semana 8- aula 01
Tipos de Modelagem de Dados
Modelagem de Dados Relacional: Entidades, Atributos e Relacionamentos
Código da aula: [SIS]ANO1C3B1S8A1

Objetivos da Aula:

- Conhecer os contextos básicos de modelagem de dados relacional.
- Conhecer frameworks de desenvolvimento ágeis, utilizando tecnologias de CI e CD que trabalham junto à segurança do ambiente funcional e entregas divididas em partes que agregam valor ao negócio de forma rápida; Trabalhar a resolução de problemas computacionais.

Exposição:

Modelagem de Banco de Dados Relacional: Uma Definição Objetiva com Exemplos Práticos

A modelagem de banco de dados relacional é o processo de estruturar e organizar os dados em um sistema de gerenciamento de banco de dados relacional (SGBDR). O objetivo principal é definir como os dados serão armazenados, acessados e relacionados, garantindo a integridade, consistência e eficiência do banco de dados.

Em essência, a modelagem relacional se baseia no conceito de relações (tabelas) que contêm tuplas (linhas ou registros) e atributos (colunas ou campos). As relações entre as tabelas são estabelecidas por meio de chaves primárias e chaves estrangeiras.

Exemplos Práticos:

1. Modelagem de um banco de dados para uma livraria:

- Entidades (Tabelas):
    - `Livros`: Contém informações sobre cada livro.
    - `Autores`: Contém informações sobre os autores.
    - `Editoras`: Contém informações sobre as editoras.
    - `Clientes`: Contém informações sobre os clientes.
    - `Pedidos`: Contém informações sobre os pedidos de compra.
    - `ItensPedido`: Contém os detalhes de cada livro em um pedido.
- Atributos (Colunas):

    - Livros: ID_Livro (chave primária), Titulo, ISBN, AnoPublicacao, Preco, ID_Autor (chave estrangeira referenciando Autores), ID_Editora (chave estrangeira referenciando Editoras).
    - Autores: ID_Autor (chave primária), Nome, Sobrenome.
    - Editoras: ID_Editora (chave primária), Nome.
    - Clientes: ID_Cliente (chave primária), Nome, Email, Endereco.
    - Pedidos: ID_Pedido (chave primária), ID_Cliente (chave estrangeira referenciando Clientes), DataPedido, ValorTotal.
    - ItensPedido: ID_ItemPedido (chave primária), ID_Pedido (chave estrangeira referenciando Pedidos), ID_Livro (chave estrangeira referenciando Livros), Quantidade, PrecoUnitario.
- Relacionamentos:
    - Um autor pode escrever vários livros (relação um-para-muitos entre Autores e Livros).
    - Uma editora pode publicar vários livros (relação um-para-muitos entre Editoras e Livros).
    - Um cliente pode fazer vários pedidos (relação um-para-muitos entre Clientes e Pedidos).
    - Um pedido pode conter vários livros (relação muitos-para-muitos entre Pedidos e Livros, resolvida pela tabela ItensPedido).

Esses exemplos ilustram como a modelagem de banco de dados relacional organiza informações complexas em estruturas lógicas e interconectadas, facilitando o armazenamento, a recuperação e a manipulação eficiente dos dados. O processo envolve a identificação de entidades, seus atributos e os relacionamentos entre elas, culminando em um esquema de banco de dados bem definido.

Uma entidade representa um objeto ou conceito do mundo real com uma existência independente. Por exemplo, em um banco de dados escolar, as entidades podem ser “Estudante”, “Professor” e “Curso”. Cada entidade é definida por um conjunto de atributos, que são as propriedades ou características que descrevem a entidade. Por exemplo, a entidade “Estudante” pode ter atributos como Nome, ID do Estudante, Curso e Data de Nascimento.

Chaves Primárias: É um atributo ou conjunto de atributos que identifica de forma única cada instância de uma entidade no banco de dados. Por exemplo, um número de ID do estudante pode atuar como a chave primária para a entidade “Estudante”. As propriedades que representam uma chave primária, além de únicas, precisam obrigatoriamente existir. Cada entidade só pode ter uma única chave primária.

