Base Dados

Sistemas de Gestão de Base de Dados (SBGD)

Software que facilita os processos de definição, construção, manipulação e compartilhamento de base de dados entre vários utilizadores e aplicações.
4 etapas: Definição, Construção, Manipulação, Partilha

Vantagens	Desvantagens
- Independência entre programas e dados - Integridade dos dados - Isolamento de utilizadores - Mecanismos de backup e recuperação de dados	- Elevados custos e complexidade na instalação e manutenção.

Diagrama Entidade-Relação (DER)

Entidades ➡ **Fortes** - Não dependem de outras entidades
➡ **Fracas** - Dependem de outras entidades

Atributos ➡ **Derivados** - Pode ser determinado a partir de outros atributos (ex: idade)
➡ **Multivalor** - Pode tomar um ou mais valores para cada entidade (ex: nfilhos)
➡ **Compostos** - Divisel em atributos mais simples com significado independente (ex: morada)

Obrigatoriedade ➡ **Participação Total (Obrigatório)** - Cada instância da entidade participa em pelo menos uma relação do conjunto das relações (linha dupla)
➡ **Participação Parcial (Opcional)** - Alguma(s) instância(s) da entidade podem não participar obrigatoriamente em qualquer relação do conjunto de relações

Cardinalidade - Relação entre o número de ocorrências numa entidade com as respetivas ocorrências na outra com que tem o relacionamento.

- ➡ **1:1** - Um funcionário gere um departamento e um departamento só tem um gestor
- ➡ **1:N** - Um funcionário trabalha para vários departamentos
- ➡ **N:M** - Vários funcionários trabalham em vários projetos

Restrições	
Sobreposição (Overlaping)	**Completude (Covering)**
- disjuntas: Uma entidade só pode pertence, no máximo, a uma subclasse de especialização. Exemplo: Uma pessoa ou é professor ou estudante **- sobrepostas:** Uma ocorrência de entidade genérico pode ter mais de uma especialização. Exemplo: Uma pessoa é tanto professor como estudante.	**- total:** Uma entidade de nível superior tem de pertencer a pelo menos uma subclasse de especialização (linha dupla). Exemplo: Toda a pessoa é um professor ou um estudante - parcial: Pode não pertencer a nenhum Exemplo: Nem toda a pessoa é um professor ou um estudante

Modelo Relacional

Baseado no conceito de “Relação”, representado por Tabelas
- Dispõe de um sistema formal de manipulação das relações (algebra relacional)
- Uma tabela de valores pode ser vista como um conjunto de linhas (tuplos)
- O número de atributos de uma relação define o grau da relação
- No modelo relacional os atributos não podem ser do tipo composto ou multivalor
- A cardinalidade da relação é obtida pelo número de tuplos
- Esquema da relação: Nome do esquema + Lista atributos (Exemplo: Pessoa(nome,bi,idade))

Restrições de Integridade - Regras que visam garantir a integridade dos dados (devem ser garantidos pelo próprio SGBD).
- Integridade do domínio dos atributos - Forma mais elementar de integridade. Os campos devem obedecer ao tipo de dados e às restrições de valores admitidos para um atributo.
- Integridade de entidade - Cada tuplo deve ser identificado de forma única com recurso a uma chave primária que não se repete e não pode ser nula
- Integridade referencial - O valor de uma chave estrangueira ou é null ou contém um valor que é chave primária na relação de onde foi importada.

Linguagem SQL - View

- Uma view pode ser utilizado como fonte de dados (idêntico a uma tabela normal) num conjunto de operações
- As vistas podem ser consideradas como tabelas virtuais. Regra geral, tem um conjunto definições e armazena fisicamente os dados. Uma vista também tem um conjunto de definições, que são criados sobre tabela(s) ou outra(s) vista(s) e não armazena fisicamente os dados.

Normalização

Preservação da informação - Todos os conceitos capturados pelo desenho conceptual que são mais tarde mapeados para desenho lógico.
Minimizar a redundância dos dados - Minimizar o armazenamento de dados duplicados em relações distintas, reduzindo a necessidade de múltiplos updates e consequentemente problemas de consistência entre múltiplas cópias da mesma informação.

