Banco de Dados

Introdução

Um banco de dados é uma coleção de dados relacionados entre si, organizada e armazenada de forma a possibilitar fácil manipulação e controle de informações, minimizar incoerências, evitar redundâncias e representar relacionamentos complexos de dados.

Modelagem de dados

É a representação dos elementos do mundo real no banco de dados, independentes do SGBD que será utilizado, depois da ser feita coleta de informações dos dados que vão ser armazenados, a modelagem representa as tabelas de dados e os relacionamentos entre elas de forma visual.

Mapeamento dos Dados

Depois que os dados são coletados eles são organizados para melhorar os relacionamentos entre os dados e evitar redundância, isso é conhecido como mapeamento.

O mapeamento é feito seguindo 10 regras simples:

1. Criar uma tabela para cada entidade

    - Entidades são dados que sentido estarem juntos, por exemplo:

        Usuario (**CPF**, nome, data_nascimento, sexo, salario)

2. Criar uma tabela para cada atributo multivalorado

    - Cria-se uma tabela para atributos que tenham vários valores, por exemplo:

        Departamento (**nro_dept**, nome, salas)

Departamento (**nro_dept**, nome)

Salas_dept (**nro_dept**, sala)

3. Substituir atributos compostos por suas partes

    - Atributos compostos são informações que podem ser divididas em informações menores, como endereços por exemplo

4. Identificar entidades fracas

    - Entidades fracas são entidades que dependem de outra, como por exemplo uma tabela que armazena informações de dependentes de um funcionário, neste caso a entidade fraca recebe a chave da sua entidade forte como sua chave primária

5. Identificar relacionamentos 1:1

    - Identificar relacionamentos entre tabelas que são únicos, como por exemplo, um único departamento tem um único gerente, e passar a chave e os atributos do relacionamento para a tabela que logicamente os deve recebê-los.
        - Depto = (**nro_depto**, nome, **id_ger**[CE] , dta_início)

6. Identificar relacionamentos 1:N

    - Identificar relacionamentos um-para-muitos, onde uma entidade pode aparecer em vários relacionamentos com outras, neste caso a chave do lado 1 é passada para a tabela do lado N. Por exemplo, um departamento pode ter vários projetos por vez, e um projeto só tem um departamento, então a chave da tabela Departamento é passada para a tabela Projetos como chave estrangeira.
        - Projeto = (**nro_proj**, nome, duração, orçamento, **nro_depto**[ce])

7. Identificar relacionamentos M:N

    - Identificar relacionamentos muitos-para-muitos, onde muitos registros podem se relacionar com muitos outros registros. Neste caso se cria uma nova tabela que contém as chaves das duas tabelas e os atributos que distinguem o relacionamento.

8. Identificar relacionamentos ternários ou superiores

    - Mesmo processo dos relacionamentos M:N

9. Identificar agregações

    - Mesmo processo dos relacionamentos M:N

10. Identificar generalizações

    - Generalizações ocorrem quando é necessário distinguir atributos de uma tabela, como por exemplo distinguir tipos de funcionários, ou tipos de clientes.
        - Geralmente se cria uma tabela com os atributos gerais e uma tabela para cada generalização, as novas tabelas são entidades fracas da tabela geral, por isso tem a mesma chave.
        - Também podemos criar tabelas separadas para cada generalização contendo todos os atributos mais os atributos da generalização.
        - E por último podemos criar uma tabela que contenha todos os atributos de todas as generalizações e preencher somente os que vão ser utilizados para cada caso.

Normalização de Dados

A Normalização de dados garante a consistência e impede a redundância de dados, e é facilitada por um bom mapeamento dos dados.

A normalização é feita seguindo alguns passos:

1. Identificar possíveis chaves primárias.

    - Identificar atributos que vão identificar, sem erros, a linha da tabela.
        - Ex. Número de Matrícula, Documentos, Nomes, etc.

2. Escolher uma chave primária ou superchave

    - Dentre os atributos identificados no passo 1, é escolhido um ou mais atributos para formar uma chave primária única.
        - Obs. Evitar utilizar documentos pessoais (RG, CPF, Passaporte, etc.) como chaves, pois apesar de serem excelentes chaves candidatas, não são indicadas pois contém informações pessoais.

3. Verificar restrições de unicidade

    - Verificar se a/as chaves escolhidas não tem possibilidade de terem valores duplicados, se houver a chave deverá ser trocada.
    - Verificar se a/as chaves tem possibilidade de ter um valor nulo, se houver a chave deverá ser trocada

4. Verificar restrições de integridade

    - Integridade Referencial: Se um registro faz referência a uma outra tabela através de uma chave estrangeira, deve-se verificar se o registro existe na outra tabela em primeiro lugar.
    - Integridade Semântica: Verificar se os registros inseridos conferem com o formato do campo no banco de dados e se os dados fazem sentido. Ex: Formato de um campo *datetime* é hh:mm:ss yyyy/mm/dd, ou um campo salário não pode ser negativo.

5. Verificar dependências funcionais

    - Verificar se os atributos dependem da chave escolhida. Se todos os atributos dependem da chave escolhida é dito uma dependência total, caso contrário ela é parcial

6. Aplicar as formas normais

    - As Formas normais validam os esquemas criados pelas regras anteriores.
    - 1ª Forma Normal
        - Se todos os atributos têm obrigatoriamente valores atômicos, se não, a regra 2 do mapeamento não foi seguida e o mapeamento precisa ser refeito.

