**Briefing Document: Bancos de Dados, SQL e Modelagem de Dados**

Este documento apresenta um resumo das principais informações e conceitos extraídos das aulas fornecidas, focando em Bancos de Dados, Linguagem SQL (DDL, DML, DQL) e Modelagem de Dados (MER).

**1. Carreira e Fundamentos de Bancos de Dados**

* **Administrador de Banco de Dados (DBA):** O DBA é o profissional responsável pela "administração, controle, segurança, gerenciamento, recuperação e backup de todos os bancos de dados de uma instituição." Este é um "cargo de confiança em todas as empresas", dado o acesso a informações críticas. Existem especializações para DBAs, como:
* **Desenvolvedor:** "especialista em conceitos de melhorias de consultas e execuções (tunning)", com expertise em acompanhamento e sugestões de desenvolvimento no banco de dados.
* **Infraestrutura:** "especialista no desenho de soluções envolvendo hardware e armazenamento (storage)", com afinidade em projetos de alta disponibilidade e grandes cargas de processamento.
* **Linguagem SQL:** É a linguagem padrão que os Sistemas de Banco de Dados Relacionais têm em comum. Desenvolvida nos anos 70 pela IBM, é dividida em várias categorias:
* **DDL (Data Definition Language):** Linguagem de Definição de Dados. Comandos incluem CREATE (DATABASE, TABLE, VIEW, USER), DROP, ALTER, TRUNCATE, COMMENT, RENAME.
* **DML (Data Manipulation Language):** Linguagem de Manipulação de Dados. Comandos incluem INSERT, DELETE, UPDATE, CALL, EXPLAINPLAN, LOCK TABLE.
* **DQL (Data Query Language):** Linguagem de Consulta de Dados. O principal comando é SELECT.
* **DCL (Data Control Language):** Linguagem de Controle de Dados. Comandos incluem GRANT, REVOKE, DENY.
* **DTL (Data Transaction Language):** Linguagem de Transação de Dados. Comandos incluem COMMIT, SAVEPOINT, ROLLBACK.

**2. Modelagem de Dados: Modelo Entidade-Relacionamento (MER)**

* **MER:** "É um modelo de dados de alto-nível criado com o objetivo de representar a semântica associada aos dados do minimundo." Utilizado na fase de projeto conceitual, seus conceitos são intuitivos e independentes de tecnologia.
* **Diagrama Entidade-Relacionamento (DER):** O esquema conceitual criado utilizando o MER.
* **Entidade:** "O objeto mais elementar que o MER representa", é "algo do mundo real que possui uma existência independente." Exemplos incluem objetos, pessoas, empregados, conceitos.
* **Atributos:** Propriedades particulares de uma entidade. Podem ser:
* **Simples:** Não podem ser divididos (ex: nome\_marca).
* **Compostos:** Podem ser divididos em subpartes com significados independentes.
* **Multivalorados:** Podem ter um conjunto de valores (ex: múltiplos telefones). Representados com borda dupla.
* **Derivados:** Valores obtidos através de processamento de outras informações no BD (ex: idade = Data\_Atual - Data\_Nascimento). Representados com borda tracejada.
* **Nulos (Null):** Atribuído quando um atributo não possui valor ou é desconhecido.
* **Atributo-Chave:** "Uma restrição importante sobre entidades de um tipo de entidade". Todo tipo de entidade deve ter um atributo-chave (simples ou composto) cujos valores devem ser distintos.
* **Chave Fraca:** "Um tipo de entidade-fraca tem uma chave-parcial, que é um conjunto de atributos que pode univocamente identificar entidades-fracas relacionadas à mesma entidade proprietária."
* **Cardinalidade:** "Expressa o número de entidades relacionadas a outras entidades por meio de um conjunto de relacionamentos." Tipos incluem:
* **Um-para-um (1:1):** Uma entidade A associada a apenas uma entidade B.
* **Um-para-muitos (1:N):** Uma entidade A associada a muitas entidades B, mas uma entidade B associada a apenas uma A.
* **Muitos-para-um (N:1):** Uma entidade A associada a apenas uma entidade B, mas uma entidade B associada a muitas A.
* **Muitos-para-muitos (M:N):** Uma entidade A pode estar associada a muitas B, e uma entidade B pode estar associada a muitas A.
* **Boas Práticas de DER:**"Não crie um Tipo de Entidade sem Atributos."
* "Não deixe de colocar a Cardinalidade nos relacionamentos."

