)
    - Exemplo

```
create table Empregado
(matr integer not null,
  salário dec(9,2) not null
    :
    :
    :
constraint c1
 check (matr>0 and matr <=99999)
constraint c2
 check (salário>980.00))
```

UFC
CC
DC

7. Restrições de Integridade
- Restrição de Chave -

- Especificação não procedimental de restrições (cont.)
  - Restrição de chave
    - Para garantir que tuplas de uma relação sejam distintas entre si
      - Duas tuplas em uma relação não têm a mesma combinação de valores para seus atributos
    - Formas de especificação não procedimental
      - Primary key
      - Unique key
    - Exemplo

```
Create table Departamento
(cod_depart integer not null,
 nome       varchar(30) not null,
 ender      varchar(30),
 primary key (cod_depart))
```


UFC
CC
DC

7. Restrições de Integridade
- Restrição de Chave -

- Especificação não procedimental de restrições (cont.)
  - Restrição de integridade referencial
    - Um conjunto de atributos FK de uma tabela R é chave estrangeira (foreign key) se
      - Os atributos em FK têm o mesmo domínio que a chave primária PK de uma tabela S, e
      - Um valor de FK em uma tupla t1 de R
        - Ou ocorre em como valor de PK para uma tupla t2 em S
          - t1[FK] = t2[PK]
        - ou é nulo
    - Forma de especificação

```
[CONSTRAINT nome-restrição]
  FOREIGN KEY (nome-coluna {, nome-coluna})
  REFERENCES nome-tabela [ON DELETE CASCADE |
                          SET NULL | NO ACTION ],
                          [ON UPDATE CASCADE],
```

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Primeiras gerações de SGBDs
  - Armazenavam dados de forma passiva
    - Nenhuma ação era executada pelo SGBD
  - Ações sobre dados só eram executadas quando especificadas por transações
  - Muitas restrições de integridade deveriam ser mantidas por aplicativos
    - Total de salários por departamento deve ser menor que K
      - Ao se atualizar salários
        - Garantir a restrição de total de salários por departamento
      - Para cada novo valor K'
        - Alterar todos os programas que atualizam salários
    - Restrições que envolvam mais de uma tabela
      - Um dependente não pode está associado a mais de um empregado

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Regras ativas
  - Mecanismos que especificam ações a serem executadas quando certas condições são satisfeitas
  - Definidas inicialmente para sistemas de bancos de dados ativos
    - Armazenam dados e regras
    - Disparam e executam as ações definidas nas regras
- Regras E[C]\A
    - Na ocorrência de um EVENTO
    - Se CONDIÇÃO
    - Então execute AÇÃO
  - Quando um evento ocorre
    - Uma ou mais ações podem ser disparadas, caso
      - A condição seja satisfeita
  - Um evento pode representar uma atualização sobre dados do BD

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Triggers
  - Mecanismos de regras ativas implementados por SGBDs Relacionais existentes no mercado
  - São armazenados no banco de dados
  - Eventos
    - Insert, Delete, Update
  - Execução das ações
    - Antes ou depois da ocorrência do evento
      - SQL Server
        - Depois
    - A condição de execução seja satisfeita
      - Definida dentro de um trigger
  - Definidos através de uma expressão DLL
    - Create trigger

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Sintaxe da cláusula CREATE TRIGGER (SQL Server)

```
CREATE TRIGGER nome_trigger ON nome_tabela
{
   {FOR | AFTER | INSTEADOF { [DELETE] [,] [INSERT] [,]
   [UPDATE] }  [NOT FOR REPLICATION]
      AS
         sentença [ {sentença ... }]  }
   |
    {FOR { [INSERT] [,] [UPDATE] } [NOT FOR REPLICATION]
       AS
        { IF UPDATE (nome_coluna)
        [{AND | OR} UPDATE (nome_coluna)]
              [ { {AND | OR} UPDATE (nome_coluna) ...} ]
                    sentença [ {sentença ... }]  }
   }
```

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- FOR
  - Gatilho é disparado junto da ação
- AFTER
  - Disparo se dá somente após as operações relacionadas a ação SQL que o gerou seja concluída
    - Default
- INSTEAD OF
  - faz com que o trigger seja executado no lugar da ação que o gerou.

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- NOT FOR REPLICATION
  - Trigger não deve ser disparado quando o mecanismo de replicação altera a tabela envolvida no trigger
- Sentença
  - Especifica a condição e ação do trigger
    - Sentença SQL
      - Update
    - Sentença da linguagem proprietária do SGBD
      - T-SQL (SQL Server)
      - PL/SQL (Oracle)
- UPDATE(nome_coluna)
  - Indica que a ação do trigger deve ser disparada para uma atualização sobre a coluna nome_coluna

7. Restrições de Integridade
- Triggers -

- Observação importante
  - O SQL Server implementa as tabelas temporárias
    - Inserted
      - Armazena cópias das tuplas afetadas pelos comandos insert e update
    - Deleted
      - Armazena cópias das tuplas afetadas pelo comando delete

