Resumo para a Prova de Banco de Dados Relacionais (TTI102)

1. Modelo Relacional e Álgebra Relacional (Slides 07 e 08)

- **Modelo Relacional:** Dados organizados em tabelas (relações) com linhas (tuplas) e colunas (atributos). Operações baseadas na Álgebra Relacional.
- **Álgebra Relacional:** Conjunto de operações para manipular dados:
  - **Seleção (σ):** Filtra tuplas com base em uma condição. Ex.: σ idade>40 (Empregado) seleciona empregados com mais de 40 anos.
    - Propriedade: Comutativa (σ cond1 (σ cond2 (R)) = σ cond2 (σ cond1 (R))); seletividade ≤ número de tuplas de R.
  - **Projeção (π):** Seleciona colunas específicas, eliminando duplicatas. Ex.: π nome,salario (Empregado) retorna apenas nome e salario.
    - Propriedade: Não comutativa (π lista1 (π lista2 (R)) ≠ π lista2 (π lista1 (R))); elimina duplicatas para manter o modelo relacional.
  - **Renomear (ρ):** Renomeia relações ou atributos. Ex.: ρ Resultado (R) ou ρ (novoNome) (atributo).
- **Operações de Conjunto:**
  - **União (∪):** Combina tuplas de R e S, eliminando duplicatas.
  - **Intersecção (∩):** Tuplas comuns a R e S.
  - **Diferença (-):** Tuplas em R que não estão em S.
  - **Produto Cartesiano (×):** Combina cada tupla de R com cada tupla de S. Grau = n+m; número de tuplas = |R| × |S|.
- **SQL Equivalente:**
  - Seleção → WHERE (ex.: WHERE idade > 40).
  - Projeção → SELECT (ex.: SELECT nome, salario FROM Empregado).
  - União → UNION, Intersecção → INTERSECT, Diferença → EXCEPT.

---

2. Produto Cartesiano e Junções (Slides 09 e 10)

- **Produto Cartesiano (×):**
  - Combina todas as tuplas de duas relações (R × S). Ex.: R com n tuplas e S com m tuplas → n × m tuplas.
  - Não tem significado isolado, mas é útil com seleção posterior.
- **Junção (⋈):**
  - Combina tuplas relacionadas de duas relações com base em uma condição.
  - **Tipos de Junção:**
    - **Theta Join:** Condição geral (ex.: A θ B, onde θ é =, <, >, etc.).
    - **Equijunção (Equijoin):** Usa igualdade (ex.: R ⋈ A=B S).
    - **Natural Join:** Equijunção com atributos de mesmo nome, eliminando redundâncias.
  - **Características:**
    - Resultado tem grau n+m; só inclui tuplas que satisfazem a condição.

        - Tuplas com valores nulos ou condição falsa não aparecem.
- **SQL Equivalente:**
    - Produto Cartesiano → CROSS JOIN ou FROM R, S sem WHERE.
    - Inner Join → INNER JOIN ... ON ... (ex.: FROM Produtos INNER JOIN Categorias ON idCategoria = idCategoria).
    - Usar USING para atributos comuns (ex.: INNER JOIN Categorias USING(idCategoria)).
- **Outer Joins (Extensão SQL):**
    - **Left Join (⟕):** Tudo da tabela à esquerda + correspondências da direita (nulos se não houver).
    - **Right Join (⟖):** Tudo da tabela à direita + correspondências da esquerda.
    - **Full Outer Join (⟗):** Tudo de ambas, com nulos onde não há correspondência.
- **Self-Join:** Junção de uma tabela consigo mesma (ex.: comparar filmes com mesmo tamanho: SELECT t1.title, t2.title FROM film t1 INNER JOIN film t2 ON t1.length = t2.length).

---

3. Generalização, Especialização e Notação (Slide 06)

- **Generalização:** Abordagem "de baixo para cima". Identifica atributos comuns entre subtipos e cria uma entidade superior. Ex.: Veículo (placa, marca) para Veículos de Passageiros e de Carga.
- **Especialização:** Abordagem "de cima para baixo". Parte de uma entidade geral (ex.: Cliente) e cria subtipos (ClientePF, ClientePJ) com atributos específicos.
- **Notação Pé de Galinha:** Representa cardinalidade (máximo) e ordinalidade (mínimo) em diagramas E-R:
    - Linhas indicam limites (ex.: 0..1, 1..N).
- **Relacionamentos Identificadores:** Usam a chave primária da entidade independente como parte da chave primária da dependente (ex.: LivroAutor com id_livro e id_autor como PK).
- **Relacionamentos Não Identificadores:** Têm chave primária própria (ex.: LivroAutor com id_livro_autor como PK).

