

Universidade Virtual Africana

INFORMÁTICA APLICADA: CSI 3305

PRINCÍPIOS DE SISTEMA DE BASE DE DADOS

Aurelio Ribeiro

Prefácio

A Universidade Virtual Africana (AVU) orgulha-se de participar do aumento do acesso à educação nos países africanos através da produção de materiais de aprendizagem de qualidade. Também estamos orgulhosos de contribuir com o conhecimento global, pois nossos Recursos Educacionais Abertos são acessados principalmente de fora do continente africano.

Este módulo foi desenvolvido como parte de um diploma e programa de graduação em Ciências da Computação Aplicada, em colaboração com 18 instituições parceiras africanas de 16 países. Um total de 156 módulos foram desenvolvidos ou traduzidos para garantir disponibilidade em inglês, francês e português. Esses módulos também foram disponibilizados como recursos de educação aberta (OER) em oer.avu.org.

Em nome da Universidade Virtual Africana e nosso patrono, nossas instituições parceiras, o Banco Africano de Desenvolvimento, convido você a usar este módulo em sua instituição, para sua própria educação, compartilhá-lo o mais amplamente possível e participar ativamente da AVU Comunidades de prática de seu interesse. Estamos empenhados em estar na linha de frente do desenvolvimento e compartilhamento de recursos educacionais abertos.

A Universidade Virtual Africana (UVA) é uma Organização Pan-Africana Intergovernamental criada por carta com o mandato de aumentar significativamente o acesso a educação e treinamento superior de qualidade através do uso inovador de tecnologias de comunicação de informação. Uma Carta, que estabelece a UVA como Organização Intergovernamental, foi assinada até agora por dezenove (19) Governos Africanos - Quênia, Senegal, Mauritânia, Mali, Costa do Marfim, Tanzânia, Moçambique, República Democrática do Congo, Benin, Gana, República da Guiné, Burkina Faso, Níger, Sudão do Sul, Sudão, Gâmbia, Guiné-Bissau, Etiópia e Cabo Verde.

As seguintes instituições participaram do Programa de Informática Aplicada: (1) Université d'Abomey Calavi em Benin; (2) Université de Ougadougou em Burkina Faso; (3) Université Lumière de Bujumbura no Burundi; (4) Universidade de Douala nos Camarões; (5) Universidade de Nouakchott na Mauritânia; (6) Université Gaston Berger no Senegal; (7) Universidade das Ciências, Técnicas e Tecnologias de Bamako no Mali (8) Instituto de Administração e Administração Pública do Gana; (9) Universidade de Ciência e Tecnologia Kwame Nkrumah em Gana; (10) Universidade Kenyatta no Quênia; (11) Universidade Egerton no Quênia; (12) Universidade de Addis Abeba na Etiópia (13) Universidade do Ruanda; (14) Universidade de Dar es Salaam na Tanzânia; (15) Université Abdou Moumouni de Niamey no Níger; (16) Université Cheikh Anta Diop no Senegal; (17) Universidade Pedagógica em Moçambique; E (18) A Universidade da Gâmbia na Gâmbia.

Bakary Diallo

O Reitor

Universidade Virtual Africana

Créditos de Produção

Autor

Aurelio Ribeiro

Par revisor(a)

Martina Barros

UVA - Coordenação Académica

Dr. Marilena Cabral

Coordenador Geral Programa de Informática Aplicada

Prof Tim Mwololo Waema

Coordenador do módulo

Robert Oboko

Designers Instrucionais

Elizabeth Mbasu

Benta Ochola

Diana Tuel

Equipa Multimédia

Sidney McGregor

Michal Abigael Koyier

Barry Savala

Mercy Tabi Ojwang

Edwin Kiprono

Josiah Mutsogu

Kelvin Muriithi

Kefa Murimi

Victor Oluoch Otieno

Gerisson Mulongo

Direitos de Autor

Este documento é publicado sob as condições do Creative Commons

[Http://en.wikipedia.org/wiki/Creative_Commons](http://en.wikipedia.org/wiki/Creative_Commons)

Atribuição <http://creativecommons.org/licenses/by/2.5/>



O Modelo do Módulo é copyright da Universidade Virtual Africana, licenciado sob uma licença Creative Commons Attribution-ShareAlike 4.0 International. CC-BY, SA

Apoiado por



Projeto Multinacional II da UVA financiado pelo Banco Africano de Desenvolvimento..

