SGBD

> *Os seguintes componentes estão contidos em um SGDB típico: um compilador DDL (data definition language), um compilador DML (data manipulation language), um runtime database processor, um query compiler e um gerenciador de dados.*

A afirmação está **Certa**.

Um Sistema de Gerenciamento de Banco de Dados (SGBD) típico contém vários componentes que ajudam na definição, manipulação e gerenciamento de dados. Os componentes mencionados na afirmação são:

1. **Compilador DDL (Data Definition Language)** : Responsável por interpretar e executar comandos que definem a estrutura do banco de dados, como criação e modificação de tabelas.
2. **Compilador DML (Data Manipulation Language)** : Responsável por interpretar e executar comandos que manipulam os dados dentro das tabelas, como inserções, atualizações e exclusões.
3. **Runtime Database Processor**: Um componente que gerencia a execução de operações no banco de dados em tempo de execução, garantindo que as operações sejam realizadas corretamente.
4. **Query Compiler**: Um componente que analisa e otimiza consultas SQL, transformando-as em um formato que pode ser executado pelo SGBD.
5. **Gerenciador de Dados**: Um componente que gerencia o armazenamento, recuperação e manipulação de dados no banco de dados, garantindo a integridade e a segurança dos dados.

> *Geralmente, o sistema de banco de dados disponibiliza uma linguagem que é, na realidade, uma combinação de pelo menos duas linguagens subordinadas, uma DDL e uma DML.*

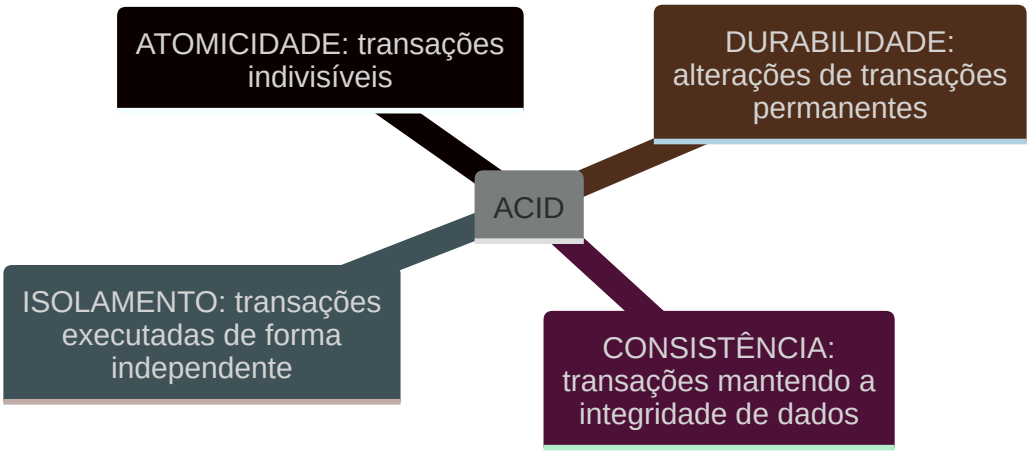
Certo. Geralmente, os sistemas de gerenciamento de banco de dados (SGBDs) disponibilizam uma linguagem que combina tanto a DDL (Data Definition Language) quanto a DML (Data Manipulation Language). A DDL é usada para definir e modificar a estrutura do banco de dados, como criar ou alterar tabelas e views, enquanto a DML é utilizada para manipular os dados dentro dessas estruturas, permitindo operações como inserção, atualização, exclusão e consulta de dados.

> *Quanto aos conceitos relativos à arquitetura de dados, julgue o item a seguir. O principal objetivo de um sistema de gerenciamento de banco de dados (SGDB) é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados.*

Certo. O principal objetivo de um sistema de gerenciamento de banco de dados (SGDB) é, de fato, abstrair e gerenciar a complexidade associada ao acesso, à persistência, à manipulação e à organização dos dados. Isso permite que os desenvolvedores de aplicações se concentrem na lógica de negócios e na funcionalidade da aplicação, sem se preocupar com os detalhes de como os dados são armazenados e gerenciados.

O SGDB fornece uma interface para que as aplicações possam interagir com os dados de forma eficiente e segura, garantindo que as operações sejam realizadas de acordo com as regras de integridade e consistência.

Propriedades de banco de dados



> *A respeito de arquitetura de bancos de dados, julgue o item a seguir. As propriedades ~~autenticidade~~, consistência, isolamento e durabilidade garantem confiabilidade às transações executadas em um banco de dados.*

Errado. As propriedades mencionadas referem-se ao conceito de ACID, que é fundamental para garantir a confiabilidade das transações em bancos de dados. A questão confunde Atomicidade com Autenticidade.

ACID é um acrônimo que representa:

- **Atomicidade**: As transações são tratadas como unidades indivisíveis, ou seja, ou todas as operações da transação são executadas ou nenhuma delas é.
- **Consistência**: As transações levam o banco de dados de um estado válido a outro estado válido, mantendo as regras de integridade.
- **Isolamento**: As transações são executadas de forma isolada, de modo que a execução concorrente de transações não afete o resultado final.
- **Durabilidade**: Uma vez que uma transação é confirmada, suas alterações são permanentes, mesmo em caso de falhas

> *As propriedades ~~autenticidade~~, consistência, isolamento e durabilidade garantem confiabilidade às transações executadas em um banco de dados.*

Errado, é atomicidade.

> *As propriedades ACID de uma transação garantem atomicidade, ~~confidencialidade~~, isolamento e durabilidade na sua execução.*

Errado, é consistência.

> *A atomicidade garante que uma transação seja completamente realizada ou*

> *completamente revertida, enquanto o isolamento assegura que as transações sejam realizadas sem interferir umas nas outras, mesmo quando executadas simultaneamente.*
>
> *A consistência dos dados é crucial para sua qualidade, pois garante que não haja discrepâncias entre conjuntos de dados diferentes ou dentro do mesmo conjunto.*

Integridade referencial

> *Julgue o seguinte item, relacionado a modelagem de dados: O fato de haver um campo de chave estrangeira com o valor* `NULL` *não viola a restrição de chave estrangeira.*

Correto. A restrição de chave estrangeira estabelece um vínculo entre uma coluna (ou conjunto de colunas) em uma tabela e a chave primária de outra tabela, assegurando que os valores da chave estrangeira correspondam a valores existentes na tabela referenciada. A restrição permite valores NULL, indicando que um registro pode não ter uma relação definida.

A restrição de chave estrangeira é uma regra em bancos de dados relacionais que garante a integridade referencial entre tabelas. Integridade referencial é a garantia de que as relações entre tabelas em um banco de dados sejam consistentes. Isso significa que um valor em uma coluna de chave estrangeira deve corresponder a um valor existente na coluna de chave primária de outra tabela, evitando referências inválidas.

> *A respeito de modelagem de dados, julgue o item a seguir. A integridade referencial é uma parte essencial de qualquer banco de dados relacional, que, entre outros benefícios, facilita as consultas.*

Correto. A integridade referencial é, de fato, uma parte essencial de qualquer banco de dados relacional e contribui para a eficiência e a precisão das consultas.

> *Quanto aos conceitos relativos à arquitetura de dados, julgue o item a seguir. Integridade referencial ocorre quando se liga duas tabelas por suas chaves primarias de forma íntegra.*

Certo. A integridade referencial é um conceito fundamental em bancos de dados relacionais que garante que as relações entre tabelas sejam mantidas de forma consistente. Isso ocorre quando uma chave estrangeira em uma tabela aponta para uma chave primária em outra tabela, assegurando que os valores da chave estrangeira correspondam a valores existentes na chave primária da tabela referenciada.

