Conceitos Importantes Mega Dados

1. Relacionamentos Identificadores e Não-Identificadores

- **Identificadores**: Quando a chave estrangeira é também parte da chave primária da tabela filha. Isso estabelece uma dependência direta entre as entidades, significando que a entidade filha não pode existir sem a entidade pai.
    - **Exemplo**: Em um sistema escolar, uma tabela **Matricula** que associa alunos a cursos pode ter como chave primária composta o **idAluno** e **idCurso**, indicando que cada matrícula é única para cada par aluno-curso.
- **Não Identificadores**: A chave estrangeira não faz parte da chave primária, permitindo que a entidade filha exista independentemente da entidade pai.
    - **Exemplo**: Um aluno (**Aluno**) pode ter várias inscrições (**Inscricao**), mas cada inscrição é identificada por um **idInscricao** único, independente do **idAluno**.

2. Chaves Primárias Compostas

- Usadas quando mais de uma coluna é necessária para identificar unicamente cada linha na tabela.
    - **Exemplo**: Na tabela **Participacao**, a combinação de **idPartida** e **idAluno** serve como chave primária composta, garantindo que um aluno não participe mais de uma vez na mesma partida.

3. Índices

- Estruturas que melhoram a velocidade de busca dos dados. Eles são especialmente úteis em colunas frequentemente usadas em cláusulas WHERE, JOIN ou ORDER BY.
    - **Exemplo**: Criar um índice na coluna **email** da tabela **Usuarios** pode acelerar consultas que buscam usuários por seu e-mail.

4. INNER JOIN

- Combina linhas de duas tabelas onde existe uma correspondência em ambas. Usado quando é essencial que existam dados correspondentes nas tabelas que estão sendo unidas.
    - **Exemplo**: **SELECT * FROM Funcionarios INNER JOIN Departamentos ON Funcionarios.DepartamentoID = Departamentos.ID** retorna apenas funcionários que têm um departamento associado.

5. LEFT JOIN (ou LEFT OUTER JOIN)

- Retorna todas as linhas da tabela à esquerda, junto com as linhas correspondentes da tabela à direita. Quando não há correspondência, retorna NULL para as colunas da tabela à direita.
    - **Exemplo**: **SELECT * FROM Funcionarios LEFT JOIN Departamentos ON Funcionarios.DepartamentoID = Departamentos.ID** retorna todos os funcionários, incluindo aqueles sem um departamento associado.

6. WHERE vs. HAVING

- **WHERE** é usado para filtrar linhas antes de qualquer agrupamento. **HAVING** é usado para filtrar grupos criados pela cláusula **GROUP BY**.
    - **Exemplo WHERE**: **SELECT * FROM Funcionarios WHERE salario > 5000** retorna funcionários com salário maior que 5000.
    - **Exemplo HAVING**: **SELECT DepartamentoID, AVG(salario) FROM Funcionarios GROUP BY DepartamentoID HAVING AVG(salario) > 5000** retorna departamentos com um salário médio superior a 5000.

7. UNION

- Combina os resultados de duas ou mais consultas em um único conjunto de resultados, excluindo duplicatas.
    - **Exemplo**: **SELECT nome FROM Funcionarios UNION SELECT nome FROM Clientes** retorna os nomes únicos tanto de funcionários quanto de clientes.

8. GROUP BY

- Agrupa linhas que têm os mesmos valores em colunas especificadas, geralmente usado com funções agregadas (COUNT, MAX, SUM, AVG).
    - **Exemplo**: **SELECT DepartamentoID, COUNT(*) FROM Funcionarios GROUP BY DepartamentoID** conta o número de funcionários em cada departamento.

9. ORDER BY

- Ordena os resultados de uma consulta em ordem ascendente ou descendente.
    - **Exemplo**: **SELECT * FROM Funcionarios ORDER BY Nome ASC** ordena os funcionários pelo nome em ordem ascendente.

10. LIMIT

- Limita o número de linhas retornadas por uma consulta.
    - **Exemplo**: **SELECT * FROM Funcionarios LIMIT 10** retorna os primeiros 10 funcionários.

11. COALESCE

- Retorna o primeiro valor não nulo na lista de argumentos.
    - **Exemplo**: **SELECT COALESCE(Telefone, 'Não Informado') FROM Funcionarios** retorna 'Não Informado' se o campo Telefone for NULL.

12. SHA2

- Calcula o hash SHA-256 de uma string.
    - **Exemplo**: **SELECT SHA2('senha123', 256)** retorna o hash SHA-256 da string 'senha123'.

13. Anonimização e Mascaramento de Dados

- Técnicas para proteger informações sensíveis, mantendo a privacidade e conformidade.
    - **Exemplo**: Usar **SHA2** para anonimizar CPFs antes de armazená-los.

14. Importância de Modelar Cuidadosamente as Relações

- Reflete a necessidade de representar corretamente as regras de negócio e garantir a integridade dos dados.
    - **Exemplo**: A decisão entre usar **INNER JOIN** ou **LEFT JOIN** baseia-se na relação entre as entidades e o que precisa ser retornado pela consulta.

15. Considerações sobre Performance

- Refere-se à importância de otimizar consultas e estruturas, como índices, para melhor desempenho.
    - **Exemplo**: A adição de índices em colunas frequentemente usadas em buscas pode melhorar significativamente a velocidade das consultas, mas também requer cuidadosa consideração do impacto na inserção e atualização de dados.