**EXPRESS -** é limitada tem menos recursos. É adequada para pequenas aplicações e projetos individuais. Embora seja mais leve e tenha menos recursos que as edições completas, ela pode ser usada em **produção** se as necessidades de armazenamento e processamento de dados forem limitadas. No entanto, quando o banco de dados cresce muito, a performance pode ser afetada e o gerenciamento de registros pode se tornar mais difícil. O SQL Server Express tem um limite de tamanho de banco de dados de 10 GB.

**DEVELOPER** - é a versão completa e gratuita, é para desenvolvimento e não para produção. -> é voltada para desenvolvedores e equipes de desenvolvimento que precisam de todas as funcionalidades do SQL Server, mas não precisam de suporte de produção. Essa edição é licenciada por usuário e permite que os desenvolvedores criem e testem aplicativos em um ambiente semelhante ao de produção, sem ter que comprar uma licença completa do SQL Server.

**SSMS -** é a interface gráfica de usuário (GUI) usada para interagir com o SQL Server. Com o SSMS, é possível executar comandos SQL, criar e gerenciar bancos de dados, tabelas e outros objetos de banco de dados, além de monitorar a performance do servidor.

Pelo SQL server ser da microsoft é mais comum que quem trabalhe com C# utilize o SQL server para manter um "padrão” de produtos

**Replicação no SQL Server:** é um recurso que permite que dados sejam copiados e distribuídos automaticamente de um banco de dados para outro, ou para vários bancos de dados, em um ambiente de banco de dados distribuído.

A replicação é útil em cenários em que é necessário manter cópias atualizadas de dados em vários locais. Por exemplo, se uma organização possui filiais em diferentes locais geográficos, cada filial pode ter seu próprio banco de dados local. A replicação permite que as atualizações feitas em um banco de dados em uma filial sejam automaticamente refletidas nos bancos de dados de outras filiais ou em um banco de dados central.

Existem vários tipos de replicação no SQL Server, como a replicação transacional, que replica cada transação realizada em um banco de dados fonte para um ou mais bancos de dados de destino; e a replicação de mesclagem, que permite que as atualizações sejam feitas em vários bancos de dados que contêm cópias dos mesmos dados e, em seguida, as atualizações são mescladas em um único banco de dados de destino.

**Após instalação do SQL:**

New query

Temos dois tipos de comandos no banco de dados:

**DDL (Data Definition Language)** 🡪 Dados que criam algo para receber alguma informação

são comandos SQL usados para criar, alterar ou excluir objetos de banco de dados, como tabelas, índices, procedimentos armazenados, funções, entre outros. Esses comandos não manipulam os dados em si, mas sim a estrutura do banco de dados.

CREATE

ALTER (altera a estrutura)

DROP (excluir indefinitivo)

**DML** (Data Manipulation Language) 🡪 manipulação de dados

são comandos SQL usados para manipular os dados em uma tabela, como inserir, atualizar ou excluir registros.

INSERT (insere dados numa tabela já criada / definida)

UPDATE

DELETE

**C** reate

**R** ead

**U** pdate 🡪 para usar as operações que envolvem o CRUD precisa ter os dados já definidos nos comandos DDL

**D** elete

CRUD: é um acrônimo para Create, Read, Update e Delete, que são as quatro operações básicas de um sistema de gerenciamento de banco de dados. O objetivo dessas operações é criar, ler, atualizar e excluir dados em uma tabela de banco de dados.

**Primary Key (chave primaria)** - é o **atributo** que não se repete, é exclusivo. identifica unicamente cada registro em uma tabela de banco de dados. Essa chave é usada para garantir a integridade dos dados e evitar a duplicação de registros. Quando não tem algo na tabela (entidade) que possa ser único, criamos o ID.

\* nem sempre o CPF pode ser usado como PK pois as vezes estamos tratando de crianças q ainda não possuem CPF

**Foreign Key (chave estrangeira)** – é uma chave que estabelece uma relação entre tabelas em um banco de dados. Ela é um atributo que referência a chave primária de outra tabela e é usada para garantir a integridade referencial entre as tabelas. Quando uma chave estrangeira é definida em uma tabela, ela cria uma relação entre duas tabelas, que pode ser uma relação de um-para-muitos, muitos-para-muitos ou de um-para-um.

* Um-para-muitos (1:N): é o tipo de relação mais comum entre duas tabelas em um banco de dados relacional. Nesse tipo de relação, um registro da tabela A pode estar relacionado a vários registros da tabela B, mas um registro da tabela B só pode estar relacionado a um registro da tabela A. Por exemplo, em um banco de dados de uma loja, cada cliente pode ter várias compras associadas, mas cada compra pertence a apenas um cliente.
* Muitos-para-muitos (N:N): nesse tipo de relação, vários registros de uma tabela podem estar relacionados a vários registros de outra tabela. Esse tipo de relação requer a criação de uma tabela intermediária (ou tabela associativa) para armazenar as chaves primárias das duas tabelas relacionadas. Por exemplo, em um banco de dados de uma biblioteca, vários livros podem ser emprestados por vários clientes. Nesse caso, é necessário criar uma tabela intermediária para armazenar as chaves primárias das tabelas de livros e clientes.
* Um-para-um (1:1): nesse tipo de relação, um registro da tabela A está relacionado a apenas um registro da tabela B e vice-versa. Esse tipo de relação é menos comum em bancos de dados relacionais, mas pode ser útil em casos específicos, como quando queremos separar informações confidenciais em uma tabela separada. Por exemplo, em um banco de dados de uma empresa, podemos ter uma tabela de funcionários com informações públicas e uma tabela separada para armazenar informações confidenciais, como salário e benefícios, que só podem ser acessadas por gerentes e funcionários autorizados.