Chaves Estrangeiras: É um atributo em uma tabela que é a chave primária de outra tabela. Ela estabelece uma relação entre duas tabelas, indicando como os dados de uma tabela se relacionam com os dados de outra.
Exemplo Tabela: Estudante ID do Estudante (Chave Primária), Nome, Data de Nascimento, ID do Curso (Chave Estrangeira)
A normalização de dados é um processo usado para organizar dados em um banco de dados. Ela envolve a criação de tabelas e a definição de relações de forma a minimizar a redundância e a dependência de dados. Esse processo é feito por meio da aplicação de regras conhecidas como formas normais.
Processo de Normalização: A normalização é realizada em várias etapas, cada uma garantindo que o banco de dados atenda a uma certa "forma normal" (1NF, 2NF, 3NF etc.). O objetivo é estruturar o banco de dados de forma que cada tabela tenha a mínima redundância possível e que as relações entre as tabelas sejam eficientes.
Minimizar Redundância: Reduzir os dados duplicados em várias tabelas para economizar espaço e prevenir inconsistências.
Minimizar Dependência: Organizar as tabelas de forma que as informações em uma tabela não dependam indevidamente das informações em outra.

semana 8- aula 02
Tipos de Modelagem de Dados
Modelagem de Dados Relacional: Entidades, Atributos e Relacionamentos
Código da aula: [SIS]ANO1C3B1S8A2

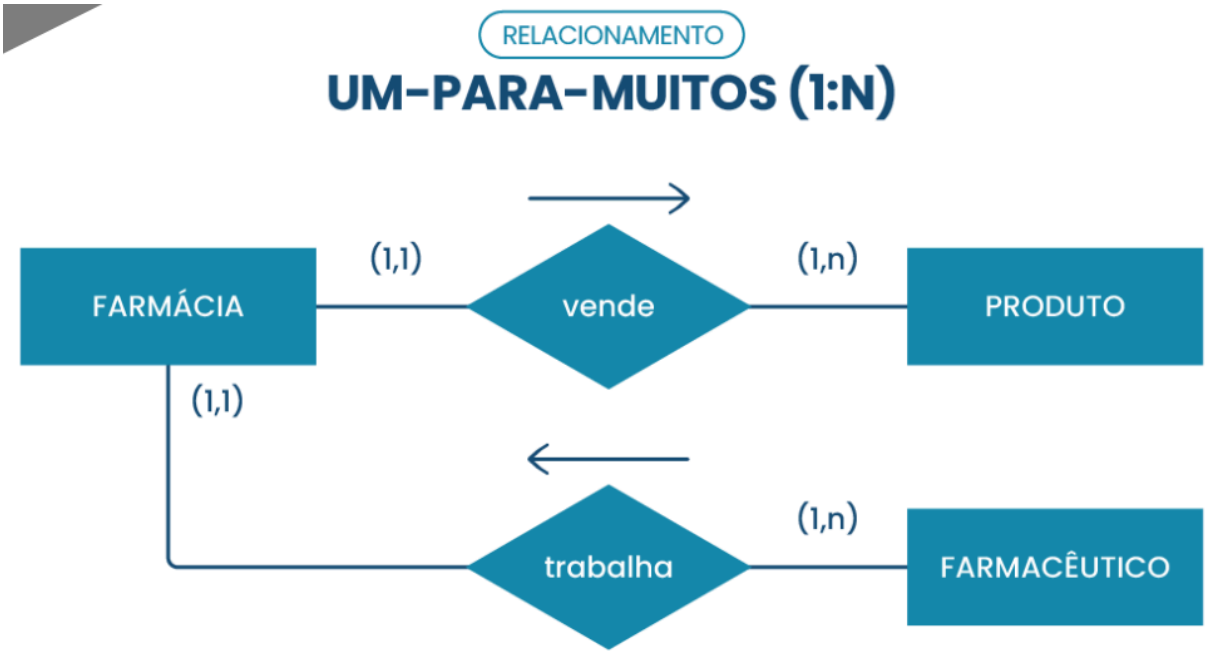
Objetivos da Aula:

- Conhecer os contextos básicos de modelagem de dados relacional.
- Conhecer frameworks de desenvolvimento ágeis, utilizando tecnologias de CI e CD que trabalham junto à segurança do ambiente funcional e entregas divididas em partes que agregam valor ao negócio de forma rápida; Trabalhar a resolução de problemas computacionais.