---

4. Projeto de Banco de Dados e Normalização (Slide 11)

- **Projeto de Banco de Dados:**
    - **Conceitual:** Usa Modelo Entidade-Relacionamento (MER).
    - **Lógico:** Mapeia o MER para o modelo relacional.
- **Boas Práticas:**
    - **Diretriz 1:** Semântica clara (não misturar entidades distintas). Ex.: Não combinar Funcionário e Departamento em uma única tabela.
    - **Diretriz 2:** Evitar anomalias (inserção, exclusão, modificação) causadas por redundância.

    - **Diretriz 3:** Minimizar valores NULL (podem gerar lógica de 3 valores: TRUE, FALSE, UNKNOWN).
- **Impactos de Má Modelagem:**
    - Redundância (ex.: repetir NomeDepto em Funcionário).
    - Anomalias:
        - **Inserção:** Novo funcionário sem departamento exige NULL ou dados inconsistentes.
        - **Exclusão:** Remover último funcionário perde dados do departamento.
        - **Modificação:** Alterar NomeDepto exige atualizar várias tuplas.
- **Dependências Funcionais (DF):**
    - X → Y: X determina Y. Ex.: CPF → Fnome (CPF determina nome).
    - **Total:** Remover qualquer atributo de X quebra a DF (ex.: {CPF, ProjNumero} → Horas).
    - **Parcial:** Pode remover algo de X e DF ainda vale (ex.: {CPF, ProjNumero} → Fnome).
    - **Transitiva:** X → Z e Z → Y, mas Z não é chave (ex.: CPF → Dnumero → CPF_Gerente).
- **Normalização:**
    - **1FN:** Atributos atômicos (sem listas). Ex.: Separar Dlocal (SP, RJ) em tuplas distintas.
    - **2FN:** Atributos não principais devem depender totalmente da chave primária. Ex.: FUNC_PROJ viola 2FN com CPF, ProjNumero → Fnome (parcial).
    - **3FN:** Atributos não principais não podem ter dependências transitivas. Ex.: FUNC_DEP viola 3FN com CPF → Dnumero → CPF_Gerente.

---

5. Transações e Controle de Concorrência (Slide 12)

- **Transações:** Conjunto de operações que devem ser "tudo ou nada" (propriedades ACID: Atomicidade, Consistência, Isolamento, Durabilidade).
- **Controle de Concorrência:**
    - **Bloqueios Binários:** Lock(X) = 1 (bloqueado) ou 0 (desbloqueado). Operações: lock_item(X), unlock_item(X).
    - **Bloqueios Múltiplos:** Leitura (read_lock) e Gravação (write_lock). Permite várias leituras simultâneas, mas gravação é exclusiva.
- **Two-Phase Locking (2PL):**
    - Fase 1 (Crescimento): Só adquire bloqueios.
    - Fase 2 (Encolhimento): Só libera bloqueios.
    - Garante serializabilidade (equivalência a uma execução serial).
- **Problemas com Bloqueios:**
    - **Deadlock:** Transações esperando umas pelas outras (ex.: T1 bloqueia X, T2 bloqueia Y, T1 espera Y, T2 espera X).
    - **Starvation:** Transação não prossegue indefinidamente (solução: FIFO ou prioridade crescente).
- **Recuperação:**
    - **Write-Ahead Logging:** Registra operações no log antes de gravar no disco.
    - **Checkpoint:** Marca transações confirmadas no log.

- Ex.: Após falha, refazer transações confirmadas (T2, T3) e desfazer não confirmadas (T4, T5).

---

6. Exemplos Práticos

- **Seleção e Projeção:** π nome,salario (σ idade>40 (Empregado)) → Em SQL: SELECT DISTINCT nome, salario FROM Empregado WHERE idade > 40;.
- **Junção:** Listar gerentes de departamentos: DEPARTAMENTO ⋈ cpf_gerente=cpf FUNCIONARIO → Em SQL: SELECT nome_departamento, nome FROM Departamento INNER JOIN Funcionario ON cpf_gerente = cpf;.
- **Normalização:** FUNC_PROJ (CPF, ProjNumero, Horas, Fnome, ProjNome, ProjLocal) → Decomposto em 3 tabelas: FP_1 (CPF, ProjNumero, Horas), FP_2 (CPF, Fnome), FP_3 (ProjNumero, ProjNome, ProjLocal).