7. Restrições de Integridade
- Triggers -

- Exemplos
  - Garantir que menor salário de vendedores será 1000

```
create trigger tr1 on Vendedores
 for insert,update
as
 if  (select salario from inserted)<1000
   begin
     raiserror('salario invalido', 16,1)
     rollback transaction
   end
```

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Sintaxe da cláusula CREATE TRIGGER PostGres

```
CREATE TRIGGER name { BEFORE | AFTER | INSTEAD OF } { evento [ OR ... ] }
    ON table
      [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( condition ) ]
    EXECUTE PROCEDURE function_name ()
```

  - Evento
    - Insert, delete ou update
  - For each row ou for each statement
    - Gatilho disparado para tupla afetada pelo evento ou apenas uma vez, após a execução do comando SQL
      - update vendedores set salario=salario
        - Na opção for each statement, gatilho disparado uma única vez
  - função()
    - Procedimento que contem as ações a serem executadas
    - Sem parametros de entrada
    - Parametro de retorno do tipo trigger

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Desafios
  - Especificar as seguintes restrições de integridade no SQL Server
    - O volume de salários pagos em uma filial não pode ser maior que 15000
    - Salário de um vendedor não pode ser maior que duas vezes a média salarial da filial em que trabalha

UFC
CC
DC

7. Restrições de Integridade
- Triggers -

- Desafios
  - Especificar uma stored procedure no SQL Server, que gera um relatório com a premiação dada a vendedores por vendas efetuadas
    - Prêmio de R$ 100 por venda efetuada. O total do prêmio não pode ser maior que o salário do vendedor
    - Deve aparecer o nome do vendedor e da filial de lotação e o valor da premiação
  - Especificar uma função que, dado o código do item e a matrícula do vendedor, retorna o nome do item, o nome do vendedor e a quantidade vendida por ele do item fornecido.
    - Especifique uma consulta que utiliza a função acima, para retornar seu resultado

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Objetivos primários da tecnologia de banco de dados
  - Reduzir (eliminar) a redundância de dados
    - Reduzir inconsistência de dados
- Conhecimento a priori de restrições sobre dados
  - Desenvolvimento de ferramentas para atingir os objetivos supracitados
  - Exemplo
    - Considere a relação escala_de_vôos com esquema
      - escala_de_vôos(piloto, voo, data, horário)
    - Especifica qual piloto voará em dado vôo em uma dada data e a hora de decolagem
    - Nem toda combinação de pilotos, vôos, datas e horários é permitida em uma instância de escala_de_vôos

7. Restrições de Integridade
- Dependência Funcional -

- Exemplo (cont.)
  - Instância de escala_de_vôos

Piloto	Voo	Data	Horário
André	JJ3056	10.12.2016	8.00 am
Bárbara	JJ3599	10.12.2016	8.10 am
Caio	LH320	12.12.2016	8.00 am
André	JJ3056	12.12.2016	8.00 am
Yami	LH320	13.12.2016	8.00 am

7. Restrições de Integridade
- Dependência Funcional -

- Exemplo (cont.)
  - As seguintes restrições aplicam-se à relação tabela_de_vôos
    - Para cada piloto, data e horário existe apenas um vôo
    - Para cada vôo, existe apenas um horário de partida
    - Para um dado vôo e data, há apenas um piloto
  - Restrições
    - Vôo depende funcionalmente de {piloto, data, horário}
    - Horário depende funcionalmente de vôo
    - Piloto depende funcionalmente de {vôo, data}
    - Dependência funcional

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Definição
  - Generalização do conceito de chave
  - Seja o esquema de relação R com $\alpha \subseteq R$ e $\beta \subseteq R$. Dizemos que α determina funcionalmente β, $\alpha \rightarrow \beta$, se e somente se
    - Para todo e qualquer par de tuplas $t_1, t_2 \in R$,
      - Caso $t1[\alpha] = t2[\alpha]$, então $t1[\beta] = t2[\beta]$
    - β depende funcionalmente de α
- Exemplo
  - Considere a relação tabela_de_vôos
    - FD1: {piloto, data. horário} → vôo
      - Restrição é satisfeita em escala_de_vôos

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Algoritmo para verficar se uma relação R satisfaz uma dependência funcional $\alpha \rightarrow \beta$
  1. Ordenar R por α
  2. Se cada conjunto de tuplas com mesmo valor de α tem valores iguais para β, retorne verdade. Caso contrário, retorne falso.
- Exemplo
  - Considere a relação tabela_de_vôos
    - FD1: {piloto, data. horário} → vôo
      - Restrição é satisfeita em escala_de_vôos
      - Escala_de_vôos satisfaz FD1
    - FD2: vôo → partida
      - Escala_de_vôos satisfaz FD2
    - FD3: {vôo, data} → piloto
      - Escala_de_vôos satisfaz FD2

7. Restrições de Integridade
- Dependência Funcional -

- Exemplo
  - Considere a tabela r abaixo
    - Verifique se as seguintes dependências funcionais são satisfeitas
      - $A \rightarrow C$
      - $\{A,B\} \rightarrow D$
      - $C \rightarrow A$
      - $B \rightarrow C$

R

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b2	c2	d3
a3	b3	c2	d4
a4	b3	c2	d5