Tabela de conteúdo

Prefácio	2
Créditos de Produção	3
Direitos de Autor	4
Apoiado por	4
Descrição Geral do Curso	7
Bem-vindo(a) a Princípios de Sistemas de Base de Dados	7
Pré-requisitos	7
Materiais	7
Objectivos do módulo	7
Unidades	8
Avaliação	9
Calendarização	9
Leituras e outros Recursos	10
Unidade 0. Diagnóstico	11
Introdução à Unidade	11
Objectivos da Unidade	11
Unidade 1. Introdução a Base de Dados	14
Introdução à Unidade	14
Objectivos da Unidade	14
Termos-chave	14
Actividades de Aprendizagem	15
Actividade 1.1 – Introdução a Base de Dados	15
Conclusão	18
Avaliação da actividade	19
Actividades de Aprendizagem	19
Actividade 1.2 - Sistemas de Gestão de Base De Dados (SGBD)	19
Conclusão	21
Avaliação da actividade	21

Actividades de Aprendizagem	21
Actividade 1.3.– Arquitectura de Base de Dados	21
Conclusão	23
Avaliação da actividade	23
Avaliação da Unidade	23
Soluções	23
Leituras e outros Recursos	24
Unidade 2. Modelos de Base de Dados	25
Introdução à Unidade	25
Objectivos da Unidade	25
Termos-chave	25
Actividades de Aprendizagem	26
Actividade 2.1 – Apresentação e descrição dos tipos de modelos de Base de Dados	26
Conclusão	32
Avaliação da actividade	32
Actividades de Aprendizagem	33
Actividade 2.2–Modelação de Dados	33
Conclusão	35
Avaliação da actividade	35
Actividades de Aprendizagem	35
Actividade 2.3–Modelo Entidade Relacionamento	35
Componentes do Modelo Entidade-Relacionamento	36
Resumo da Unidade	44
Conclusão	44
Avaliação da actividade	44
Avaliação da Unidade	44
Soluções	44
Leituras e outros Recursos	45
Unidade 3.Normalização de Dados	46

Introdução à Unidade	46
Objectivos da Unidade.	46
Termos-chave	46
Actividades de Aprendizagem	47
Actividade 3.1 – Dependências funcionais	47
Conclusão	50
Avaliação da actividade	50
Actividades de Aprendizagem	51
Actividade 3.2 – O que é Normalização?	51
1ª FORMA NORMAL	52
2ª FORMA NORMAL	53
3ª FORMA NORMAL	53
VANTAGENS / DESVANTAGENS DA NORMALIZAÇÃO	55
Vantagens	55
Desvantagens	55
Conclusão	55
Avaliação da actividade	55
Actividades de Aprendizagem	56
Actividade 3.3 – Boyce Codd até 5 Forma Normal	56
Conclusão	61
Avaliação da actividade	61
Avaliação da Unidade	62
Resumo da Unidade	62
Resolução:	63
Leituras e outros Recursos.	64
Unidade 4. Introdução à Álgebra Relacional e SQL	65
Introdução à Unidade	65
Objectivos da Unidade.	65
Termos-chave	65
Actividades de Aprendizagem	66

Actividade 4.1 –Operações Básicas em Álgebra relacional	66
Resumo da actividade..	69
Avaliação da actividade	69
Actividades de Aprendizagem	69
Actividade 4.2 –SQL	69
Resumo da actividade	75
Avaliação da actividade	76
Actividades de Aprendizagem	76
Actividade 4.3 – Comandos SQL	76
1. DDL - Data Definition Language (CREATE, ALTER, DROP, etc)	76
2.DML-Data Manipulation Language (SELECT, INSERT, UPDATE, DELETE)	77
3.DCL-Data Control Language (GRANT, REVOKE, etc)	78
Resumo da actividade..	80
Avaliação da actividade	80
Avaliação da Unidade	80
Soluções	80
Avaliação do Curso.	81
Avaliação Sumativa	81
Correção da Avaliação	83
Referências do Curso	86

Descrição Geral do Curso

Bem-vindo(a) a Princípios de Sistemas de Base de Dados

Neste módulo, vamos estudar os fundamentos necessários para compreender o Princípios de Sistema de Base De Dados. O

conteúdo deste módulo lhe ajudará a ter uma visão geral sobre introdução a base de dados. O módulo fornece ferramentas para o desenvolvimento de base de dados e utilização de sistemas de gestão de base de dados para aplicações. São também apresentados modelos de dados, linguagens de definição e manipulação de dados, entre outros conceitos. Além disso, apresenta-se oportunidades para explorar as diversas etapas na construção de uma base de dados como: design, desenvolvimento de aplicações e administração de base de dados. No final do módulo, espera-se que esteja familiarizado (a) com conceitos básicos e fundamentais na área de Base de Dados . O módulo está organizado em sete Unidades: Introdução à Base de Dados; Modelos de Base de Dados; Modelação de Dados; Normalização de Dados, Álgebra Relacional, e Introdução ao Structured Query Language (SQL).

Para cada unidade, o módulo fornece as actividades de ensino e de aprendizagem por forma a levar o aprendizes a aprender e a testar a qualidade da sua aprendizagem . As actividades podem ser feitas em grupo ou individualmente e é sugerida bibliografia adicional para complementar os seus estudos na área de base de dados. Finalmente, segue uma avaliação sumativa para ajudar os estudantes a avaliar a qualidade da sua aprendizagem.