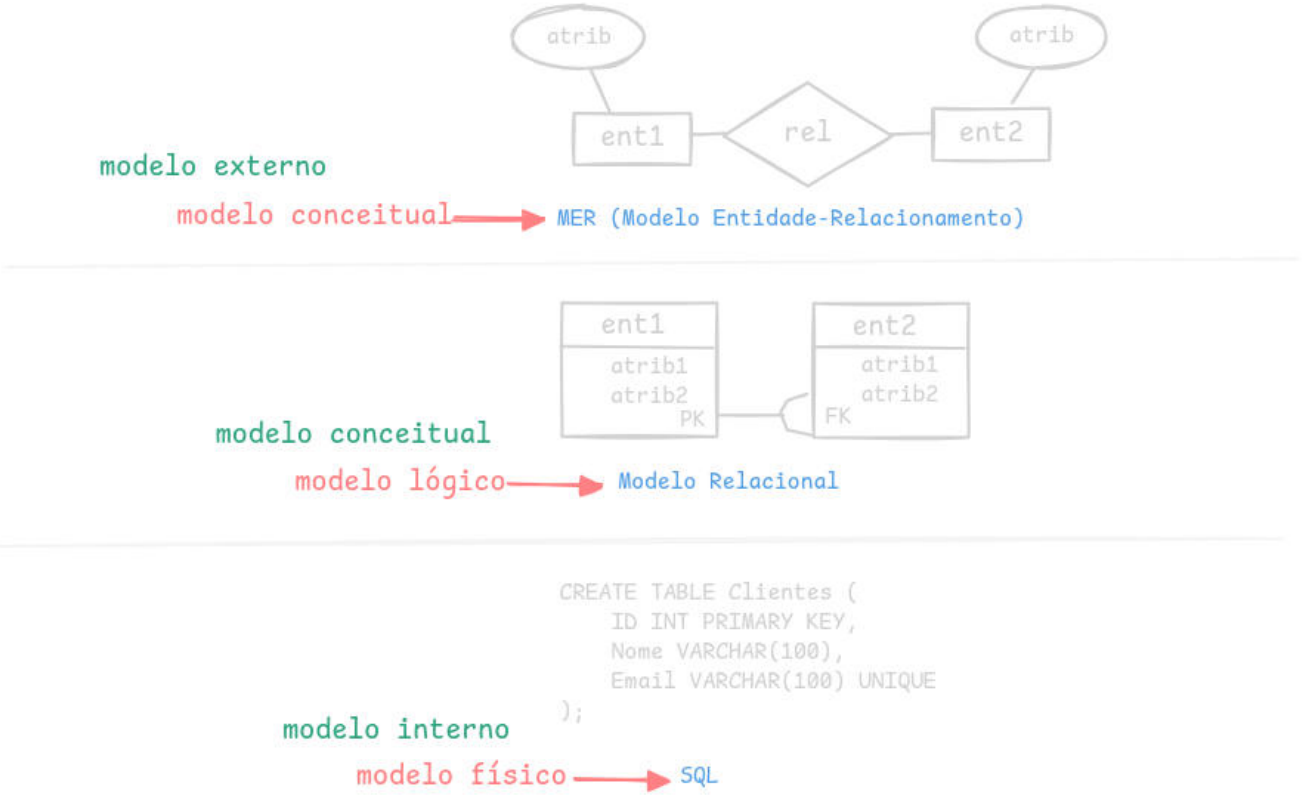
Se um registro na tabela referenciada (com a chave primária) for excluído ou alterado, a integridade referencial assegura que não haja referências inválidas na tabela que contém a chave estrangeira. Portanto, a afirmação está correta ao afirmar que a integridade referencial ocorre quando se liga duas tabelas por suas chaves primárias de forma íntegra.

Arquitetura de Banco de Dados

> *A respeito da arquitetura de banco de dados relacional, julgue o item seguinte. Um modelo relacional de banco de dados ~~não separa as estruturas de armazenamento físicas das estruturas de dados lógicas~~, assim administradores de banco de dados ~~podem gerenciar tanto o armazenamento de dados físicos quanto o acesso a esses dados~~*

Errado. Um modelo relacional de banco de dados, de acordo com a arquitetura de três esquemas (externo, conceitual e interno), separa as estruturas de armazenamento físicas das estruturas de dados lógicas. Essa separação permite que os administradores de banco de dados (DBAs) gerenciem o armazenamento físico dos dados de forma independente da forma como os dados são apresentados e acessados logicamente.

Essa abstração é uma das características fundamentais dos bancos de dados relacionais, pois permite que mudanças na estrutura física (como a forma de armazenamento) não afetem a forma como os dados são acessados logicamente pelos usuários e aplicações.



> *Em uma arquitetura de três esquemas de banco de dados, o nível que descreve a estrutura de todo o banco de dados para uma comunidade de usuários com foco na descrição de entidades, tipos de dados, relacionamentos, operações do usuário e restrições é denominado nível conceitual.*

O nível conceitual descreve a estrutura lógica do banco de dados como um todo, incluindo entidades, tipos de dados, relacionamentos, operações do usuário e

restrições, de forma que seja compreensível para uma comunidade de usuários. Ele fornece uma visão unificada e abstrata dos dados, independentemente de como eles são armazenados fisicamente.

O nível externo não é a resposta correta porque ele se refere a visões específicas dos dados que são apresentadas a diferentes usuários ou grupos de usuários. Cada visão externa pode ser adaptada às necessidades e requisitos de um usuário específico, mas não fornece uma descrição abrangente da estrutura do banco de dados como um todo.

> *A respeito de arquitetura de bancos de dados, julgue o item a seguir. Na arquitetura de três esquemas, o nível externo apresenta uma série de visões do usuário.*

Certo. Na arquitetura de três esquemas, o nível externo é, de fato, responsável por apresentar uma série de visões do usuário. Cada visão externa pode ser adaptada às necessidades específicas de diferentes usuários ou grupos de usuários, permitindo que eles interajam com o banco de dados de maneira que faça sentido para suas atividades. Isso proporciona uma camada de abstração que oculta a complexidade do modelo conceitual e do modelo interno, permitindo que os usuários trabalhem com uma representação simplificada e relevante dos dados.

> *A respeito de banco de dados, julgue o próximo item. Os bancos de dados que têm a arquitetura orientada a objeto armazenam os dados em ~~tabelas com seus respectivos atributos~~.*

Errado. Os bancos de dados que têm a arquitetura orientada a objeto não armazenam os dados em tabelas, como ocorre nos bancos de dados relacionais. Em vez disso, eles armazenam dados como objetos, que podem incluir tanto dados (atributos) quanto comportamentos (métodos). Essa abordagem é baseada nos princípios da programação orientada a objetos, onde os dados são encapsulados em objetos que podem ser manipulados de maneira mais flexível e complexa.

> *Com relação à arquitetura e à estrutura de banco de dados, julgue o próximo item. Em banco de dados, uma superchave se caracteriza por um conjunto de um ou mais atributos que permitem identificar uma única entidade em um conjunto de entidades.*

Certo. Uma superchave é, de fato, um conjunto de um ou mais atributos que pode ser usado para identificar de forma única uma entidade em um conjunto de entidades. Em um banco de dados relacional, uma superchave pode incluir atributos que não são necessários para a identificação única, mas que, em conjunto, garantem que cada registro na tabela seja distinto.

Por exemplo, em uma tabela de clientes, um conjunto de atributos como "ID do Cliente" pode ser uma superchave, mas também "Nome" e "Data de Nascimento" juntos

poderiam formar uma superchave, desde que essa combinação seja única para cada cliente.