7. Restrições de Integridade
- Dependência Funcional -

- Axiomas de Armstrong
  - Seja uma relação r e α, β e γ subconjuntos de atributos de r
  - Regra de Inclusão
    - Se $\beta \subseteq \alpha$, então $\alpha \rightarrow \beta$
    - Prova. Considere α e β subconjuntos de atributos de uma relação r, onde $\beta \subseteq \alpha$. Suponha, por contradição, β não depende funcionalmente de α. Portanto, pode existir um par de tuplas t_1,t_2 em r, onde $t_1[\alpha]=t_2[\alpha]$ e $t_1[\beta]\neq t_2[\beta]$. Mas isto é um absurdo, pois significa que podem existir par de tuplas com valores de iguais para α, mas com valores diferentes para um subconjunto de α (no caso, β). Portanto, β depende funcionalmente de α, c.q.d.

7. Restrições de Integridade
- Dependência Funcional -

- Axiomas de Armstrong
  - Seja uma relação r e α, β e γ subconjuntos de atributos de r
  - Regra de reflexividade
    - $\alpha \rightarrow \alpha$
    - Prova. Basta aplicar a regra de inclusão, pois $\alpha \subseteq \alpha$.
  - Regra de transitividade
    - Se $\alpha \rightarrow \beta$ e $\beta \rightarrow \gamma$, então $\alpha \rightarrow \gamma$
    - Prova. Desafio (0,2 na 2a. aval)

7. Restrições de Integridade
- Dependência Funcional -

- Axiomas de Armstrong (cont.)
  - Seja uma relação r e α, β e γ subconjuntos de atributos de r
  - Regra de aumento
    - Se $\alpha \rightarrow \beta$, então $\alpha\gamma \rightarrow \beta\gamma$, onde $\alpha\gamma = \alpha \cup \gamma$ e $\beta\gamma = \beta \cup \gamma$
    - Prova. Considere α, β e γ subconjuntos de atributos de uma relação r, onde $\alpha \rightarrow \beta$. Suponha, por contradição, que αγ não determina funcionalmente βγ. Portanto, poderia existir um par de tuplas t_1 e t_2 em r, onde $t_1[\alpha\gamma]=t_2[\alpha\gamma]$ e $t_1[\beta\gamma]\neq t_2[\beta\gamma]$. Para que $t_1[\alpha\gamma]=t_2[\alpha\gamma]$, temos $t_1[\alpha]=t_2[\alpha]$ e $t_1[\gamma]=t_2[\gamma]$. Por outro lado, para que $t1[\beta\gamma]\neq t_2[\beta\gamma]$ seja verdade, necessariamente $t_1[\beta]\neq t_2[\beta]$, pois $t_1[\gamma]=t_2[\gamma]$. Contudo, $t_1[\beta]\neq t_2[\beta]$ é uma contradição, pois $\alpha \rightarrow \beta$. Portanto, necessariamente $t_1[\beta]=t_2[\beta]$. Consequentemente para todo par de tuplas em r, αγ determina funcionalmente βγ, c.q.d.

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Regras derivadas dos axiomas de Armstrong
  - Regra de união
    - Se $\alpha \rightarrow \beta$ e $\alpha \rightarrow \gamma$, então $\alpha \rightarrow \beta\gamma$, onde $\beta\gamma = \beta \cup \gamma$
  - Regra de decomposição
    - Se $\alpha \rightarrow \beta\gamma$, então $\alpha \rightarrow \beta$ e $\alpha \rightarrow \gamma$, onde $\beta\gamma = \beta \cup \gamma$
  - Regra de pseudo-transitividade
    - Se $\alpha \rightarrow \beta$ e $\gamma\beta \rightarrow \omega$, então $\alpha\gamma \rightarrow \omega$, onde $\alpha\gamma = \alpha \cup \gamma$ e $\beta\gamma = \beta \cup \gamma$
      - Prove inicialmente
        - Se $\gamma\beta \rightarrow \omega$, então $\beta\gamma \rightarrow \omega$
  - Demonstrações
    - Desafio (0,2 cada na 2a. aval)

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Exercício
  - Dado o esquema de relação R(A,B,C,D,E,F) e as seguintes dependências funcionais:
    - $A \rightarrow B$, $A \rightarrow C$, $CD \rightarrow E$, $CD \rightarrow F$ e $B \rightarrow E$
  - Verifique a existência das seguintes dependências funcionais em R
    - $A \rightarrow E$

      $A \rightarrow B$ e $B \rightarrow E$, assim, por transitividade, $A \rightarrow E$
    - $CD \rightarrow EF$

      $CD \rightarrow E$ e $CD \rightarrow F$, portanto, pela regra de união, $CD \rightarrow EF$
    - $AD \rightarrow F$

      $A \rightarrow C$ e $CD \rightarrow F$, portanto, por pseudo-transitividade, $AD \rightarrow F$