Redução de NULLS nos tuplos - Há situações em que temos uma grande quantidade de atributos numa relação: muitos dos atributos não se aplicam a todos os tuplos da relação.
consequência: desperdício de espaço, difícil interpretação dos atributos nulo
recomendação: criar outra relação para esses atributos

Tipos de Dependências Funcionais
- Dependência parcial - Atributo depende de parte dos atributos que compõem a chave da relação.
- Dependência total - Atributo depende de toda a chave da relação
- Dependência transitiva - Atributo que não faz parte da chave da relação depende de um atributo que também não faz parte da chave da relação.

Formas Normais
- 1ª Forma Normal - Uma relação está na 1FN se:
  - Cada atributo contém apenas valores atómicos (simples e indivisíveis - não é permitido atributos compostos ou multi-valor)
  - Não suporta relações dentro de relações - não é possível utilizar uma relação como valor de um atributo de um tuplo.

- 2ª Forma Normal - Uma relação está na 2FN se a relação estiver na 1FN e…
  - Os atributos que não são chave dependem da totalidade da chave, isto é, não existem dependências parciais.

- 3ª Forma Normal - Uma relação está na 3FN se a relação estiver na 2FN e…
  - Não existem dependências funcionais entre atributos não chave, isto é, não existem dependências transitivas.

- BCNF - Usualmente, a 3FN é aquela que termina o processo de normalização. No entanto, em algumas situações a 3FN ainda apresenta algumas anomalias. Todos os atributos são funcionalmente dependentes da chave da relação, de toda a chave e de nada mais.

Indexação e Otimização

Índices são estruturas de dados que oferecem uma segunda forma (rápida) de acesso aos dados.
- Melhora o tempo de consulta - crítico para o desempenho da BD
- Pode aumentar o volume dos dados armazenados (overhead) e o tempo das inserções.

É possível:
- Indexar qualquer atributo da relação
- Criar multiplos índices (sobre atributos distintos)
- Criar indices com vários atributos

Os índices são estruturas que têm um valor ordenado (atributo indexado)

Single-Level Ordered (complexidade: log2 - bi)
São estruturas de um único nível que indexam um atributo da relação.
- armazena cada valor do atríbuto indexado e a respetiva localização da relação (ponteiro para a estrutra física que suporta a tabela)
- indices são ordenados o que permite pesquisa binária sobre o atributo

Single-Level Index - Tipos

Primary Index - Indexa um atributo chave da relação (não se repete).
- Tuplos armazenados em blocos (páginas) de tamanho fixo.

Clustered Index - Indexa um atributo que pode ter valores duplicados
- Os atributos estão agrupados

Secundary Index - Indexa outros atributos (chave candidata ou não chave)
- Podemos ter vários índices deste tipo

➡ Só podemos ter um Primary ou Clustered

Multilevel Index
A ideia do multilevel é ter vários níveis de indexação passando a complexidade para logfo bi
Estes índices são tipicamente implementados com estruturas em árvores balanceadas
- B-Tree

Árvore de Pesquisa Balanceada
Uma árvore diz-se balanceada se a distância de qualquer folha ao nó raiz for sempre a mesma, isto é, os nós folha estão todos ao mesmo nível
- B-Tree são árvores balanceadas muito utilizadas pelos SGBD para implementar índices multilevel
  - permitem uniformizar os tempos de pesquisa de valores de estrutura.

Indices - SQL Server
SQL Server tem dois tipos de índices (ambos implementados com B-Trees)

Clustered	Non-Clustered
- Os nós folha contêm os próprios dados na relação - A tabela está ordenada pelo próprio índice (só existe um por relação) - Analogia: Agenda de contatos telefónicos	- Os índices apontam para a tabela base - Podemos ter vários numa relação - Analogia: Índice no fim de um livro

Programação SQL

Batch - Grupo de uma ou mais instruções SQL que constituem uma unidade lógica.
Script - Ficheiro de texto contendo uma ou mais batches delimitadas por GO.

Declaração	Atribuição de um valor	Atribuição de um valor numa instrução SELECT
- DECLARE @nome VARCHAR(10) = 'João' - DECLARE @min INT	SET @nome = 'João Alegria'	SELECT @price = price FROM titles WHERE title_id = 901

Cursor - Ferramenta que permite percorrer sequencialmente os tuplos retornados por determinada consulta (SELECT).