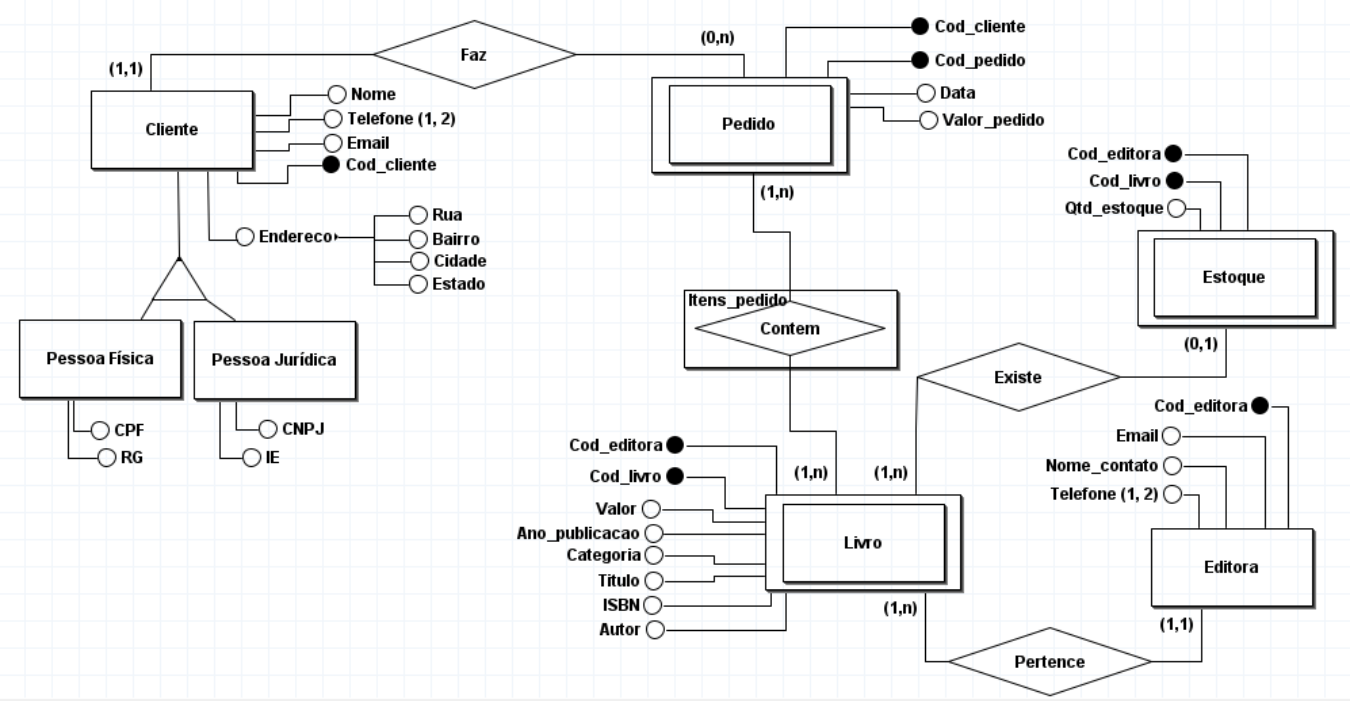
Portanto, a afirmação está correta.

Diagramas entidade-relacionamento e mapeamento para modelo relacional

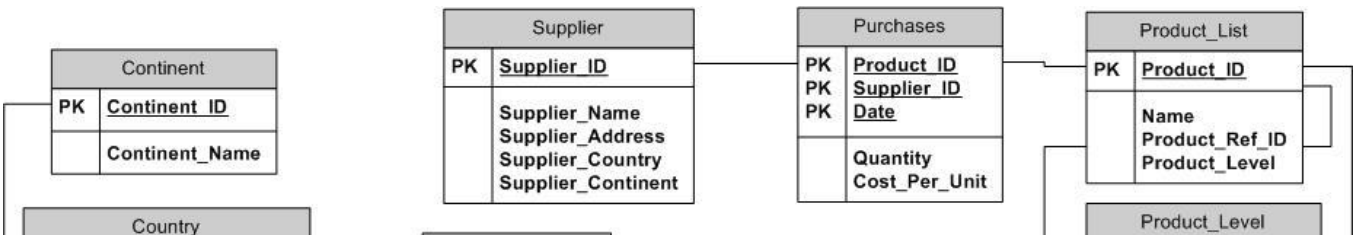
> *Olhando pra um grafico, como distinguir modelo entidade-relacionamento (MER) do modelo relacional?*

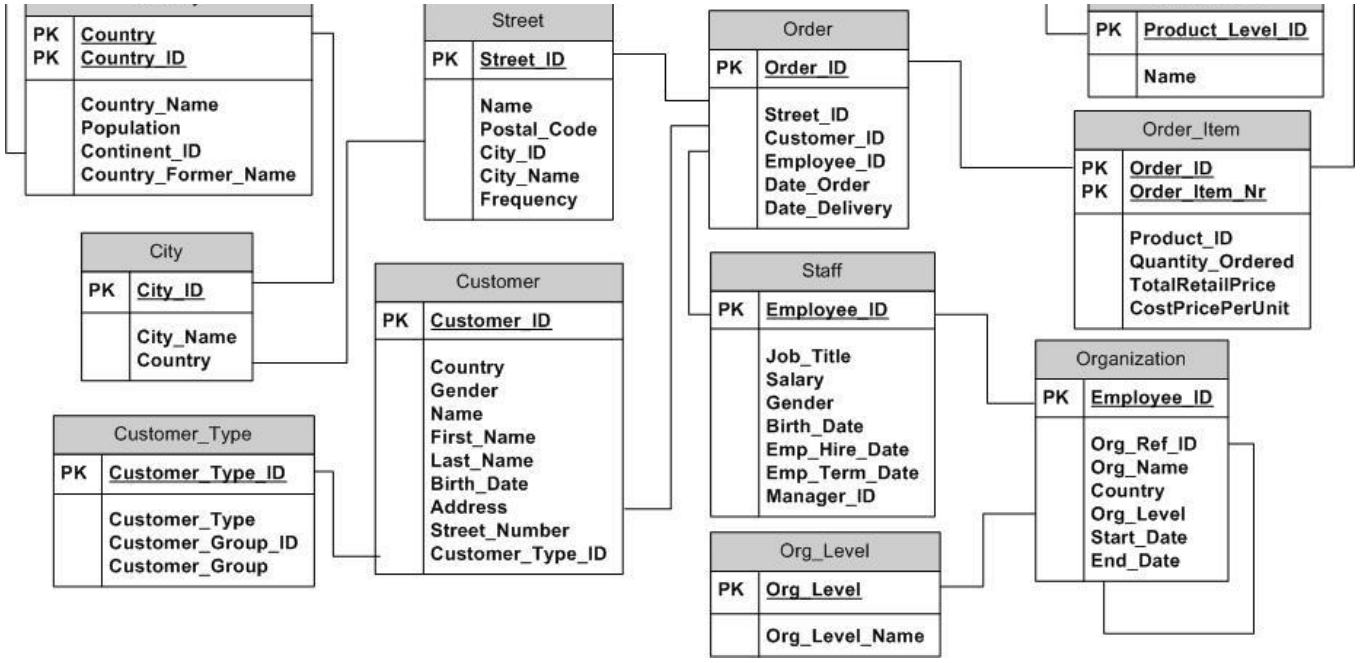
Aqui está um resumo das principais diferenças entre o modelo entidade-relacionamento (MER) e o modelo relacional:

- Modelo Entidade-Relacionamento (MER):
  - **Entidades**: Representadas por retângulos.
  - **Atributos**: Representados por elipses conectadas às entidades.
  - **Relacionamentos**: Representados por losangos ou linhas entre entidades.
  - **Cardinalidade**: Indica a quantidade de associações entre entidades.



- Modelo Relacional:
  - **Tabelas**: Representadas por retângulos que correspondem a entidades ou relacionamentos.
  - **Colunas**: Representam os atributos da entidade.
  - **Chaves Primárias e Estrangeiras**: Identificam registros e estabelecem relacionamentos entre tabelas.
  - **Linhas**: Cada linha representa uma instância da entidade.





Aqui está uma tabela comparativa que destaca as principais diferenças entre o Modelo Entidade-Relacionamento (MER) e o Modelo Relacional:

Aspecto	**Modelo Entidade-Relacionamento (MER)**	**Modelo Relacional**
Representação	Diagramas com entidades, atributos e relacionamentos	Tabelas com linhas e colunas
Entidades	Representadas por retângulos	Representadas por tabelas
Atributos	Representados por elipses conectadas às entidades	Representados por colunas nas tabelas
Relacionamentos	Representados por losangos ou linhas	Representados por chaves estrangeiras ou tabelas de relacionamento
Chave Primária	Não é explicitamente definida	Deve ser definida para cada tabela
Cardinalidade	Indicada nas linhas que conectam entidades	Representada por chaves estrangeiras e tabelas de relacionamento
Complexidade	Mais conceitual e visual	Mais técnica e focada na implementação
Objetivo	Modelar a estrutura de dados de forma intuitiva	Implementar a estrutura de dados em um SGBD
Normalização	Normalização não é uma preocupação direta	Normalização é essencial para evitar redundâncias

Aqui está um resumo do mapeamento de diagramas entidade-relacionamento (DER) para o modelo relacional:

1. **Entidades**: Cada entidade se torna uma tabela.
2. **Atributos**: Atributos da entidade se tornam colunas da tabela.
3. **Chave Primária**: Defina uma chave primária para cada tabela.
4. **Relacionamentos**:
    - **Um-para-Um**: Adicione a chave primária de uma tabela como chave estrangeira na outra.
    - **Um-para-Muitos**: A chave primária da tabela "um" é a chave estrangeira na tabela "muitos".
    - **Muitos-para-Muitos**: Crie uma nova tabela de relacionamento com chaves primárias de ambas as tabelas como chaves estrangeiras.
5. **Atributos de Relacionamento**: Se houver, adicione-os na tabela de relacionamento.
6. **Normalização**: Revise as tabelas para garantir que estejam normalizadas.