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Fechamento de conjuntos de dependências funcionais F
  - Seja F um conjunto de dependências funcionais válidos para uma relação r. O fechamento de F representa:
    - Conjunto de todas as dependências deduzidas a partir de F
      - F^+
      - Construído aplicando-se os axiomas de Armstrong e regras derivadas
- Fechamento de um conjunto de atributos
  - Seja α um subconjunto de atributos de uma relação r. O fechamento de α para um conjunto F de dependências funcionais é definido como:
    - Conjunto de todos os atributos determinados funcionalmente por α
      - α^+

UFC
CC
DC

7. Restrições de Integridade
- Dependência Funcional -

- Algoritmo para calcular α+ para um conjunto F de dependências funcionais

```
resultado:= α;
enquanto (resultado é alterado)
  para cada dependência funcional β→γ em F
    faça
      if β⊆resultado
          resultado:=resultado ∪ γ ;
```

- Prova. O laço é iniciado com α→resultado (reflexividade). Podemos incluir γ em resultado somente se β⊆resultado e β→γ. Mas, como resultado→β (regra de inclusão), por transitividade, α→β. Se α→β e β→γ, então α→γ (transitividade). Pela regra de união, pode-se afirmar que α→resultado ∪ γ.

7. Restrições de Integridade
- Dependência Funcional -

- Exercício
  - Dado o esquema de relação R(A,B,C,D,E,F) e o seguinte conjunto de dependências funcionais:
    - $A \rightarrow B$, $A \rightarrow C$, $CD \rightarrow E$, $CD \rightarrow F$ e $B \rightarrow E$
  - Calcule AD^+

7. Restrições de Integridade
- Dependência Funcional -

- Dependência Funcional Parcial
  - Se $\alpha\gamma \rightarrow \beta$
    - $\alpha \rightarrow \beta$ ou
    - $\gamma \rightarrow \beta$
- Dependência Funcional total
  - Se $\alpha\gamma \rightarrow \beta$
    - $\alpha \rightarrow \beta$ e
    - $\gamma \rightarrow \beta$

UFC
CC
DC

8. Normalização
- Conceitos básicos -

- Critérios de qualidade em projetos de BDs relacionais
  - Junção sem perda
  - Manutenção de dependências
  - Diminuir redundância
  - A garantia dos critérios de qualidade
    - Reduz anomalias durante
      - Inclusão
      - Atualização
      - Remoção
- Estratégia de verificação da qualidade do projeto de BD
  - Normalização
- Definição
  - Processo para analisar esquemas de relações com base nas dependências funcionais e chaves primárias definidas
    - Para garantir os critérios de qualidade

UFC
CC
DC

8. Normalização
- Formas Normais -

- Primeira Forma Normal (1FN)
  - Domínio de valores de atributos de uma relação
    - Apresenta valores atômicos
  - Valor de qualquer atributo em uma tupla
    - Representa um único valor no domínio do atributo
  - Objetivo
    - Evitar aninhamento de relações
    - Atributos multivalorados
  - Exemplo de relação que viola 1FN
    - Empregado(matr,nome,salário,telefones, (cod_dep,nome_dep))

8. Normalização
- Formas Normais -

- Segunda Forma Normal (2FN)
  - A relação tem que estar em 1FN
  - Se k é chave primária da relação, então nenhum atributo não chave
    - Mantém uma relação de dependência funcional parcial com k
  - Exemplo de relação que viola 2FN
    - Nota_Fiscal(num_nota, cod_item, qtde_ven, pr_ven, dt_emi, valor_icms)
      - {num_nota,cod_item}→ valor_icms
        - Dependêncial parcial
          - cod_item → valor_icms

8. Normalização
- Formas Normais -

- Terceira Forma Normal (3FN)
  - Dado um conjunto de dependências funcionais F, uma relação R está em 3FN, se R está na 2FN e para todas dependências de F^+ da forma $\alpha \rightarrow \beta$, com $\beta \subseteq R$ e $\alpha \subseteq R$, pelo menos umas das seguintes condições é garantida
    - $\alpha \rightarrow \beta$ é uma dependência funcional trivial
    - α é chave de R
    - Não há atributos A fora da chave com dependência funcional transitiva com esta

8. Normalização
- Formas Normais -

- Forma Normal Boyce-Codd (BCFN)
  - Dado um conjunto de dependências funcionais F, uma relação R está em BCFN, se para todas dependências de F^+ da forma $\alpha \rightarrow \beta$, com $\beta \subseteq R$ e $\alpha \subseteq R$, pelo menos umas das seguintes condições é garantida
    - $\alpha \rightarrow \beta$ é uma dependência funcional trivial
      - $\beta \subseteq \alpha$
    - α é chave de R
  - Intuitivamente
    - Todos atributos não chave só mantem relação de dependência funcional com a chave

8. Normalização
- Formas Normais -

- 3FN e BCNF
  - Se R está na BCNF
    - R está na 3FN
    - O inverso não é verdade
  - 3FN é menos restritiva que a BCNF
    - Exemplo
      - Conta(num_conta, saldo, cod_cliente, cod_ag)
        - A superchave é {num_conta,cod_cliente,cod_ag}
        - Se não houver repetição de número de conta no banco
          - num_conta→saldo
            - num_conta não é a super-chave
              - Conta não está na BCFN, mas está na 3FN