**3. Mapeamento de DER para Modelo Relacional**

O processo de mapeamento de um Diagrama Entidade-Relacionamento (DER) para um Modelo Relacional (MR) envolve várias etapas para transformar entidades e relacionamentos em tabelas e chaves:

* **Passo 1 (Tipos de Entidade Normal):** Para cada tipo de entidade normal E, crie uma relação R com todos os atributos simples de E (incluindo atributos de atributos compostos). Escolha um dos atributos-chave de E como a chave primária de R.
* **Passo 2 (Tipos de Entidade Fraca):** Para cada tipo de entidade fraca W com um relacionamento de identificação E, crie uma relação R. Inclua todos os atributos simples de W e a chave primária da relação correspondente ao tipo de entidade proprietária como chave estrangeira de R. A chave primária de R é a combinação da chave primária da entidade proprietária e a chave parcial da entidade fraca.
* **Passo 3 (Cardinalidade 1:1):** Para cada relacionamento binário 1:1 R, identifique as relações S e T. Escolha uma das relações (S, preferencialmente a com participação total) e inclua a chave primária de T como chave estrangeira em S. Inclua também quaisquer atributos simples do relacionamento R em S.
* **Passo 4 (Cardinalidade 1:N):** Para cada relacionamento binário regular 1:N R, identifique a relação S (lado N) e T (lado 1). Inclua a chave primária de T como chave estrangeira em S.
* **Passo 5 (Cardinalidade M:N):** Para cada relacionamento binário M:N R, crie uma **nova relação S** para representá-lo. Inclua as chaves primárias das relações participantes como chaves estrangeiras em S, e a combinação dessas chaves estrangeiras formará a chave primária de S. Isso é necessário porque o MR não permite atributos multivalorados diretamente.
* **Passo 6 (Atributos Multivalorados):** Para cada atributo multivalorado A, crie uma nova relação R que inclua A e a chave primária K da relação que representa a entidade ou relacionamento que possui A.

**4. SQL: Criação e Manipulação de Tabelas (DDL, DML, DQL)**

* **Tipos de Dados:** Bancos de dados associam tipos de dados a colunas, expressões, variáveis e parâmetros, determinando os valores permitidos. Dados são armazenados em bytes e traduzidos para exibição. Categorias principais incluem:
* **Numéricos exatos:** Tinyint, Smallint, MediumInt, Int/Integer, Decimal.
* **Numéricos aproximados:** Float, Double.
* **Data e hora:** DATE, TIME, DATETIME, TIMESTAMP.
* **Cadeias de caracteres (Strings):** Tinytext, Text, MediumText, Longtext, Char, VarChar.
* **Caractere Unicode:** Para suportar uma gama maior de caracteres.
* **Binários:** Para dados como imagens, vídeos.
* **Comentários em SQL:** Podem ser -- Comentário, # Comentário ou /\* Múltiplas linhas de Comentário \*/.
* **CREATE DATABASE:** Cria um novo banco de dados. Ex: CREATE DATABASE IF NOT EXISTS NOME\_BANCO;.
* **USE:** Abre um banco de dados para uso. Ex: USE NOME\_BANCO;.
* **DROP DATABASE:** Exclui um banco de dados. Ex: DROP DATABASE NOME\_BANCO;.
* **CREATE TABLE:** Cria uma nova tabela. Sintaxe básica:
* CREATE TABLE IF NOT EXISTS NOME\_TABELA (
* "coluna 1" "tipo\_dados\_para\_coluna\_1",
* "coluna 2" "tipo\_dados\_para\_coluna\_2",
* ...
* );
* **DROP TABLE:** Exclui uma tabela. Ex: DROP TABLE EXISTS GENERO;.
* **SHOW DATABASES;** e **SHOW TABLES;**: Listam bancos de dados e tabelas, respectivamente.
* **DESCRIBE / DESC:** Mostra a estrutura de uma tabela. Ex: DESC nome\_tabela;.
* **Restrições (Constraints):** Usadas para limitar os dados em uma tabela, garantindo a integridade dos dados:
* **NOT NULL:** Garante que uma coluna não pode ter valor NULL.
* **DEFAULT:** Fornece um valor padrão quando nenhum é especificado.
* **UNIQUE:** Garante que todos os valores em uma coluna são diferentes.
* **CHECK:** Garante que todos os valores em uma coluna satisfazem um critério.
* **PRIMARY KEY:** Identifica unicamente uma linha na tabela. (Ex: pk\_midia int auto\_increment primary key).
* **FOREIGN KEY:** Garante a integridade referencial dos dados, ligando tabelas. (Ex: CONSTRAINT FK\_EST FOREIGN KEY (SIGLA\_EST) REFERENCES ESTADO (SIGLA\_EST)). Ações em FOREIGN KEY incluem ON DELETE / ON UPDATE: RESTRICT, CASCADE, NO ACTION, SET NULL, SET DEFAULT.
* **TRUNCATE TABLE:** Remove todos os registros de uma tabela, redefinindo o auto incremento. Ex: TRUNCATE frutas;.