Esse processo assegura que a estrutura do modelo relacional reflita corretamente as entidades e relacionamentos do DER.

Modelagem de Dados

O **projeto conceitual** de um banco de dados é representado através do **MER (Modelo Entidade-Relacionamento)** , que define as entidades, atributos e relacionamentos de forma abstrata. Em seguida, esse modelo é transformado em um **modelo lógico**, que pode ser implementado como um **modelo relacional**, onde as entidades se tornam tabelas e os relacionamentos são representados por chaves primárias e estrangeiras. Por fim, o modelo lógico é convertido em um **modelo físico**, que especifica como os dados serão armazenados fisicamente no sistema de gerenciamento de banco de dados (SGBD), incluindo detalhes como tipos de dados, índices, particionamento, e a estrutura de armazenamento, otimizando o desempenho e a eficiência do acesso aos dados.

> *No que se refere a níveis de abstração, modelos de dados e normalização em bancos de dados, julgue o próximo item. Os modelos de dados representacionais incorporam alguns conceitos acerca da forma como os dados são organizados no armazenamento computacional, enquanto mantêm a facilidade de compreensão pelos usuários finais.*

A afirmação está **certa**. Os modelos de dados representacionais, como o modelo entidade-relacionamento (MER) e o modelo relacional, são projetados para representar a estrutura e a organização dos dados de uma forma que seja compreensível para os usuários finais, ao mesmo tempo em que incorporam conceitos sobre como os dados são organizados e armazenados no sistema computacional. Esses modelos buscam equilibrar a complexidade técnica do armazenamento de dados com a necessidade de uma representação clara e intuitiva para os usuários que interagem com o sistema.

Modelagem conceitual (MER)

Notação Peter Chen

A notação de Peter Chen, ou notação ER (Entidade-Relacionamento), é uma técnica de modelagem de dados introduzida por Peter Chen em 1976. Ela é utilizada para representar visualmente a estrutura de um banco de dados, facilitando a compreensão das entidades, atributos e relacionamentos.

Elementos Principais:

1. **Entidades**: Representadas por retângulos (ex: "Cliente").
2. **Atributos**: Representados por elipses conectadas às entidades (ex: "Nome").
3. **Relacionamentos**: Representados por losangos ou linhas (ex: "Compra").
4. **Cardinalidade**: Indica a quantidade de associações entre entidades (ex: um para muitos).
5. **Chaves**: Indicam como os dados são identificados e relacionados.

> *A respeito de modelagem de dados, julgue o item a seguir: A **modelagem conceitual***

> *representa o negócio sob a perspectiva dos dados, e, em projetos de TI, o principal objetivo de um modelo de dados conceitual é fornecer uma visão geral dos requisitos de informação envolvidos no projeto.*

> *O modelo de abstração que tem como objetivo representar uma estrutura de banco de dados sem preocupação com a implementação é o modelo conceitual.*

Correto. O modelo conceitual tem como objetivo representar a estrutura de um banco de dados de forma abstrata, sem se preocupar com detalhes de implementação. Ele foca em descrever as entidades, atributos e relacionamentos de maneira que seja compreensível para os usuários e analistas, antes de se passar para os modelos lógico e físico, que tratam da implementação e armazenamento dos dados.

Atributos

Aqui está uma tabela explicativa para os diferentes tipos de atributos em modelagem de dados:

Tipo de Atributo	**Descrição**	**Exemplo**
Simples	Atributo que não pode ser dividido em partes menores.	"Nome" (ex: "João")
Composto	Atributo que pode ser dividido em subatributos, representando partes menores.	"Endereço" (ex: "Rua", "Número", "Cidade")
Derivado	Atributo cujo valor é calculado a partir de outros atributos.	"Idade" (calculada a partir da "Data de Nascimento")
Chave	Atributo que identifica de forma única cada registro em uma tabela.	"CPF" ou "ID do Cliente"
Monovalorado	Atributo que pode ter apenas um valor por registro.	"Telefone" (um único número)
Multivalorado	Atributo que pode ter múltiplos valores por registro.	"Hobbies" (ex: "Futebol", "Leitura", "Música")

Modelagem lógica (Relacional)

> *Na modelagem de dados **relacional**, diferentes tabelas de dados se unem ou se vinculam a partir de chaves que representam a relação de entidades do mundo real.*

Correto. A afirmação destaca que, na modelagem de dados relacional, as tabelas se conectam por meio de chaves. Cada tabela representa uma entidade do mundo real, com uma chave primária que identifica de forma única cada registro. As chaves estrangeiras permitem relacionar diferentes tabelas, refletindo como as entidades interagem. Isso ajuda a manter a integridade referencial, garantindo que as relações entre as tabelas sejam válidas. Em resumo, as chaves são essenciais para unir e organizar dados em um banco de dados relacional.

> *Julgue o seguinte item, relacionado a modelagem de dados: O **modelo lógico** pode conter chaves primárias e estrangeiras e pode ser usado em vários bancos de dados, tais como SQL Server, MySql, Oracle e PostgreSql.*

A afirmação está **certa**.

- **Modelo Lógico**: O modelo lógico de dados é uma representação detalhada da estrutura dos dados, incluindo entidades, atributos, chaves primárias e chaves estrangeiras. Ele é independente do sistema de gerenciamento de banco de dados (SGBD) específico.
- **Chaves Primárias e Estrangeiras**: O modelo lógico inclui a definição de chaves primárias (que identificam de forma única os registros em uma tabela) e chaves estrangeiras (que estabelecem relações entre tabelas).
- **Compatibilidade com Vários SGBDs**: O modelo lógico pode ser implementado em diferentes sistemas de gerenciamento de banco de dados, como SQL Server, MySQL, Oracle e PostgreSQL, pois esses SGBDs suportam conceitos de chaves primárias e estrangeiras.

> *No que se refere aos conceitos de modelagem de dados, julgue o seguinte item. Modelo ~~físico~~ é o modelo de dados que representa a estrutura de dados de um banco de dados conforme vista pelo usuário do sistema de gerenciamento de banco de dados.*

A afirmação está **errada**. O modelo físico é a representação da estrutura de dados de um banco de dados conforme implementado no sistema de gerenciamento de banco de dados (SGBD), incluindo detalhes como tipos de dados, índices, e a forma como os dados são armazenados fisicamente. Por outro lado, o que representa a estrutura de dados conforme vista pelo usuário é o **modelo lógico**, que é uma abstração que descreve a estrutura dos dados sem se preocupar com como eles são armazenados fisicamente. Portanto, a descrição dada se refere ao modelo lógico, não ao modelo físico.