    - 2ª Forma Normal
        - Se todos os atributos são dependentes ou parcialmente dependentes da chave escolhida
    - 3ª Forma Normal
        - Se existe algum atributo que não é dependente da chave escolhida e em vez disso é dependente de algum outro atributo da entidade
    - Forma normal de Boyce-Codd
        - Se existe algum atributo que não é dependente da chave escolhida (3ª Forma Normal), então o atributo que é "dependido" vira parte da chave, criando uma superchave.
    - 4ª Forma Normal
        - Existe uma dependência multivalorada se um valor X é associado com uma coleção de valores Y, como um professor estar apto a ministrar várias disciplinas, isso é um relacionamento M:N e deve ter sido resolvido na regra 7 do mapeamento

SQL

SQL, ou *Structured Query Language,* é a linguagem utilizada na criação e manipulação de banco de dados. A linguagem utiliza comandos em inglês simples e não é *case sensitive.*

Conceitos chave

- **Tabelas:** A estrutura primária para organizar dados em bancos de dados SQL. As tabelas consistem em linhas (registros) e colunas (campos) que definem a estrutura dos dados.
- **Chaves primárias:** Identificadores exclusivos para cada registro em uma tabela. Elas garantem a integridade dos dados e fornecem uma maneira de referenciar linhas específicas.
- **Chaves estrangeiras:** Campos em uma tabela que se refere a chaves primárias em outra tabela. Elas estabelecem relacionamentos entre tabelas, permitindo a normalização de dados e mantendo a integridade referencial.
- **Índices:** Estruturas de dados que melhoram a velocidade das operações de recuperação de dados em tabelas de banco de dados. Embora possam melhorar significativamente o desempenho da consulta, também exigem armazenamento adicional e podem tornar mais lenta a inserção e as atualizações de dados.
- **Visualizações:** Tabelas virtuais criadas por uma consulta, combinando dados de uma ou mais tabelas. As visualizações podem simplificar consultas complexas e fornecer uma camada adicional de segurança restringindo o acesso às tabelas subjacentes.
- **Normalização:** O processo de organizar dados para minimizar redundância e dependência. Normalmente envolve dividir tabelas grandes em tabelas menores e mais gerenciáveis e definir relacionamentos entre elas.
- **Gatilhos:** Procedimentos armazenados especiais que são executados automaticamente em resposta a certos eventos em uma tabela ou exibição específica. Eles são usados para impor regras de integridade complexas e automatizar certas tarefas.
- **Procedimentos armazenados:** Instruções SQL pré-compiladas armazenadas no banco de dados. Eles podem aceitar parâmetros, executar operações complexas e retornar resultados, melhorando o desempenho e a reutilização do código.

- **Transações:** Sequências de operações de banco de dados que são tratadas como uma única unidade de trabalho. Eles garantem a consistência e a integridade dos dados, especialmente em ambientes multi-usuários.

Criação

- **CREATE:** Usado para criar novos objetos de banco de dados, como tabelas, índices ou visualizações.
    - CREATE TABLE Users (id INT PRIMARY KEY, username VARCHAR(50), age INT);
- **ALTER:** Permite a modificação de objetos de banco de dados existentes. Isso pode incluir adicionar, modificar ou remover colunas em uma tabela.
    - ALTER TABLE Users ADD COLUMN email VARCHAR(100);
- **DROP:** Remove um objeto de banco de dados existente. Este comando deve ser usado com cautela, pois exclui permanentemente o objeto e seus dados:
    - DROP TABLE Users;
- **RENAME:** Altera o nome de um objeto existente no banco de dados.
    - RENAME TABLE OldTableName TO NewTableName;
- **COMMENT:** Adiciona comentários ao dicionário de dados. Isso é útil para fins de documentação.
- **TRUNCATE:** Remove todos os registros de uma tabela, incluindo todos os espaços alocados para os registros. Ao contrário de DELETE, TRUNCATE é mais rápido e não gera logs de exclusão de linhas individuais.

Manipulação

- **INSERT:** Usado para adicionar novos registros em uma tabela.
    - INSERT INTO Users (nome de usuário, idade) VALUES ('JohnDoe', 30);
- **UPDATE:** Modifica registros existentes em uma tabela. Pode ser usado para alterar um ou mais valores de coluna:
    - UPDATE Users SET age = 31 WHERE username = 'JohnDoe';
- **DELETE:** Remove registros específicos de uma tabela com base em uma condição:
    - DELETE FROM Users WHERE age 25;

Consulta

- **SELECT**: Usado para selecionar registros do banco de dados em uma ou mais tabelas
    - SELECT campo_tabela[1] FROM nome_tabela;
        - 1 - Pode ser definido mais de um campo utilizando vírgulas ou todos os campos da tabela utilizando um asterisco (*)
- **WHERE:** Usado para adicionar uma condição em comandos ALTER, DROP, UPDATE, DELETE e SELECT
    - SELECT * FROM tabela WHERE id_tabela=1;
- **JOIN**: Utilizado dentro do SELECT para unir os resultados da consulta de duas ou mais tabelas
    - SELECT campos_tabela FROM tabela1

JOIN tabela2 ON chave_estrangeira_tabela2

WHERE tabela1.campo = tabela2.campo;

- Existem 5 tipos de junção: FULL, INNER, OUTER, LEFT e RIGHT;
    - FULL resultados de todas as tabelas
    - INNER resulta somente nos campos que conectam as duas tabelas
    - OUTER resulta em todos as linhas que correspondem nas duas tabelas
    - LEFT retorna todos as linhas que correspondem na primeira tabela da junção
    - RIGHT retorna todas as linhas que correspondem na segunda tabela da junção