Exposição:
No modelo de Entidade-Relacionamento (MER), utilizado para representar estruturas de dados em bancos de dados relacionais, os relacionamentos entre entidades são categorizados em três tipos principais que veremos a seguir:
Relacionamento Um-para-Um (1:1): Esse tipo de relacionamento implica que uma entidade em uma tabela está associada a, no máximo, uma entidade em outra tabela. No MER, isso é representado por uma linha conectando duas entidades com uma marcação de '1' em ambas as extremidades. Por exemplo, considerando as

entidades ‘Pessoa’ e ‘Passaporte’, cada pessoa possui no máximo um passaporte, e cada passaporte está vinculado a uma única pessoa.

Relacionamento Um-para-Muitos (1:N): Esse relacionamento indica que uma entidade em uma tabela pode estar associada a várias entidades em outra tabela. No MER, isso é ilustrado por uma linha com uma marcação de ‘1’ em uma extremidade e ‘N’ na outra. Um exemplo clássico é a relação entre as entidades ‘Professor’ e ‘Curso’. Um professor pode ministrar vários cursos, mas cada curso é ministrado por apenas um professor.



Muitos-para-Muitos (N:N): Nesse relacionamento, linhas em uma tabela podem se relacionar com várias linhas em outra tabela e vice-versa. Um exemplo seria a relação entre Alunos e Cursos, em que alunos podem se matricular em vários cursos e cada curso pode ter vários alunos matriculados. Geralmente, é necessária uma tabela de associação para implementar esse tipo de relacionamento de forma eficaz.

A integridade referencial é um conceito fundamental em um banco de dados relacional que garante a validade dos relacionamentos entre tabelas. Isso é alcançado por meio de regras e restrições que mantêm a consistência dos dados. Por exemplo: Se uma tabela A tem uma chave estrangeira que aponta para uma tabela B, então todos os valores dessa chave estrangeira devem existir como um valor de chave primária na tabela B. Isso evita “órfãos”, ou seja, registros que apontam para um não existente registro relacionado. Ações como deletar ou alterar um registro em B devem considerar o relacionamento com A. Restrições de integridade referencial, como “cascade delete” ou “cascade update” podem ser aplicadas para manter a integridade dos dados.
Na modelagem de dados usando o Modelo de EntidadeRelacionamento (MER), representamos as entidades e seus relacionamentos, bem como os atributos dessas relações. As associações entre entidades podem variar em complexidade, sendo particularmente interessantes no caso de relacionamentos muitos-para-muitos que possuem atributos próprios. Exemplo com Alunos, Cursos e Matrículas (Entidades): Aluno: Uma entidade que representa os alunos. Possui atributos como ID do Aluno, Nome etc. Curso: Uma entidade que representa os cursos oferecidos. Inclui atributos como ID do Curso, Nome do Curso etc.
A integridade referencial é uma propriedade dos bancos de dados relacionais que garante a validade das relações entre as tabelas. Em termos simples, ela assegura que as referências entre os dados nas diferentes tabelas permaneçam consistentes e corretas ao longo do tempo.

Essa integridade é mantida através do uso de chaves estrangeiras. Uma chave estrangeira em uma tabela referencia a chave primária de outra tabela. A integridade referencial impõe regras sobre como os valores dessas chaves estrangeiras podem ser manipulados, prevenindo a criação de "registros órfãos" ou referências inválidas.