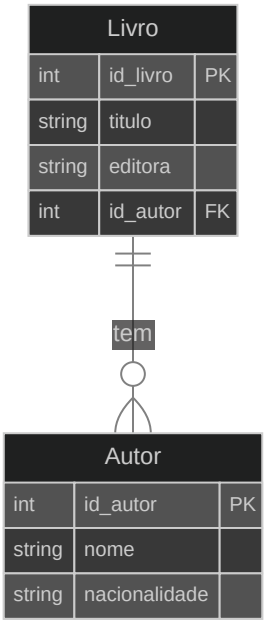
Cardinalidade

> *Com base nos princípios relacionais e nas regras associadas do modelo entidade-relacionamento, julgue o item subsecutivo: Em geral, sempre que existir um relacionamento com cardinalidade de um-para-muitos, a referência lógica estará*

> *colocada na entidade que possuir o lado um da cardinalidade.*

Certo. De acordo com os princípios do modelo entidade-relacionamento, quando existe um relacionamento com cardinalidade de um-para-muitos, a referência lógica (chave estrangeira) é colocada na entidade que possui o lado "muitos" da cardinalidade. Isso porque a entidade do lado "um" da cardinalidade é a entidade principal, enquanto a entidade do lado "muitos" é a entidade dependente que faz referência à entidade principal.

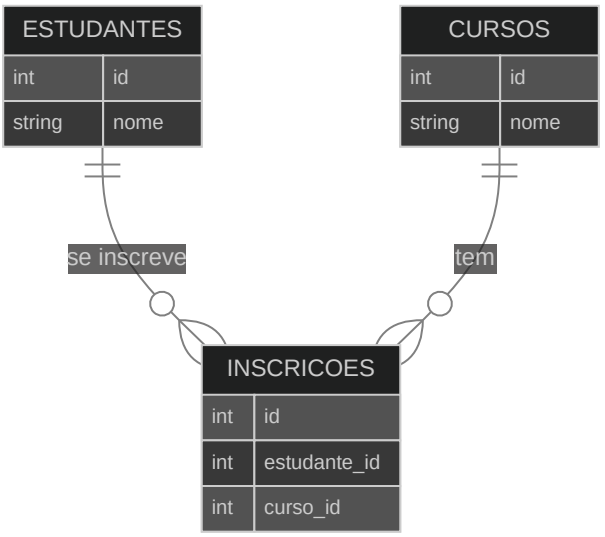
Na representação gráfica abaixo fica clara que a referência lógica (chave estrangeira) deve estar na entidade "Livro", pois ela possui o lado "muitos" da cardinalidade.



> *Com base nos princípios relacionais e nas regras associadas do modelo entidade-relacionamento, julgue o item subsecutivo: Todo relacionamento muitos-para-muitos pode ser entendido como uma entidade dissociativa.*

Certo. De acordo com os princípios do modelo entidade-relacionamento, todo relacionamento muitos-para-muitos pode ser entendido como uma entidade dissociativa. Isso significa que, quando temos um relacionamento muitos-para-muitos entre duas entidades, é necessário criar uma nova entidade intermediária (entidade dissociativa) para representar esse relacionamento. A entidade dissociativa possui as chaves primárias das duas entidades originais como chaves estrangeiras, formando uma nova chave primária composta.

Vamos considerar um exemplo simples com duas entidades: **Estudantes** e **Cursos**. Um estudante pode se inscrever em vários cursos, e um curso pode ter vários estudantes inscritos. Para representar isso, podemos criar uma entidade dissociativa chamada **Inscrições**.



> *Com relação aos níveis de abstração e modelagem de dados, julgue o item seguinte. Durante a construção de um modelo entidade-relacionamento, os atributos multivalorados devem ser evitados, pois eles podem ocultar entidades e relacionamentos.*

A afirmação está **certa**. Durante a construção de um modelo entidade-relacionamento (MER), os atributos multivalorados devem ser evitados sempre que possível, pois eles podem complicar a modelagem e ocultar a verdadeira estrutura de entidades e relacionamentos. Atributos multivalorados podem indicar que uma entidade pode estar relacionada a várias instâncias de um atributo, o que pode ser melhor representado por uma nova entidade ou relacionamento. Portanto, a prática recomendada é normalizar esses atributos para manter a clareza e a integridade do modelo.

> *Acerca de modelagem de dados, julgue o item a seguir. Qualquer subconjunto dos campos de uma relação pode ser uma chave de pesquisa em um arquivo de índice.*

A afirmação está **errada**.

Uma chave de pesquisa em um arquivo de índice deve ser um conjunto de campos que, em conjunto, identifica de forma única cada registro na relação (tabela). Para que um subconjunto de campos possa ser utilizado como chave de pesquisa, ele deve atender ao critério de unicidade, ou seja, não pode haver registros duplicados para os valores desse subconjunto. Portanto, nem qualquer subconjunto de campos pode ser uma chave de pesquisa; apenas aqueles que garantem a unicidade dos registros podem ser utilizados como tal.

Normalização de dados

> *Uma tabela estará na segunda forma normal (2FN) quando, ~~além de estar na terceira forma normal (3FN), ela contiver dependências funcionais parciais~~*

Errado. Uma tabela está na Segunda Forma Normal (2FN) quando, além de estar na Primeira Forma Normal (1FN), não contém dependências funcionais parciais. Isso significa que todos os atributos não-chave devem depender da chave primária como um todo, e não apenas de parte dela.

> *A passagem à terceira forma normal (3FN) tem como objetivo principal gerar o modelo lógico de dados; por isso, ~~ela não visa eliminar redundância de dados~~, como ocorre com as demais formas normais.*

Errado. A passagem à terceira forma normal (3FN) tem como objetivo principal eliminar redundâncias e garantir a integridade dos dados em um banco de dados. A 3FN busca assegurar que todos os atributos não-chave sejam dependentes apenas da chave primária, evitando dependências transitivas. Portanto, a 3FN não apenas gera o modelo lógico de dados, mas também visa eliminar redundâncias, ao contrário do que foi afirmado.

> *O modelo conceitual, que reflete uma estrutura simplificada do banco de dados, é ~~responsável por registrar como os dados estão armazenados no sistema de gerenciamento de banco de dados~~ (SGBD).*

Errado. O **modelo conceitual** é uma representação abstrata e simplificada dos dados que descreve o que os dados representam e como eles se relacionam, sem se preocupar com a forma como os dados estão fisicamente armazenados no sistema de gerenciamento de banco de dados (SGBD). O **modelo físico,** por outro lado, é que se preocupa com a implementação e o armazenamento dos dados no SGBD. Portanto, a afirmação de que o modelo conceitual é responsável por registrar como os dados estão armazenados no SGBD está incorreta.

> *Comparativamente aos usados pelos usuários leigos, os modelos de dados utilizados por programadores são considerados menos abstratos, pois contêm mais detalhes de como as informações estão organizadas internamente no banco de dados.*

Certo. Os modelos de dados utilizados por programadores geralmente são considerados menos abstratos em comparação com os modelos usados por usuários leigos. Isso ocorre porque os programadores precisam de um entendimento mais detalhado da estrutura interna do banco de dados, incluindo como os dados são organizados, armazenados e acessados. Enquanto os usuários leigos podem se

concentrar mais nas entidades e relacionamentos de alto nível, os programadores lidam com aspectos técnicos e implementacionais, como tabelas, colunas, tipos de dados e índices.

> *A transformação do esquema de tabela não normalizada em um esquema relacional na primeira forma normal (1FN) consiste da eliminação das tabelas aninhadas.*

Certo. A transformação de um esquema de tabela não normalizada para um esquema relacional na primeira forma normal (1FN) envolve a eliminação de tabelas aninhadas e a garantia de que todos os atributos em uma tabela contenham valores atômicos (ou seja, indivisíveis). Na 1FN, cada coluna deve conter apenas um valor por linha, e não deve haver grupos repetitivos ou múltiplos valores em uma única célula. Portanto, a afirmação está correta.

> *Um bom projeto de banco de dados deve ter por objetivo evitar falhas provocadas por problemas que podem ser sanados por meio da normalização, que também contribui para eliminar as misturas de assuntos e as redundâncias desnecessárias de dados correspondentes. Quanto ao processo de normalização em projetos de banco de dados, julgue o item a seguir: A desnormalização é apenas tolerável quando houver imperativos rígidos de desempenho, como no caso dedata warehouse, ou se o sistema não conseguir atingir um patamar mínimo de desempenho sem o processo de desnormalização.*

A afirmação está **certa**. A desnormalização é aceitável principalmente quando há exigências rigorosas de desempenho, como em data warehouses, ou quando o sistema não consegue atingir um desempenho mínimo sem ela. Embora a normalização evite redundâncias e mantenha a integridade dos dados, a desnormalização pode ser necessária para otimizar consultas e melhorar a velocidade de acesso aos dados em situações específicas.