8. Normalização
- Formas Normais -

- 3FN e BCNF
  - Na prática
    - A grande maioria de relações que está na 3FN
      - É também BCNF
- Modelagem de BDs utilizando o MER e o DR
  - Garante o projeto lógico na 3FN
    - Não há demonstração forma, mas ainda não foi apresentado um contra exemplo

8. Normalização
- Formas Normais -

- Exercício
  - Considere a seguinte definição da tabela curso_disc do BD ACAD

```
create table curso_disc (
cod_curso int not null,
cod_disc int not null,
constraint FK_curso_disc foreign key (cod_curso)
references curso,
constraint FK_disc_c foreign key (cod_disc) references
disciplina
on delete no action
on update cascade)
```

- Em que forma normal se encontra a tabela curso_disc acima?
  - A tabela não possui chave
    - Está na 1FN

8. Normalização
- Formas Normais -

- Exercício
  - Considere a seguinte definição da tabela curso_disc do BD ACAD

```
create table curso_disc (
cod_curso int not null,
cod_disc int not null,
constraint PK_curso_disc
primary key (cod_curso,cod_disc),
constraint FK_curso_disc foreign key (cod_curso)
references curso,
constraint FK_disc_c foreign key (cod_disc) references
disciplina on delete no action on update cascade)
```

  - Em que forma normal se encontra a tabela curso_disc acima?
    - BCNF
    - Consequentemente 3FN

9. Armazenamento de Dados
- Armazenamento em Disco -

9. Armazenamento de Dados - Armazenamento em Disco -

9. Armazenamento de Dados - Armazenamento em Disco -

Surface (2 per platter)

- Covered with a thin layer of magnetic material
- Organized into tracks
  - Concentric circles

Track

- Circular path on disk surface
- Stores information (bits)
- Subdivided into Sectors

Sector

- Smallest physical storage unit on the disk
  - Read/Write unit
  - Indivisible unit as far as errors are concerned
    - Should a small region *R* of the magnetic coating be corrupted in some way
      - The entire sector containing *R* is corrupted as well
        - It can not store information anymore
- Each sector stores the same amount of data
  - 4096 Bytes

Gap

- non-magnetized area
- Separates sectors
- ~10% of the track

UFC
CC
DC

9. Armazenamento de Dados
- Armazenamento em Disco -

- Acessando uma unidade de disco
  - O motor de rotação faz o disco girar a uma velocidade constante
  - Cada superfície do disco possui um cabeçote de leitura/escrita
  - O cabeçote está montado em um dispositivo chamado braço do disco
    - O braço movimenta-se transversalmente pelo disco para acessar as diferentes trilhas do disco.
  - Uma unidade de disco possui geralmente um conjunto de discos
    - Os cabeçotes de leitura/escrita de todas as superfícies de discos de uma unidade movimentam-se em conjunto (simultaneamente) e são acionadas pelo suporte do braço.

UFC
CC
DC

9. Armazenamento de Dados
- Armazenamento em Disco -

- Medidas de Desempenho de Discos
  - Tempo de procura (TP)
    - Tempo gasto para posicionar o braço sobre a trilha correta
  - Tempo médio de latência rotacional (TL)
    - Tempo médio para que o setor desejado passe pelo cabeçote de leitura/escrita. Definido como 1/2 do tempo de uma rotação. Se r é a taxa de rotações/segundo, então
      - $T=(2r)^{-1}$
  - Taxa de transferência de dados (TTr)
    - Taxa na qual os dados são lidos/escritos por segundo. Se N é a capacidade das trilhas, então
    - $TTr= rN$
  - Tempo de transferência de dados de um setor
    - Tempo (segundos) necessário para transferir dados de um setor. Se n é o tamanho de um setor, então
    - $TT= n(rN)^{-1}$

UFC CC DC

9. Armazenamento de Dados
- Armazenamento em Disco -

- Discos Magnéticos (cont.)
  - Medidas de Desempenho de Discos
    - Tempo de acesso a um setor (TA)
      - Tempo gasto desde um pedido de leitura/escrita até o fim da transferência de dados
      - A fórmula para determinar o tempo de acesso de um disco é dada por:
        - TA=TP + TL + TT
        - $TA= TP + (2r)^{-1} + n(rN)^{-1}$
  - Acesso a disco é lento

UFC CC DC

10. Projeto Físico de Banco de Dados
- Conceitos Básicos -

- Definição da configuração física do banco de dados
  - Localização dos dados em memória secundária
    - Arquivos
      - Armazenados em diferentes dispositivos e/ou meios físicos de armazenamento
  - Definição de índices e de sua localização física
  - Particionamento de tabelas

10. Projeto Físico de Banco de Dados
- Organização Lógica de Banco de Dados -

- SQL Server

10. Projeto Físico de Banco de Dados
- Definição do Espaço de Armazenamento -

- SQL Server
  - CREATE DATABASE nome_banco_de_dados
    [ON [PRIMARY]
    [<especifica_arquivo> [,...n]]
    [, <filegroup> [,...n]]]
    [LOG ON { <especifica_arquivo> [,...n]}]
  - <especifica_arquivo> ::= ([NAME = nome_do_arquivo,]
    FILENAME = 'nome_arquivo_com_path'
    [, SIZE = tamanho_inicial]
    [, MAXSIZE = { tamanho_max | UNLIMITED }]
    [, FILEGROWTH = taxa_crescimento])
  - <filegroup> ::= FILEGROUP nome_filegroup
    <especifica_arquivo> [,...n]