Em essência, a integridade referencial garante que:

- Todo valor de chave estrangeira em uma tabela corresponda a um valor existente na chave primária da tabela referenciada.
- As ações que possam comprometer essa correspondência (como excluir um registro referenciado sem tratar as chaves estrangeiras) sejam restritas ou gerenciadas de forma controlada pelo SGBDR.

Exemplos Práticos:

1. Banco de Dados de Livraria (Continuando o exemplo anterior):

- Temos as tabelas `Livros` e `Autores`.
- A tabela `Livros` possui uma coluna `ID_Autor` que é uma chave estrangeira, referenciando a coluna `ID_Autor` (chave primária) da tabela `Autores`.

Regras de Integridade Referencial neste cenário:

- Inserção: Não é possível inserir um novo livro na tabela `Livros` com um valor de `ID_Autor` que não exista na coluna `ID_Autor` da tabela `Autores`. Isso garante que todo livro esteja associado a um autor válido.
    - Exemplo de violação: Tentar inserir um livro com `ID_Autor = 99`, se não houver nenhum autor com `ID_Autor = 99` na tabela `Autores`, resultaria em um erro de integridade referencial.
- Exclusão: Ao tentar excluir um autor da tabela `Autores`, o SGBDR pode implementar diferentes comportamentos para manter a integridade:
    - Restringir (RESTRICT): A exclusão do autor é impedida se existirem livros na tabela `Livros` que referenciam esse autor (ou seja, possuem o `ID_Autor` do autor a ser excluído).
        - Exemplo: Se o autor com `ID_Autor = 1` (digamos, "Machado de Assis") tiver livros cadastrados na tabela `Livros` com `ID_Autor = 1`, o sistema impediria a exclusão de Machado de Assis até que esses livros fossem removidos ou tivessem seu `ID_Autor` alterado.
    - Cascata (CASCADE): A exclusão do autor na tabela `Autores` automaticamente excluiria todos os livros correspondentes na tabela `Livros`.
        - Exemplo: Se o autor com `ID_Autor = 2` tiver vários livros cadastrados, ao excluir esse autor, todos os seus livros seriam automaticamente removidos da tabela `Livros`.
    - Definir Nulo (SET NULL): Ao excluir o autor na tabela `Autores`, a coluna `ID_Autor` nos livros correspondentes na tabela `Livros` seria automaticamente definida como `NULL`. Isso indica que o livro não está mais associado a um autor específico (pode ser usado em cenários onde a autoria é opcional ou desconhecida após a exclusão).
        - Exemplo: Ao excluir um autor, os livros que tinham seu `ID_Autor` agora teriam o valor `NULL` na coluna `ID_Autor`.
    - Definir Valor Padrão (SET DEFAULT): Similar ao `SET NULL`, mas define a chave estrangeira com um valor padrão pré-definido.
- Atualização: Ao tentar atualizar o valor da chave primária (`ID_Autor`) na tabela `Autores`, o SGBDR pode implementar comportamentos semelhantes aos da

exclusão (restringir, cascata, definir nulo ou padrão) para garantir que as referências na tabela `Livros` permaneçam válidas.

2. Banco de Dados de Pedidos (Continuando o exemplo da livraria):

- Temos as tabelas `Pedidos` e `Clientes`.
- A tabela `Pedidos` possui uma coluna `ID_Cliente` como chave estrangeira, referenciando a coluna `ID_Cliente` (chave primária) da tabela `Clientes`.

Regras de Integridade Referencial neste cenário:

- Todo pedido registrado na tabela `Pedidos` deve estar associado a um cliente existente na tabela `Clientes`.
- Ao excluir um cliente da tabela `Clientes`, as ações de integridade referencial (restringir, cascata, definir nulo ou padrão) definiriam como os pedidos associados a esse cliente seriam tratados.

Importância da Integridade Referencial:

- Consistência dos Dados: Garante que os dados relacionados entre as tabelas permaneçam coerentes.
- Prevenção de Erros: Evita a criação de informações inconsistentes e a perda de relacionamentos importantes.
- Confiabilidade do Banco de Dados: Aumenta a confiança na precisão e validade dos dados armazenados.
- Facilita a Manutenção: Simplifica a manutenção e a evolução do banco de dados, pois as relações entre os dados são bem definidas e protegidas.