**5. Consultas SQL (DQL) - SELECT e Funções**

* **SELECT:** Permite pesquisar dados em uma ou mais tabelas.
* Pode retornar todas as colunas (SELECT \*) ou colunas específicas.
* Pode usar expressões aritméticas e apelidos (ALIAS) para colunas (Ex: SELECT nome\_autor AS 'nome do autor').
* **Ordem de Escrita da Cláusula SELECT:** SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY.
* **Ordem de Execução da Cláusula SELECT:** FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY.
* **DISTINCT:** Usado para eliminar linhas duplicadas nos resultados da consulta. (Ex: SELECT DISTINCT nome\_autor).
* **Boa Prática no SELECT:** "Sempre explicitar os atributos que são necessários", evitando SELECT \* para melhor desempenho e clareza.
* **Cláusula FROM:** Especifica as tabelas envolvidas na consulta. É uma boa prática referenciar o nome do esquema (Ex: SELECT \* FROM Livraria.livro;). Tabelas podem ser renomeadas/apelidadas (Ex: FROM LIVRARIA.LIVRO AS LIV;).
* **Cláusula WHERE:** Filtra as linhas (tuplas) desejadas usando operadores de comparação (>, >=, <, <=, <>, =), BETWEEN (intervalo), IN e NOT IN (listas), LIKE e NOT LIKE (padrões com %), IS NULL e IS NOT NULL.
* **Funções de Arredondamento:**ROUND(valor, casas\_decimais): Arredonda para o número de casas decimais especificado.
* TRUNCATE(valor, casas\_decimais): Trunca um número para o número de casas decimais.
* FLOOR(valor): Retorna o maior inteiro menor ou igual ao valor.
* CEILING(valor): Retorna o menor inteiro maior ou igual ao valor.
* **Cálculo de Idade:** Exemplo: SELECT FLOOR(DATEDIFF(NOW(), p.nascimento) / 365) AS idade FROM pessoa p.
* **Funções de String:LEFT(Texto/campo, número de caracteres):** Extrai caracteres do início de uma string.
* **RIGHT(Texto/campo, número de caracteres):** Extrai caracteres do final de uma string.
* **TRIM(Texto/campo):** Remove espaços iniciais e finais de um texto/campo. Pode remover caracteres específicos (Ex: TRIM(')' FROM RIGHT(rubrica,9))).
* **POSITION(substr IN texto):** Retorna a posição de uma substring dentro de uma string.
* **SUBSTRING(string, start, length):** Extrai uma parte específica de uma string.

**6. Subconsultas (Subqueries/Sub Selects)**

* **Definição:** Uma subquery é um comando SELECT que é "embutido" dentro de outro comando SELECT, UPDATE, DELETE ou dentro de outra subquery. Sua finalidade é "retornar um conjunto de linhas para a query principal."
* **Uso Comum:** Tipicamente utilizadas na filtragem de pesquisas (WHERE) nas cláusulas IN() e EXISTS(), mas também podem aparecer na cláusula FROM ou como substituto de expressões.
* **Onde Podem Ser Usadas:** INSERT, UPDATE, DELETE, SELECT.
* **Tipos de Subconsultas:De uma única linha:** "retornam zero ou uma linha para a instrução SQL externa." Podem usar operadores de comparação como =, >, etc. Um erro comum é quando uma subconsulta de única linha retorna mais de uma linha.
* **De várias linhas:** "retornam uma ou mais linhas para uma instrução SQL externa." Para tratá-las, a consulta externa pode usar os operadores IN, NOT IN, ANY, ou ALL.
* IN: Verifica se um valor está contido em uma lista de valores retornada pela subconsulta.
* NOT IN: Verifica se um valor **não** está contido na lista.
* ANY: Compara um valor com *qualquer* valor presente em uma lista (deve ser precedido por =, <>, <, >, <=, ou >=).
* ALL: Compara um valor com *todos* os valores presentes em uma lista.
* **De várias colunas:** Retornam mais de uma coluna para a instrução SQL externa. Exemplo: recuperar produtos com o menor preço por categoria.
* **Correlacionadas:** "fazem referência a uma ou mais colunas na instrução SQL externa." São relacionadas à instrução SQL externa pelas mesmas colunas. Usadas quando a resposta depende de um valor em cada linha da consulta externa, ou para verificar a existência de linhas sem se preocupar com a quantidade. Podem trabalhar com valores nulos.
* **Aninhadas:** Subconsultas dentro de outras subconsultas, até 255 níveis (muito raro). Essa técnica deve ser usada com moderação, pois junções de tabela podem ter melhor desempenho.
* **Operadores EXISTS e NOT EXISTS:**EXISTS: Verifica a existência de linhas retornadas por uma subconsulta. Geralmente usado em subconsultas correlacionadas. "Não nos preocupamos com a quantidade de linhas retornadas pela subconsulta, só nos preocupamos em saber se alguma linha é retornada."
* NOT EXISTS: Executa a lógica oposta de EXISTS, verificando se linhas **não** existem. Usado para localizar registros não correspondentes.