Linguagem de Consulta SQL

> *O comando HAVING é usado para ~~unir duas ou mais tabelas~~.*

A afirmação está **errada**. O comando `HAVING` não é usado para unir duas ou mais tabelas. Em SQL, a cláusula `HAVING` é utilizada para filtrar resultados de grupos agregados, ou seja, você pode aplicar condições sobre os resultados que foram agrupados.

```
Plain text
SELECT produto, SUM(vendas) AS total_vendas
FROM vendas
GROUP BY produto
HAVING SUM(vendas) > 1000;
```

Para unir duas ou mais tabelas, você usaria comandos como `JOIN`.

> ***SELECT, COUNT e DISTINCT*** *~~são instruções mutuamente exclusivas em SQL~~ e, dessa maneira, ~~não podem ser utilizadas em uma mesma sentença SQL.~~*

Errado, a afirmação está incorreta. As instruções SQL SELECT, COUNT e DISTINCT não são mutuamente exclusivas e podem ser utilizadas em uma mesma sentença SQL.

Por exemplo, a seguinte consulta é perfeitamente válida:

```
Plain text
SELECT DISTINCT COUNT(*) FROM clientes;
```

Nesta consulta, a instrução SELECT é utilizada para selecionar os dados, a instrução DISTINCT é utilizada para retornar apenas valores únicos, e a instrução COUNT(*) é utilizada para contar o número de registros retornados.

> *Em uma consulta SQL, ~~tuplas duplicadas são automaticamente eliminadas pelo SGBD~~, com o objetivo de aumentar o desempenho.*

Não, tuplas duplicadas não são eliminadas automaticamente pelo SGBD em uma consulta SQL. Para remover duplicatas, é necessário usar a cláusula `DISTINCT`. Sem isso, todas as tuplas, incluindo duplicatas, serão retornadas. `SELECT DISTINCT coluna1, coluna2 FROM tabela;`

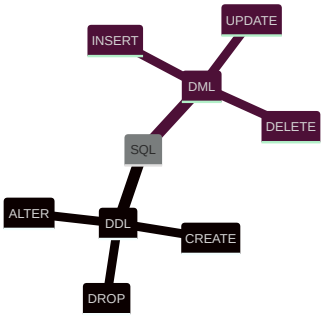
> *O comando HAVING é ~~usado para unir duas ou mais tabelas~~.*

Errado. O comando HAVING é utilizado em consultas SQL para filtrar resultados de grupos criados pela cláusula GROUP BY, e não para unir tabelas. Para unir tabelas, utiliza-se a cláusula JOIN.

> *Quando utilizado o comando SELECT, podem-se incluir várias subqueries internas na mesma instrução.*

Certo. No comando SELECT, é possível incluir várias subqueries internas na mesma instrução. Essas subqueries podem ser utilizadas em diferentes partes da consulta, como na cláusula SELECT, WHERE ou FROM, permitindo realizar operações complexas e obter resultados mais refinados.

DDL



Enter image caption

> *Na DDL, que é uma linguagem declarativa, é descrito o que se deseja fazer, em vez de como fazê-lo, como comandos para definir tabelas e procedures, tal qual exemplificado a seguir:* `CREATE TABLE <tablename> (col1 int, col2 int, col3 int)`

A afirmação está **Certa**. A DDL (Data Definition Language) é realmente uma linguagem declarativa que descreve a estrutura dos dados que se deseja criar ou modificar, sem especificar como essas operações devem ser realizadas em termos de implementação. O exemplo fornecido, `CREATE TABLE <tablename> (col1 int, col2 int, col3 int)`, é um comando DDL que define a criação de uma tabela com três colunas.

> *A seguir, é apresentado um ~~exemplo de comando DDL~~:* `DELETE TABLE DISCIPLINA WHERE Carga_horaria > 3;`

A afirmação está **Errada**. O comando DDL é utilizado para definir e modificar a estrutura de objetos no banco de dados, como tabelas, índices e esquemas. Exemplos de comandos DDL incluem `CREATE`, `ALTER` e `DROP`. O comando `DELETE` é utilizado para remover registros de uma tabela, o que o classifica como um comando DML.

> *As bases de dados são criadas via comando CREATE DATABASE e destruídas via ~~DELETE~~ DATABASE*

Errado, o certo é **DROP DATABASE** comando é utilizado para remover uma base de dados existente.

> *Em banco de dados, existem linguagens de definição e de manipulação de dados. CREATE e ALTER são palavras reservadas utilizadas em DDL (data definition language).*

Correto

- **CREATE**: É um comando DDL utilizado para criar novos objetos no banco de dados, como tabelas, índices, visões (views), entre outros. Por exemplo, `CREATE TABLE` é usado para criar uma nova tabela.
- **ALTER**: É um comando DDL utilizado para modificar a estrutura de objetos existentes no banco de dados. Por exemplo, `ALTER TABLE` pode ser usado para adicionar, modificar ou excluir colunas em uma tabela existente enquanto `UPDATE` é um comando DML que altera os dados dentro das tabelas

> *A DDL (Data Definition Language) é utilizada em bancos de dados ~~para comandos de UPDATE nas tabelas~~.*

A afirmação está **Errada**. A DDL (Data Definition Language) é utilizada para definir e modificar a estrutura de objetos no banco de dados, como tabelas, índices e esquemas. Comandos típicos de DDL incluem `CREATE`, `ALTER` e `DROP`.

O comando `UPDATE`, por outro lado, é um comando da DML (Data Manipulation Language), que é utilizado para manipular os dados dentro das tabelas, ou seja, para atualizar registros existentes.

> *As cláusulas DROP e ALTER podem ser utilizadas em ambientes de desenvolvimento PLSQL, ~~mas não em SQL~~.*

A afirmação está **Errada**. As cláusulas `DROP` e `ALTER` são comandos da DDL (Data Definition Language) e podem ser utilizadas tanto em SQL quanto em PL/SQL.

- `DROP`: É usado para remover objetos do banco de dados, como tabelas, índices ou visões. Por exemplo, `DROP TABLE nome_da_tabela;`.

PL/SQL (Procedural Language/Structured Query Language) é uma extensão da linguagem SQL desenvolvida pela Oracle. É uma linguagem de programação procedural que permite combinar comandos SQL com lógica de programação

> *A linguagem SQL, que é importante para o gerenciamento de dados em SGBD permite a criação de dados, possibilitando a estruturação de dados em DDL; admite o controle de transações de dados e ~~é ideal para o suporte a dados em tempo real~~.*

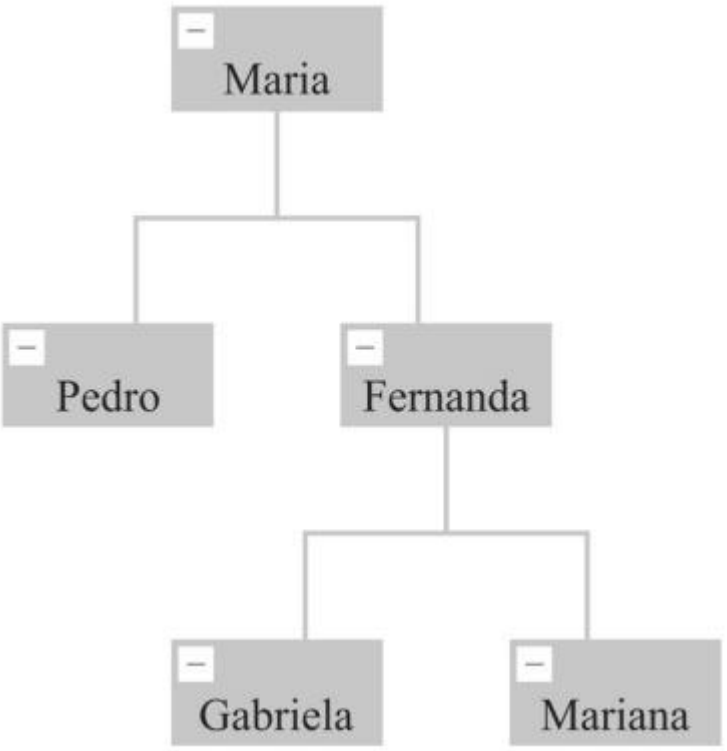
Errado.