Em resumo, a integridade referencial é um mecanismo crucial para manter a qualidade e a confiabilidade dos bancos de dados relacionais, assegurando que os laços entre as informações armazenadas sejam sempre válidos e significativos.

semana 8- aula 03
Tipos de Modelagem de Dados
Modelagem de Dados Relacional: Entidades, Atributos e Relacionamentos
Código da aula: [SIS]ANO1C3B1S8A3

Objetivos da Aula:

- Conhecer os contextos básicos de modelagem de dados relacional.
- Conhecer frameworks de desenvolvimento ágeis, utilizando tecnologias de CI e CD que trabalham junto à segurança do ambiente funcional e entregas divididas em partes que agregam valor ao negócio de forma rápida; Trabalhar a resolução de problemas computacionais.

Exposição:

Um Diagrama de Entidade-Relacionamento (DER) é uma representação visual utilizada na modelagem de dados para descrever a estrutura de um banco de dados. Ele ilustra as entidades (objetos ou conceitos importantes), seus atributos (características dessas entidades) e os relacionamentos entre essas entidades. O objetivo principal de um DER é fornecer um modelo conceitual claro e compreensível da organização dos dados antes da implementação física do banco de dados.

Componentes Principais de um DER:

1. Entidades: Representadas geralmente por retângulos. Uma entidade é algo sobre o qual se deseja armazenar informações.
    - Exemplo: Em um sistema de biblioteca, `Livro`, `Autor` e `Cliente` seriam entidades.
2. Atributos: Representados geralmente por elipses ligadas às entidades. Um atributo descreve uma propriedade ou característica de uma entidade.
    - Exemplo (para a entidade `Livro`): `Título`, `ISBN`, `AnoPublicacao`, `Preco`.
    - Exemplo (para a entidade `Autor`): `Nome`, `Sobrenome`, `Nacionalidade`.
3. Relacionamentos: Representados geralmente por losangos ligados às entidades que se relacionam. Um relacionamento descreve como as entidades interagem entre si. A cardinalidade é frequentemente especificada nas linhas que conectam o losango às entidades, indicando quantos экземпляра de uma entidade podem estar relacionados a quantos экземпляра de outra.
    - Cardinalidade:

- Um para Um (1:1): Uma instância de uma entidade se relaciona com no máximo uma instância de outra entidade, e vice-versa.
    - Exemplo: Uma pessoa pode ter apenas um número de CPF, e cada número de CPF pertence a apenas uma pessoa.
- Um para Muitos (1:N): Uma instância de uma entidade pode se relacionar com várias instâncias de outra entidade, mas cada instância da segunda entidade se relaciona com apenas uma instância da primeira.
    - Exemplo: Um autor pode escrever vários livros, mas cada livro é escrito por um único autor (considerando um autor principal).
- Muitos para Muitos (M:N): Várias instâncias de uma entidade podem se relacionar com várias instâncias de outra entidade.
    - Exemplo: Um aluno pode se matricular em várias disciplinas, e uma disciplina pode ter vários alunos matriculados.

Exemplo Prático de um DER Simplificado para um Sistema de Biblioteca:

Snippet de código

```
erDiagram
    LIVRO ||--o{ AUTOR : escreve
    LIVRO {
        VARCHAR Titulo
        VARCHAR ISBN PK
        INTEGER AnoPublicacao
        DECIMAL Preco
        INTEGER ID_Autor FK
    }
    AUTOR {
        INTEGER ID_Autor PK
        VARCHAR Nome
        VARCHAR Sobrenome
        VARCHAR Nacionalidade
    }
    CLIENTE ||--o{ EMPRESTIMO : pega emprestado
    LIVRO ||--o{ EMPRESTIMO : é emprestado
    CLIENTE {
        INTEGER ID_Cliente PK
        VARCHAR Nome
        VARCHAR Endereco
        VARCHAR Telefone
    }
```

```
EMPRESTIMO {
    INTEGER ID_Emprestimo PK
    INTEGER ID_Cliente FK
    VARCHAR ISBN_Livro FK
    DATE DataEmprestimo
    DATE DataDevolucaoPrevista
    DATE DataDevolucaoReal
}
```