UFC CC DC

10. Projeto Físico de Banco de Dados
- Definição do Espaço de Armazenamento -

- SQL Server
  - PRIMARY
    - Especifica o filegroup primário do banco de dados
      - Armazena todas as tabelas de sistema (controle)
    - Primeiro arquivo da lista
      - Arquivo primário
        - Contem dados metadados e dados de controle do banco de dados.
      - Extensão .mdf
  - LOG ON
    - Especifica onde será armazenado o log do BD
      - Extensão .ldf
  - Arquivos secundários
    - Extensão .ndf

UFC CC DC

10. Projeto Físico de Banco de Dados
- Organização Lógica de Banco de Dados -

- Oracle

Pages (2/4/8/16/32 kb)

UFC CC DC

10. Projeto Físico de Banco de Dados
- Definição do Espaço de Armazenamento -

- Oracle

Um único Arquivo Até 1022 arquivos

```
CREATE [BIGFILE / SMALLFILE]
TABLESPACE Nome-TB
DATAFILE 'C:/db/data01.dbf' SIZE 10M REUSE
AUTOEXTEND ON NEXT 1M MAXSIZE 1G,
'C:/db/data02.dbf' SIZE 5M,
'C:/db/data03.dbf' REUSE
BLOCKSIZE 2k/4k/8k/32k
ONLINE/OFFLINE
PERMANENT/TEMPORARY
EXTENT MANAGEMENT LOCAL / AUTOALOCATE
```

Arquivo não pode existir

Arquivo tem existir

UFC CC DC

10. Projeto Físico de Banco de Dados
- Definição do Espaço de Armazenamento -

- Oracle
  - Arquivos
    - Sistema de arquivo do SO
    - Automatic Storage Management (ASM)
      - Sistema de arquivos proprietário do Oracle
        - Discos gerenciados pelo ASM

UFC CC DC

10. Projeto Físico de Banco de Dados
- Definição do Espaço de Armazenamento -

- Postgres

```
CREATE TABLESPACE Nome-TB
 [OWNER {new_owner|CURRENT_USER|SESSION_USER }]
    LOCATION 'directory'
    [ WITH ( tablespace_option = value [, ... ] ) ]
```

  - tablespace_option
    - seq_page_cost, random_page_cost, effective_io_concurrency

UFC CC DC

10. Projeto Físico de Banco de Dados
- Indexação -

- Which is the most efficient procedure to find a given word in a book?
  - Go to the book's index
  - Search for the word in the index structure
    - The entries in the index structure are ordered
  - If there is an entry in the index equal to the word we are looking for
    - The entry has the address of the pages which contain the word
  - Otherwise, there is no occurrence for that word in the book

10. Projeto Físico de Banco de Dados
- Indexação -

- Each index entry
  - Has a search key
    - One or more attributes
  - Pointer to pages which contain tuples with the value of the search key
- Reduces the amount of disk access for finding a subset of tuples

10. Projeto Físico de Banco de Dados
- Indexação -

- Classification
  - Ordered index
    - The search keys are ordered in the index structure
    - Primary (Clustered)
      - The table is physically sorted by the search key
      - Dense
        - One entry for each tuple
      - Sparse
        - One entry for each data page
    - Secondary (Non-clustered)
      - The table is not physically sorted by the search key
      - Dense

10. Projeto Físico de Banco de Dados
- Indexação -

- Classification (cont.)
  - Non-ordered index
    - The search keys are not ordered in the index structure
    - Hash index

10. Projeto Físico de Banco de Dados
- Indexação -

- Why a secondary index can not be implemented as a sparse index ?
- Differentiate the concepts of Primary Index and Primary Key

10. Projeto Físico de Banco de Dados - Indexação -

- Hash Index
  - h= ⌊ID/3⌋

How many disk accesses are necessary to process the following query?
Select * from Employee where ID=7

10. Projeto Físico de Banco de Dados - Indexação -

- B+-Tree
  - Balanced tree
    - All paths from the root to the leaves have the same amount of nodes
      - Same length
  - n-ary tree
    - n children per node
      - Variable
    - n represents the order of tree (fanout)
      - Branching factor

10. Projeto Físico de Banco de Dados - Indexação -

- Rules for a B^+-Tree of order n
  - The root has at least two pointers
  - Internal nodes have at most n pointers, which point recursively to a B^+-Tree
    - Structure
      - $<PT_1, K_1, PT_2, K_2, \ldots, PT_{m-1}, K_{m-1}, PT_m>$, where $m \leq n$
    - n-1 search keys
  - Internal nodes have at least $\lceil n/2 \rceil$ pointers
    - $\lfloor ((n-1)/2) \rfloor$ keys

