Campus polo Rua Teresa - Desenvolvimento Full Stack

RPG0015 - Vamos manter as informações?
Turma 9001 - 3o semestre
Aluno: Marcos Valerio R S de Carvalho
Github: https://github.com/Voidrrrr/Mundo-3-nivel-2

RPG 0015 - Vamos manter as informações?

Objetivos da prática:

- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
- No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de

implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

Análise e Conclusão:

Quais as diferenças no uso de sequence e identity?

- Em SQL, tanto SEQUENCE quanto IDENTITY geram números automaticamente, mas diferem em uso e flexibilidade:
    - SEQUENCE é um objeto independente, utilizado para gerar números sequenciais que podem ser aplicados em várias tabelas e colunas. Oferece mais controle e opções de personalização, como definir incrementos, reiniciar a sequência, e valores máximos/mínimos.

    - IDENTITY é uma propriedade de coluna, usada para gerar números automaticamente em uma coluna específica de uma tabela. É mais simples e menos flexível, com opções limitadas de personalização (início e incremento). Diferenças principais: SEQUENCE: Mais flexível e reutilizável. IDENTITY: Simples e vinculado a uma coluna de tabela.

Qual a importância das chaves estrangeiras para a consistência do banco?

- As *chaves estrangeiras* são essenciais para a consistência de um banco de dados, pois garantem que os relacionamentos entre tabelas sejam válidos. Elas asseguram que o valor em uma coluna de uma tabela (chave estrangeira) corresponda a um valor existente em outra tabela (chave primária). Isso previne dados órfãos ou inconsistentes, como registros sem correspondência. Além disso, ajudam a manter a integridade referencial, ou seja, que os dados relacionados entre diferentes tabelas permaneçam coerentes e corretos ao longo do tempo.

Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

- No SQL, os operadores se baseiam em dois formalismos:

  Álgebra Relacional: Define operações sobre conjuntos de dados, como:

  - **Seleção** (`SELECT` com `WHERE`): Filtra linhas de uma tabela.
  - **Projeção** (`SELECT`): Seleciona colunas específicas.
  - **União** (`UNION`): Combina resultados de duas consultas.
  - **Diferença** (`EXCEPT` ou `MINUS`): Retorna registros que estão em uma tabela, mas não na outra.
  - **Produto Cartesiano** (`CROSS JOIN`): Combina todas as linhas de duas tabelas.
  - **Junção** (`INNER JOIN`, `LEFT JOIN`, etc.): Combina tabelas com base em uma condição.

  Cálculo Relacional: Define consultas declarativas que especificam o que se quer como resultado, sem determinar o caminho para obtê-lo. Em SQL, isso se reflete nas consultas utilizando `SELECT`, que indicam as condições e projeções esperadas.

Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

- O agrupamento em SQL é feito com a cláusula GROUP BY, que agrupa linhas com valores iguais em uma ou mais colunas. Após o agrupamento, é possível usar funções agregadas como SUM(), COUNT(), AVG(), MAX() e MIN() para realizar cálculos sobre cada grupo. Requisito obrigatório: Se você usa GROUP BY, todas as colunas selecionadas fora das funções agregadas devem estar presentes no GROUP BY, pois os dados estão sendo agrupados com base nessas colunas.