Explicação do Exemplo:

- Entidades: `LIVRO`, `AUTOR`, `CLIENTE`, `EMPRESTIMO`.
- Atributos (alguns exemplos):
    - `LIVRO`: `Titulo`, `ISBN` (chave primária), `AnoPublicacao`, `Preco`, `ID_Autor` (chave estrangeira referenciando `AUTOR`).
    - `AUTOR`: `ID_Autor` (chave primária), `Nome`, `Sobrenome`, `Nacionalidade`.
    - `CLIENTE`: `ID_Cliente` (chave primária), `Nome`, `Endereco`, `Telefone`.
    - `EMPRESTIMO`: `ID_Emprestimo` (chave primária), `ID_Cliente` (chave estrangeira referenciando `CLIENTE`), `ISBN_Livro` (chave estrangeira referenciando `LIVRO`), `DataEmprestimo`, `DataDevolucaoPrevista`, `DataDevolucaoReal`.
- Relacionamentos:
    - Um `AUTOR` escreve muitos `LIVRO`s (1:N).
    - Um `CLIENTE` pega emprestado muitos `LIVRO`s (M:N, resolvido pela entidade associativa `EMPRESTIMO`). Um `LIVRO` pode ser emprestado por muitos `CLIENTE`s. A entidade `EMPRESTIMO` registra cada instância desse relacionamento, incluindo a data do empréstimo e a data de devolução.

Em resumo, um DER é uma ferramenta essencial para visualizar e comunicar a estrutura lógica de um banco de dados, facilitando o entendimento dos dados, suas propriedades e como eles se relacionam antes da criação física do banco de dados. Ele ajuda a garantir que o banco de dados atenda aos requisitos do sistema e seja bem organizado.

A desnormalização é o processo inverso da normalização e pode ser usada para melhorar o desempenho do banco de dados em situações específicas. Enquanto a normalização separa os dados em tabelas diferentes para reduzir a redundância, a desnormalização reintroduz deliberadamente essa redundância para acelerar as operações de leitura.
Quando aplicar: • Quando as consultas estão se tornando complexas e envolvem muitas junções que degradam o desempenho; • Em sistemas em que as operações de leitura são muito mais frequentes que as operações de escrita; • Em situações

em que os dados não mudam frequentemente, portanto, o custo de manter a redundância é menor que o benefício do aumento de velocidade nas consultas.
Como aplicar: • Combine tabelas que são frequentemente usadas juntas para evitar junções; • Adicione colunas redundantes em que o acesso rápido aos dados é crítico, como o total pré-calculado de vendas em uma tabela de pedidos; • Use a desnormalização para préagregar dados, como somas e médias, para relatórios e dashboards.
Indexação e performance: Índices são estruturas de dados especiais que os bancos de dados usam para acelerar a recuperação de registros. Atuam como "sumários" de um livro para dados; eles permitem que o sistema de banco de dados encontre dados rapidamente sem ter que procurar em toda a tabela a cada consulta.
Utilização de índices: • Crie índices em colunas que são frequentemente usadas como critérios de busca em suas consultas; • Índices são particularmente úteis em colunas utilizadas para junções, cláusulas WHERE, ORDER BY e GROUP BY; • Tenha cuidado com a indexação excessiva, pois isso pode tornar as operações de inserção, atualização e exclusão mais lentas, já que os índices também precisam ser atualizados.