10. Projeto Físico de Banco de Dados - Indexação -

- Rules for a B^+-Tree of order n (cont.)
  - Leaf node
    - At most n-1 and at least $\lfloor (n/2) \rfloor$ pointers are used to pointer to data pages
    - The last pointer points to the next node
  - At each node
    - $K_1 < K_2 < K_3 < \ldots < K_{m-1}$, where $m \leq n$
  - For every search key value X belonging to subtree pointed by PT_i
    - For $i=1$, $X \leq K_1$
    - For $1<i<m$, $K_{i-1} < X \leq K_i$
    - For $i=m$, $X > K_{m-1}$

10. Projeto Físico de Banco de Dados - Indexação -

```
B+_Tree ( )
    Allocate a new node R; /* the root */
    While there are entries
        do
            insert_new_entry (L, K); /* L is the node address where to insert k */
    End-While;
insert_new_entry (L, K)
If L has n-1 search-keys      /* overflow */
        Allocate new node L' sibling of L;  /* node split */
        If Parent(L) ≠ ∅      /* L has a parent */
            /* the median search-key value M should be inserted into Parent(L) */
            insert_new_entry (Parent(L), M)
        Otherwise
            Allocate a new node R, where R is the parent of L and L´
            /* the median search-key value M should be inserted into R */
            insert_new_entry (R, M)
            /* In R,  the pointer PT1 points to L and PT2 points to L' */
        End-If
        search-key values X ≤ M remain in node L;
        search-key values Y > M should be inserted into L';
Otherwise
        allocate K in L
```


10. Projeto Físico de Banco de Dados - Indexação -

- Build a B+-Tree of order 3 with the following insertion sequence: 8,5,1,7,3,12,9,6

10. Projeto Físico de Banco de Dados - Indexação -

- Build a B+-Tree of order 3 with the following insertion sequence: 8,5,1,7,3,12,9,6

10. Projeto Físico de Banco de Dados - Indexação -

- The height should be computed for the worst-case
  - Each node contains the minimum number of pointers
    - $\lceil n/2 \rceil$
- There is a relation between a level L and number of nodes in L in a B+-Tree

  L=0; M=1
  L=1; M=n/2
  L=2; M=$(n/2)^2$
  L=3; M=$(n/2)^3$
  ⋮
  L=h; M=$(n/2)^h$;
  $\log_{n/2} M = \log_{n/2} (n/2)^h$

  $h = \log_{n/2} M$

Log M
to the base n/2

x/2 x over 2

10. Projeto Físico de Banco de Dados - Indexação -

- Each node corresponds to page
  - Let T be the size of a page in bytes, PT be the size of a pointer in bytes and S the size of a search key in bytes
    - Order of a B^+-Tree
      - **$T=n*PT+(n-1)*S$**

10. Projeto Físico de Banco de Dados - Indexação -

- Complexity

 # read page to find a tuple with key K_2 = 1+ $\log_{n/2}$ M
 # written pages to insert a new entry K'= 1+ $\log_{n/2}$ M

10. Projeto Físico de Banco de Dados - Indexação -

- Árvores B+ (cont.)
  - Remoção de entradas em um nó

Função remoção_entrada(L, chave_busca)
Se L é nó raiz
  remoção_entrada_raiz(L, chave_busca)
Senão
  remoção_entrada_nó(L, chave_busca)
Fim;

10. Projeto Físico de Banco de Dados - Indexação -

- Árvores B+ (cont.)

Função remoção_entrada_raiz(L, chave_busca)
Se L possui apenas uma entrada <chave,PD> /* L possui 2 filhos */
  S=filho(L) /* retorna filho de L que possui no mínimo (n/2) entradas */
    Se S ≠ ∅
    Se S=filho_esquerda(L)
      /* Inserir na raiz maior valor de chave de busca em S toda subárvore */
      remoção_entrada(S, K)
      inserção_nova_entrada(L, K)
    Senão
      /* Inserir na raiz menor valor de chave de busca em S */
      remoção_entrada(S, K)
      inserção_nova_entrada(L, K)
  Senão /* fundir os nós filhos de L em um nó R que será a nova raiz da árvore */
Senão
  remover entrada <Ki> de L
  Se Pi ≠ NULL ou Pi+1 ≠ NULL
    atualizar_chave busca(L,i)