I. **Permite a criação de dados, possibilitando a estruturação de dados em DDL.** - Correto. A linguagem SQL inclui comandos DDL (Data Definition Language) que permitem a criação e modificação da estrutura de dados, como tabelas e índices.

II. **Admite o controle de transações de dados.** - Correto. SQL suporta controle de transações através de comandos como `BEGIN`, `COMMIT` e `ROLLBACK`, permitindo gerenciar

transações de forma segura.

III. **É ideal para o suporte a dados em tempo real.** - Incorreto. Embora SQL possa ser usado em sistemas que requerem dados em tempo real, não é especificamente projetado para isso. Existem outras tecnologias e bancos de dados que são mais adequados para aplicações em tempo real.

> *Considere que se deseje criar uma tabela que represente o organograma a seguir e o código de definição da tabela mais abaixo:*
>
> 
>

Plain text

```
-- Criação da tabela 'colaborador'
CREATE TABLE colaborador (
    -- Coluna 'id' do tipo inteiro, que é a chave primária da tabela
    id INTEGER PRIMARY KEY,

    -- Coluna 'nome' do tipo varchar, que pode armazenar até 50 caracteres
    nome VARCHAR(50),

    -- Coluna 'chefia' do tipo inteiro, que pode referenciar o 'id' de
outro colaborador
    chefia INTEGER,

    -- Definição da restrição de chave estrangeira
    -- A coluna 'chefia' referencia a coluna 'id' na mesma tabela
'colaborador'
    CONSTRAINT fk_colaborador FOREIGN KEY (chefia) REFERENCES
colaborador(id)
);
```

> *A criação de visões em um banco de dados relacional pode ser feita por meio de sentenças escritas em uma DDL.*

Certo. A criação de visões (ou views) em um banco de dados relacional é feita por meio de sentenças escritas em uma DDL (Data Definition Language). O comando SQL utilizado para criar uma view é `CREATE VIEW`, que é uma instrução DDL.

> *Uma view é considerada uma tabela virtual derivada de outras tabelas ou de outras views previamente definidas e não necessariamente existe em forma física.*

Certo. Uma view (ou visão) é, de fato, uma tabela virtual que é derivada de outras tabelas ou de outras views. Ela não armazena os dados fisicamente, mas sim uma consulta que é executada quando a view é acessada. As views são usadas para simplificar consultas complexas, fornecer uma camada de segurança (limitando o acesso a certos dados) e apresentar dados de uma maneira específica.

> *A implementação de uma view consiste na geração de uma tabela ~~física~~ que consolida os dados de outras tabelas do banco de dados.*

Errado. A implementação de uma view não consiste na geração de uma tabela física. Uma view é uma tabela virtual que é baseada em uma consulta a outras tabelas ou views. Ela não armazena dados fisicamente; em vez disso, armazena a definição da consulta que é executada quando a view é acessada.

> *Todas as views criadas especificamente pelo administrador de banco de dados são ~~armazenadas fisicamente~~ no banco de dados.*

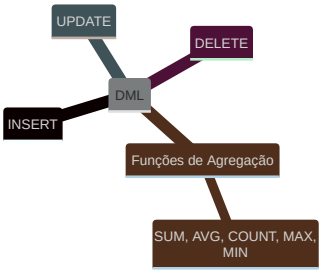
Errado. As views, por definição, são tabelas virtuais que não armazenam dados fisicamente. Elas contêm apenas a definição da consulta que deve ser executada para recuperar os dados de outras tabelas ou views. Quando uma view é criada, sua definição é armazenada no banco de dados, mas os dados em si não são armazenados fisicamente na view.

> *Uma view materializada armazena ~~apenas a consulta que define e apresenta o resultado sempre atualizado de todas as operações de definição de dados DDL.~~*

Errado. Uma view materializada armazena os resultados da consulta em um formato físico, ou seja, ela armazena os dados resultantes da consulta em vez de apenas a definição da consulta. Isso significa que os dados em uma view materializada não são atualizados automaticamente com cada operação de definição de dados (DDL) nas

tabelas subjacentes. Para manter os dados atualizados, é necessário realizar uma operação de refresh na view materializada.

DML



> *A linguagem de manipulação de dados (DML) inclui instruções que ~~modificam a estrutura de um banco de dados.~~*

Errado. A linguagem de manipulação de dados (DML) é responsável por operações que **manipulam os dados** dentro de um banco de dados, como inserção, atualização, exclusão e consulta de dados.

As instruções que modificam a estrutura do banco de dados, como criação ou alteração de tabelas, índices e views, pertencem à linguagem de definição de dados (DDL - Data Definition Language).

> *A DML não procedimental exige que um usuário especifique quais dados são necessários sem especificar como obtê-los.*

DML não procedimental refere-se a uma abordagem de manipulação de dados em que o usuário especifica **o que** deseja obter ou manipular, sem precisar descrever **como** isso deve ser feito. Essa abordagem é comum em linguagens de consulta de bancos de dados, como SQL (Structured Query Language).

> *A DML é uma linguagem que interage com os ~~objetos do banco de dados, em vez de interagir com os dados.~~*

Errado. A DML (Data Manipulation Language) é uma linguagem que interage diretamente com os dados armazenados em um banco de dados, permitindo operações como inserção, atualização, exclusão e consulta de dados. Embora a DML possa envolver objetos do banco de dados (como tabelas e registros), seu foco principal é a manipulação dos dados contidos nesses objetos.

> *Os exemplos mais comuns de palavras reservadas SQL utilizadas na DML são* `INSERT, UPDATE, SELECT e DELETE.`

> *A expressão SQL a seguir ~~está sintaticamente correta~~ e permite inserir dois alunos de nomes Pedro e Maria na tabela alunos:* `INSERT VALUES ('Pedro', 'Maria') INTO`

> *`alunos;`*

Errado. A sintaxe da expressão SQL apresentada está incorreta. A forma correta de inserir múltiplos registros em uma tabela usando o comando `INSERT` é especificar a tabela e os valores a serem inseridos de maneira adequada.

```
Plain text
INSERT INTO alunos (nome) VALUES ('Pedro'), ('Maria');
```

> *A query apresentada a seguir ~~gera erro de sintaxe em sua execução, porque VALUES não é palavra reservada em SQL~~. `INSERT INTO clientes (nome, endereco) VALUES ('Ana', 'rua A')`*

A afirmação está incorreta. A palavra-chave "`VALUES`" é, sim, uma palavra reservada em SQL e é utilizada corretamente na consulta apresentada para inserir novos registros na tabela "clientes".

> *Em linguagem de manipulação de dados DML, o operador SQL `BETWEEN` serve para delimitar o valor de uma coluna na cláusula `WHERE`.*

Correto. O operador SQL `BETWEEN` é utilizado na cláusula `WHERE` para filtrar resultados com base em um intervalo de valores. Ele permite que você especifique um valor mínimo e um valor máximo, e retorna os registros que têm valores dentro desse intervalo. Por exemplo, a consulta abaixo retornaria todos os produtos cujo preço está entre 10 e 50.

```
Plain text
SELECT * FROM produtos WHERE preco BETWEEN 10 AND 50;
```

> *O comando DML a seguir está correto, pois a cláusula WHERE é opcional. `UPDATE ALUNOS SET Cidade='Curitiba', UF='PR';`*

Correto. Embora a cláusula `WHERE` seja opcional em um comando `UPDATE`, a ausência dela fará com que todos os registros da tabela sejam atualizados. Portanto, o comando que você forneceu atualiza a cidade e a UF de todos os alunos na tabela `ALUNOS` para 'Curitiba' e 'PR', respectivamente. Se a intenção era atualizar apenas alguns registros, a cláusula `WHERE` deve ser utilizada para especificar quais registros devem ser alterados.

> *A cláusula DELETE é considerada recurso DML.*

Correto, o comando `DELETE` é utilizado especificamente para remover registros de uma tabela.