Modelagem de dados para Big Data

A modelagem de dados para Big Data é o processo de estruturar, organizar e representar grandes volumes de dados complexos e variados, com o objetivo de torná-los adequados para análise, interpretação e tomada de decisões. Diferente da modelagem tradicional de bancos de dados relacionais, a modelagem para Big Data precisa lidar com características como volume massivo, alta velocidade de geração, variedade de formatos (estruturados, semiestruturados e não estruturados) e veracidade (qualidade e confiabilidade dos dados).

O objetivo principal da modelagem de dados para Big Data é otimizar o armazenamento, o processamento e a análise eficiente desses grandes conjuntos de dados, considerando as limitações e capacidades das tecnologias de Big Data.

Principais Diferenças e Desafios em Relação à Modelagem Tradicional:

- Escalabilidade: Os modelos devem ser capazes de escalar horizontalmente para acomodar o crescimento exponencial dos dados.
- Flexibilidade: A variedade dos dados exige modelos mais flexíveis que possam acomodar diferentes formatos sem a rigidez de um esquema relacional predefinido.
- Desempenho: O foco está em otimizar o processamento paralelo e distribuído para realizar análises em tempo hábil.
- Tolerância a Falhas: Os modelos e os sistemas subjacentes devem ser resilientes a falhas em nós de processamento ou armazenamento.
- Esquema-on-Read vs. Esquema-on-Write: Enquanto os bancos de dados relacionais tradicionais geralmente seguem um modelo "esquema-on-write" (o esquema é definido antes da inserção dos dados), as abordagens de Big

Data frequentemente utilizam um modelo "esquema-on-read" (o esquema é aplicado durante a análise).

Exemplos de Abordagens e Modelos em Big Data:

- Data Lakes: Um modelo de armazenamento centralizado que permite armazenar dados em seu formato nativo, sejam eles estruturados, semiestruturados ou não estruturados. O esquema é aplicado quando os dados são acessados para análise.
    - Exemplo: Armazenar logs de servidores web, dados de sensores IoT, posts de redes sociais e dados transacionais em um único repositório, sem a necessidade de transformá-los em um esquema relacional rígido antecipadamente.
- NoSQL Databases: Uma ampla categoria de bancos de dados que não seguem o modelo relacional tradicional. Eles oferecem diferentes modelos de dados otimizados para escalabilidade e flexibilidade.
    - Bancos de Dados Chave-Valor (Key-Value Stores): Armazenam dados como pares chave-valor.
        - Exemplo: Redis, Cassandra (em alguns usos).
    - Bancos de Dados de Documentos: Armazenam dados como documentos JSON ou XML.
        - Exemplo: MongoDB, Couchbase.
    - Bancos de Dados de Colunas Familiares (Column-Family Stores): Organizam dados em colunas que podem ser agrupadas em famílias de colunas.
        - Exemplo: HBase, Cassandra.
    - Bancos de Dados de Grafos: Modelam dados como nós e arestas para representar relacionamentos complexos.
        - Exemplo: Neo4j.
- Modelos Dimensionais para Big Data: Adaptações de modelos dimensionais (como star schema ou snowflake schema) para ambientes de Big Data, frequentemente implementados em plataformas como Hadoop ou Spark utilizando ferramentas como Hive ou Spark SQL.
    - Exemplo: Criar uma estrutura dimensional com tabelas de fatos (por exemplo, eventos de clique em um site) e tabelas de dimensões (por exemplo, informações do usuário, detalhes do produto, data) armazenadas em formato columnar para otimizar consultas analíticas.
- Grafos de Conhecimento: Modelos que representam o conhecimento como um grafo, onde nós representam entidades e arestas representam as relações entre elas. Útil para descobrir padrões e inferências em dados complexos e interconectados.
    - Exemplo: Modelar relacionamentos entre clientes, produtos, categorias e interações para sistemas de recomendação ou análise de redes sociais.

Em resumo, a modelagem de dados para Big Data exige uma mudança de paradigma em relação à modelagem tradicional, priorizando a escalabilidade, a flexibilidade e o desempenho no processamento de grandes volumes de dados variados. A escolha do modelo depende dos tipos de dados, dos requisitos de análise e das tecnologias utilizadas no ambiente de Big Data.