10. Projeto Físico de Banco de Dados - Indexação -

```
Função remoção_entrada_nó(L, k)
remover entrada <K> de L;
Se nó L possui ((n- 1)/2) entradas <k>  /* underflow */
   L'= irmão (L);  /* retorna  irmão de L que possui (n/2) entradas <k> */
   Se L' ≠ ∅
    inserção_nova_entrada(L,k);   /*inserir V (valor entre L e L'  no pai(L)) em L */
    Se L'=irmão_esq(L) /* Substituir V no nó pai por maior valor de chave em L' */
      remoção_entrada(L', K); inserção_nova_entrada(Pai(L), K)
    Senão   /* Substituir V' no nó pai  por menor valor de chave de busca  em   L' */
      remoção_entrada(L', F); inserção_nova_entrada(Pai(L), F);
    Senão  /* Irmãos de L contêm exatamente n/2 - 1 entradas <k> */
    Se L'=irmão_esq(L)
      inserção_nova_entrada(L', V, PD); /*inserir V  em L' */
      remoção_entrada(Pai(L), V, PD); /*remove V de Pai(L) */
      inserção_nova_entrada(Pai(L), K); /*Substituir V  em Pai(L) por maior valor  K em L*/
      Transferir entradas de L para L'; remover nó L;
    Senão
      inserção_nova_entrada(L, V, PD) /*inserir V  em L' */
      remoção_entrada(Pai(L), V, PD); /*remove V de Pai(L) */
      inserção_nova_entrada(Pai(L), K) /* Substituir V  no nó pai por maior chave em L' */
      Transferir entradas de L' para L; remover nó L';
Senão
    Se PTi ≠ NULL ou PTi+1 ≠ NULL
       atualizar_chave  busca(L,PTi);
```


9. Armazenamento e Indexação - Indexação -

```
Função atualiza_chave_busca(L, PTi)
S=filho(L, PTi) /*retorna verdade se filho de L apontado por PTI tem  (n/
   2) entradas                <chave,PD> */
S´= filho(L, PTi+1)
 Se (S) /* Inserir na raiz  maior  valor de chave de busca da subárvore apontada
   PTI*/
     remoção_entrada(L´, K)
     inserção_nova_entrada(L, K, PD)
   Senão
        Se (S´) /* Inserir na raiz  menor  valor de chave da subárvore apontada
PTi+1 */
          remoção_entrada(L´, K)
            inserção_nova_entrada(L, K, PD)
     Senão
            /* fundir os nós filhos de L em um nó S que será a nova raiz da
      árvore */
```

UFC CC DC

10. Projeto Físico de Banco de Dados
- Indexação -

- Definindo-se índices no SQL Server

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]
INDEX nome_índice
ON nome_tabela (nome_coluna [, {nome_coluna}])
[With (PAD_INDEX= ON | OFF]
[FILLFACTOR = taxa_preenchimento_nós)]
[ON filegroup]

UFC CC DC

10. Projeto Físico de Banco de Dados
- Indexação -

- Definindo-se índices no SQL Server (cont.)
  - UNIQUE
    - Não permite tuplas com valores de índices repetidos
  - CLUSTERED
    - Índice primário
  - NONCLUSTERED
    - Índice secundário
    - Máximo de 249 por tabela
  - nome_coluna
    - Atributo que funcionará como chave de busca
    - Máximo por tabela
      - 16 colunas
      - 900 bytes

10. Projeto Físico de Banco de Dados
- Indexação -

- Definindo-se índices no SQL Server (cont.)
  - FILLFACTOR
    - Parâmetro que define a utilização de espaço de um nó (página) no momento de criação de um índice
      - Utilizado apenas durante a criação de índice
    - Fornecido como percentual de preenchimento
    - Qualquer inteiro no intervalo [1,100]
    - Exemplo
      - FILLFACTOR=30
        - Indica que apenas 30% do tamanho da página será utilizado durante a criação de índice
  - PAD_INDEX ON
    - FILLFACTOR vale para nós folhas e nós internos
  - PAD_INDEX OFF
    - FILLFACTOR vale apenas para nós folhas
      - Default

10. Projeto Físico de Banco de Dados
- Indexação -

- Definindo-se índices no SQL Server (cont.)
  - Ordem da árvore
    - $T=(n*P) + ((n-1) *C)$
    - T no SQL Server corresponde a 8KB (96 bytes para o header)

10. Projeto Físico de Banco de Dados
- Indexação -

- Definindo-se índices no PostGres

CREATE [UNIQUE] INDEX [CONCURRENTLY] [[IF NOT EXISTS] nome_indice] ON nome_tabela [USING tipo_estrutura_índice]

({ nome_coluna | (expression) } [ASC | DESC] [NULLS { FIRST | LAST }] [, ...])

[WITH (parametro_armazenamento = value [, ...])]

[TABLESPACE nome_tablespace]

[WHERE predicado]

10. Projeto Físico de Banco de Dados
- Indexação -

- Definindo-se índices no PostGres
  - Índice secundário
  - tipo_estrutura_índice
    - btree, hash
      - btree é o default
  - Predicado
    - Predicado para selecionar que tuplas devem ser indexadas
  - NULLS { FIRST | LAST}
    - Especifica ordem de nulls (ordenação)
  - Parâmetro de armazenamento
    - FILLFACTOR
      - Qualquer inteiro no intervalo [10, 100]
        - Default é 90

10. Projeto Físico de Banco de Dados
- Indexação -

- Exercícios (SQL Server)
  - Crie um índice primário na tabela Vendedores com chave de busca o atributo codfil. A taxa de preenchimento deve ser de 100%, para todos os nós
  - Crie um índice secundário na tabela Vendedores com chave de busca o atributo CPF. A taxa de preenchimento deve ser de 100%, só para os nós folhas

HD Drive
- Information Encoding -

- Head
  - One head for each surface, riding close to the surface
    - Never touches the surface
  - Reads and alters magnetism
    - R/W operation