**PLANO DE NEGÓCIOS**

Estrutura de banco de dados relacional, projetado usando um Modelo Entidade-Relacionamento (MER).

O MER é uma ferramenta de modelagem de dados que permite representar as entidades (como "alunos", "matrículas", "cursos") e seus relacionamentos de forma gráfica e intuitiva. É uma técnica muito utilizada no desenvolvimento de software para definir a estrutura do banco de dados a ser utilizado.

Para construir um MER, o primeiro passo é identificar as entidades principais do sistema que se deseja modelar. Em seguida, é necessário estabelecer as relações entre essas entidades e definir os atributos que cada uma delas terá. Por exemplo, na sua descrição, a tabela "alunos" é uma entidade com os atributos "nome", "cpf", "data de nascimento" e "ID". A tabela "matrículas" é outra entidade que se relaciona com a tabela "alunos" por meio da chave estrangeira "fk aluno" e também com a tabela "cursos" por meio da chave estrangeira "fk curso".

Com base nessas informações, seria possível construir um MER que representasse essas entidades e relacionamentos. É importante destacar que a construção de um MER bem feito requer uma boa compreensão do negócio em questão e um planejamento cuidadoso da estrutura do banco de dados.

Uma vez construído o MER, é possível utilizá-lo como base para criar o banco de dados físico, que será responsável por armazenar os dados de forma concreta. O SQL Server, por exemplo, oferece ferramentas para a criação de bancos de dados a partir de modelos pré-existentes, incluindo aqueles baseados em MER.

Diagrama

Descrição gerada automaticamente



Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Você pode pensar na entidade (Faculdade) como um conceito geral que engloba todas as tabelas relacionadas a ela. Cada tabela representa uma entidade específica, como alunos, curso e matriculo, e cada tabela é composta por colunas (também chamadas de campos ou atributos), que representam os diferentes tipos de dados que serão armazenados (exemplo: ID, Nome, CPF na tabela alunos)

Os registros são as linhas dentro de cada tabela, que contêm os dados reais que foram inseridos para cada registro individual. Por exemplo, se você tem uma coluna chamada "Nome" em sua tabela "Alunos", então cada linha na tabela representará um aluno diferente, com seu próprio título inserido como um valor na coluna "Nome", como por exemplo: Lucas.

o SQL é case-sensitive, o que significa que as letras maiúsculas e minúsculas são tratadas de forma diferente. No entanto, muitas implementações do SQL, incluindo o Microsoft SQL Server e o MySQL, são configuradas por padrão para não serem sensíveis à caixa, o que significa que você pode escrever com letras maiúsculas ou minúsculas e ele executará da mesma maneira. Porém, muitas pessoas preferem seguir a convenção de capitalização do SQL, que geralmente é maiúscula para palavras-chave e minúscula para nomes de colunas e tabelas, para facilitar a leitura e a manutenção do código SQL.

- Seleciona o que quer executar e dá F5 Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamenteS

- CREATE DATABASE aulaAtos; irá criar um novo BD, ele pode ser acessado no togglebutton (dropdown list)

- DROP DATABASE aulaAtos; irá “excluir” o BD

A saber:

* Joins: a capacidade de combinar dados de várias tabelas em uma única consulta é uma das características mais poderosas do SQL. Existem vários tipos de joins, como inner join, left join, right join e full outer join, e é importante entender a diferença entre eles e quando usá-los.
* Índices: índices são estruturas de dados que aceleram as consultas em tabelas grandes, permitindo que o banco de dados localize rapidamente os registros relevantes. É importante saber como criar e gerenciar índices para otimizar o desempenho do banco de dados.
* Stored procedures e funções: são blocos de código SQL que podem ser armazenados no banco de dados e reutilizados em várias consultas. Eles podem melhorar a eficiência e a manutenção do código SQL e fornecer um nível adicional de segurança.
* Transações: uma transação é uma sequência de operações que devem ser executadas como uma unidade indivisível. É importante entender como usar transações para garantir que as alterações no banco de dados sejam consistentes e duráveis.
* Segurança: é importante saber como garantir que o acesso ao banco de dados seja restrito apenas aos usuários autorizados e como criptografar dados confidenciais.

Esses são apenas alguns exemplos de tópicos que você pode querer explorar para se aprofundar no SQL. O modelo relacional de entidades e relacionamentos (ER) e o modelo entidade-relacionamento (MER) são importantes para entender como projetar um banco de dados eficiente, mas são apenas uma parte de um conjunto mais amplo de habilidades em SQL.