> *A seguir, é apresentado o comando correto em SQL para se alterar o valor do salário (Salario) para o maior salário da tabela apenas para o funcionário que possui IdFuncionario igual a 1.*

```
UPDATE funcionario AS f, (SELECT max(salario) as maxsalario FROM
funcionario) AS s
SET f.salario = s.maxsalario
WHERE f.idfuncionario = 1;
```

Plain text

> *Os comandos `SELECT, SUM, FROM, > e AND` podem ser classificados, respectivamente, como de manipulação de dados, função de agregação, cláusula, operador relacional e operador lógico.*

1. **SELECT**: Comando de manipulação de dados (DML) - **Correto**.
2. **SUM**: Função de agregação - **Correto**.
3. **FROM**: Cláusula - **Correto**.
4. **AND**: Operador lógico - **Correto**.

Dados Estruturados e não Estruturados

Dados estruturados

> *Uma empresa está coletando os dados de campanhas sazonais de publicidade e mantém organizada uma planilha com as informações de cada campanha, com os respectivos períodos de duração e valores investidos. Nessa situação hipotética, a empresa possui dados do tipo* ***estruturado****.*

> *No que se refere a dados estruturados e não estruturados e a técnicas de integração e ingestão de dados, julgue os próximos itens. Dados estruturados normalmente são armazenados em arquivos dos tipos* `doc, wmv, mpw, mp3, wav, jpg`*; grandes volumes de dados estruturados são armazenados em data lakes, os quais podem ser acessados quando necessário.*

Errado. Dados estruturados são aqueles que têm um formato fixo e são organizados em tabelas, como em bancos de dados relacionais (por exemplo, arquivos CSV, tabelas SQL). Os tipos de arquivos mencionados (doc, wmv, mp3, jpg, etc.) são exemplos de dados não estruturados ou semi-estruturados. Além disso, grandes volumes de dados estruturados são geralmente armazenados em bancos de dados, enquanto data lakes são mais comumente usados para armazenar dados não estruturados e semi-estruturados.

> *Dados estruturados são dados que foram formatados e transformados para se adaptar a um modelo de dados relacional bem definido. Os dados brutos são mapeados em campos predefinidos que podem ser extraídos e lidos por meio de declarações SQL.*

> *Assinale a alternativa que* ***não*** *apresenta um dado não estruturado.*

- áudio gravado de uma conversa
- dados de logs de servidores ou sistemas de computadores
- comentário em rede social
- documento de texto sem formatação específica
- código de barras

Um código de barras é um dado estruturado, pois possui um formato específico e é utilizado para representar informações de maneira organizada, geralmente em um padrão que pode ser lido por scanners. As outras opções (áudio, logs, comentários em redes sociais e documentos de texto sem formatação específica) são exemplos de dados não estruturados, pois não seguem uma estrutura rígida e podem variar em formato e conteúdo.

> *Arquivos `.xml, .csv` e `.json` são exemplos de dados estruturados.*

Certo. Arquivos .xml, .csv e .json são, de fato, exemplos de dados estruturados.

- **.xml (Extensible Markup Language)** : Embora seja mais flexível e possa ser considerado semiestruturado em alguns contextos, ele é frequentemente usado para representar dados de forma estruturada, com uma hierarquia clara de elementos e atributos.
- **.csv (Comma-Separated Values)** : Este formato é um exemplo clássico de dados estruturados, pois organiza os dados em linhas e colunas, onde cada linha representa um registro e cada coluna representa um campo.
- **.json (JavaScript Object Notation)** : Assim como o XML, o JSON é frequentemente usado para representar dados de forma estruturada, com uma sintaxe que permite a organização de dados em pares de chave-valor.

> *A organização de dados estruturados é totalmente aberta.*

Errado.

A organização de dados estruturados não é totalmente aberta. Dados estruturados são organizados em um formato fixo e definido, geralmente em tabelas dentro de bancos de dados relacionais, onde cada coluna tem um tipo de dado específico e cada linha representa um registro. Essa estrutura rígida impõe regras sobre como os dados devem ser armazenados, acessados e manipulados.

Além disso, o acesso a dados estruturados geralmente é controlado por sistemas de gerenciamento de banco de dados (SGBDs), que definem permissões e restrições sobre quem pode visualizar ou modificar os dados. Portanto, a afirmação de que a organização de dados estruturados é totalmente aberta é incorreta.

Dados semi estruturados

> *Dados semiestruturados possuem características definidas, mas não seguem uma estrutura rígida, situando-se entre dados estruturados e não estruturados.*

Dados não estruturados

> *Uma empresa está coletando feedback de clientes sobre suas campanhas publicitárias por meio de comentários em redes sociais, e-mails e gravações de chamadas de atendimento ao cliente. Esses dados são armazenados em um repositório sem uma organização específica, contendo textos livres, opiniões e gravações de áudio. Nessa situação hipotética, a empresa possui dados do tipo* ***não estruturado****.*

> *Dados não estruturados não têm um formato específico e são frequentemente armazenados em sistemas de arquivos, com exemplos como imagens e sons.*

> *Se, em certa organização, grande parte dos dados for constituída de dados estruturados, mas também houver muitos dados de arquivos* `.pdf` *e* `.docx`*, então os dados, em geral, serão classificados como semiestruturados.*

Errado, os esses tipos de dados mencionados são dados não estruturados.

> *Dados de redes sociais são exemplos de dados não estruturados.*

Certo.

Dados de redes sociais são, em sua maioria, considerados exemplos de dados não estruturados. Isso se deve ao fato de que as informações postadas em redes sociais, como textos de comentários, postagens, imagens, vídeos e até mesmo interações (curtidas, compartilhamentos), não seguem uma estrutura rígida ou um formato predefinido.

Embora algumas partes dos dados de redes sociais possam ser organizadas (como dados de perfil que têm campos específicos), a maior parte do conteúdo gerado pelos usuários é não estruturada, pois pode variar amplamente em formato e conteúdo. Portanto, a afirmação está correta.

> *Dados não estruturados são aqueles que, desde a elaboração da estrutura, são pensados estritamente com uma finalidade.*

Dados não estruturados são aqueles que não têm uma organização ou formato fixo e não são pensados estritamente com uma finalidade específica desde a sua criação. Eles podem incluir uma variedade de formatos, como textos livres, imagens, vídeos, áudios e

outros tipos de conteúdo que não se encaixam em um modelo de dados rígido.

> *Assinale a alternativa que corresponde a um metadado não estruturado.*

- Data
- CPF
- Texto livre descritivo de uma foto em rede social
- Sistema de classificação de arquivos
- Dicionário de dados de um SGBD

A alternativa que corresponde a um metadado não estruturado é:

C: Texto livre descritivo de uma foto em rede social.

Um texto livre descritivo é considerado um metadado não estruturado porque não segue uma estrutura rígida e pode variar em formato e conteúdo. As outras opções (data, CPF, sistema de classificação de arquivos e dicionário de dados de um SGBD) são exemplos de metadados estruturados, pois têm formatos definidos e organizados.

Outros

> *Os metadados provêm uma descrição concisa dos dados e desempenham um papel na gestão dos dados; a partir dos metadados, as informações são processadas, atualizadas e consultadas.*

Certo. Os metadados fornecem uma descrição concisa dos dados, incluindo informações sobre a origem, estrutura, formato e contexto dos dados. Eles desempenham um papel fundamental na gestão dos dados, facilitando o processamento, a atualização e a consulta das informações.

> *Acerca do banco de dados MYSQL, julgue os seguintes itens.*

As contas dos usuários podem ter nomes com até ~~30~~ caracteres, serem criadas apenas via *CREATE USER*, removidas via *DROP USER* e terem senhas atribuídas via *DEFINE PASSWORD* .

A afirmação está **Errada**.

1. **As contas dos usuários podem ter nomes com até 30 caracteres**: Isso está incorreto. No MySQL, os nomes de usuários podem ter até 32 caracteres.
2. **Terem senhas atribuídas via DEFINE PASSWORD**: Isso está incorreto. O comando correto para definir ou alterar a senha de um usuário no MySQL é `SET PASSWORD` ou, em versões mais recentes, você pode usar `ALTER USER` para definir a senha.