Stored Procedure - Trata-se de uma batch armazenada com um nome.
- Os procedimentos são guardados em memória cache na primeira vez em que são executados, o que leva a uma rápida execução daí em diante.
- O procedimento pode: - ter parâmetros de entrada
  - ter valores de retorno (parâmetros de saída, mensagens de sucesso ou falha
  - devolver um conjunto de registos (tuplos)

```
CREATE PROCEDURE AlterarPassword
@IDMedico       INT,
@oldpassword VARCHAR(20),
@newpassword VARCHAR(20),
@status              INT OUTPUT

AS
BEGIN
        IF EXISTS ( SELECT *
                                FROM MedicalOne.Medico
                                WHERE IDMedico = @IDMedico
                                AND [password] = @oldpassword )

                BEGIN
                        UPDATE MedicalOne.Medico
                        SET [password] = @newpassword
                        WHERE IDMedico = @IDMedico
                        SET @status = 1
                END

        ELSE
                BEGIN
                        SET @status = 0
                END
END
RETURN @status
```

A implementação de um Stored Procedure tem como mais valia melhorias no desempenho, eliminando a necessidade de múltiplas transmissões de informação através da rede, segurança, uma vez que as operações de inserção, alteração, eliminação e consulta podem ser executadas a partir de um SP e assim as aplicações cliente deixam de ter necessidade de um conhecimento completo e minucioso da estrutra da BD.
E também a separação Cliente-Servidor já que facilitam a identificação entre tarefas que devem ser executadas pelo lado do cliente e do servidor, ou seja, o cliente analisa e interpreta os resultados e faz pedidos, e o servidor processa toda a informação.

UDF - User Defined Functions
- Mesmos benefícios dos Stored Procedures
- Mesmos benefícios das vistas pois podem ser utilizados como fonte de dados
  - Acresce o facto de aceitar parâmetros, algo impossível em views.

UDF Escalar - Aceita múltiplos parâmetros; retorna um único valor
- Não são permitidos updates à BD ou invocações do comando DBCC.

UDF Inline Table-Valued - Similares a vistas
- Tem as mesmas valias das vistas, acrescido do facto de suportar parâmetros de entrada

UDF Multi-Statement Table-Valued - Combina a capacidade das funções escalares (conter código complexo) com a capacidade das inline table-valued (retornar um conjunto).

Trigger - Um tipo especial de Stored Procedure que é executado em determinadas circunstâncias (eventos) associados à manipulação dos dados.
- Quando ocorre uma situação prevista, os triggers são "disparados" (executados)
- Disparados uma vez por cada operação de modificação dos dados

Trigger After - É possível ter vários triggers after por tabela.
- Validação de dados, efetuar autorias aos dados, atualizar campos calculados

Trigger Instead of - Apenas um por tabela (vista)
- Não é executada a ação associada, fica à responsabilidade do trigger.
- Devemos utilizar este tipo de trigger quando sabemos que a ação tem uma elevada probabilidade de ser roles back e pretendemos que outra lógica seja executada em vez (instead of) dela.

```
GO
CREATE Trigger highsales ON dbo.[Order Details] AFTER INSERT, UPDATE
AS

SET NOCOUNT ON;

DECLARE @total as real
SELECT @total = unitprice * (1-discount) * quantity FROM IF @total < 0.99

GO

BEGIN
RAISERROR ('Encomenda nao processada. Valor muito baixo', 16,1);
ROLLBACK TRAN; -- Anula a inserção

END
ELSE IF @total > 1000

PRINT 'Log: Encomenda de valor elevado'

GO
```

Segurança

SQL Injection - Tipo de ameaça de segurança que se aproveita de falha do sistema e que interage na BD via comandos SQL. SQL Injection ocorre quando é inserido uma série de comandos SQL dentro de uma consulta (query) através da manipulação das entradas dos dados de uma aplicação. Um ataque deste tipo pode:
- Expor informação
- Introduzir / alterar dados
- Eliminar dados
- Ganhar acesso a contas / privilégios de outros utilizadores

Como Prevenir?
- Não confiar nos dados introduzidos pelo utilizador - devemos validar toda a entrada com controles de validação, expressões regulares, etc..
- Não utilizar SQL Dinâmico - utilizar SQL parametrizado ou Stored Procedures
- Não armazenar informação sensível (ex: passwords) em texto simples - utilizar processos de cifragem ou hash.