Pré-requisitos

Antes de iniciar o estudo deste módulo, deverá ter conhecimentos básicos de: Competências Básicas de TIC; Estruturas de dados e algoritmos; Princípios da programação e Matemática básica (teoria dos conjuntos).

Materiais

Os materiais necessários para completar este curso incluem:

- Laboratório de informática;
- Requisitos de software (UMLStart ou outros modeladores e,MySQL 5 ou Oracle 11g);
- Ligação à Internet.

Objectivos do módulo

O módulo é uma introdução a base de dados com sistemas de gestão de base de dados (SGBD). Os(as) estudantes irão aprender a língua utilizada para gestão de BD, e as técnicas para trabalhar com uma base de dados. Assim, os objectivos do módulo são os seguintes:

- Mostrar como os dados podem ser armazenados em uma base de dados, as propriedades e o relacionamento de dados;
- Mostrar os passos para elaboração de uma base de dados;
- Indicar as técnicas de modelagem de base de dados;
- Identificar os passos para criar um modelo suportado por um SGBD;
- Mostrar como a teoria da álgebra relacional serve como base para uma organização eficiente e recuperação de grandes quantidades de dados;
- Apresentar algumas notações básicas (por exemplo, SQL) que implementam partes importantes da álgebra relacional;
- Providenciar aos alunos a experiência prática do uso e limitações de uma linguagem de consulta de banco de dados (como SQL), que são amplamente utilizados na indústria e nos negócios.

Unidades

Unidade 0: Diagnóstico

Esta Unidade centra-se na avaliação dos conhecimentos prévios do(a) estudante para o estudo deste módulo.

Unidade 1: Introdução a Base de Dados

Esta Unidade centra-se na definição de conceitos básicos de base de dados, assim como aborda os SGBD e os seus benefícios em relação ao sistema de arquivos.

Unidade 2: Modelos e Modelação de Base de Dados

Esta Unidade mostra os diferentes modelos de bases de dados e os seus benefícios em relação aos outros, assim como aborda sobre a necessidade de modelagem de dados, com objectivo de ter uma Base de Dados bem projectada.

Unidade 4: Normalização de Dados

Esta Unidade mostra os passos para normalizar uma base de dados até à terceira forma normal, de maneira a evitar anomalias em projectos de BD.

Unidade 4: Introdução à Álgebra Relacional e ao SQL

Esta Unidade apresenta uma ferramenta matemática chamada Álgebra Relacional, que é muito poderosa para escrever consultas em base de dados. A par destes assuntos apresentaremos também dos comandos básicos: DDL, DML e DCL assim como algumas restrições de integridade de dados.

Avaliação

Em cada unidade encontram-se incluídos instrumentos de avaliação formativa a fim de verificar o progresso do(a)s estudante(s).

No final de cada módulo são apresentados instrumentos de avaliação sumativa, tais como testes e trabalhos finais, que compreendem os conhecimentos e as competências estudadas no módulo.

A implementação dos instrumentos de avaliação sumativa fica ao critério da instituição que oferece o curso. A estratégia de avaliação sugerida é a seguinte:

Calendarização

Unidade	Temas e Atividades	Estimativa do tempo
1: Introdução a Base de Dados	Atividade 1.1 – Introdução a Base de Dados Atividade 1.2 - Sistemas de Gestão de Base De Dados Atividade 1.2 – Arquitectura de Base de dados	10
2: Modelos e Modelação de Base de Dados	Atividade 2.1 – Apresentação e descrição dos tipos de modelos de BD Atividade 2.2 – Modelação de Dados Actividade 2.3–Modelo Entidade Relacionamento	40
3. Normalização de Dados	Atividade 3.1 – Dependências funcionais Atividade 3.2 – O que é Normalização? Atividade 3.3 – Boyce Codd até 5 Forma Normal	25
4: Introdução a Álgebra Relacional e ao SQL	Atividade 4.1 – Operações Básicas em Álgebra relacional Atividade 4.2 – Trabalhando com SQL Atividade 4.3 – Comandos SQL	45

Leituras e outros Recursos

As leituras e outros recursos deste curso são:

Unidade 1

Leituras e outros recursos obrigatórios:

Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431

Introdução a Sistemas de Banco de Dados, DATE, C. J. Elsevier Editora, 2004.

Banco de Dados- Vol I, Sandra de Albuquerque SIEBRA, Recife, 2010

Unidade 2

Leituras e outros recursos obrigatórios:

Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431

Banco de Dados- Vol II, Sandra de Albuquerque SIEBRA, Recife, 2010

Unidade 3

Leituras e outros recursos obrigatórios:

Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431

Banco de Dados- Vol II, Sandra de Albuquerque SIEBRA, Recife, 2010

Unidade 4

Leituras e outros recursos obrigatórios:

Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431.

Banco de Dados- Vol III, Sandra de Albuquerque SIEBRA, Recife, 2010

<http://www.ic.unicamp.br/~celio/inf325-2011/conexao-algebra-relacional-sql.html> , capturado em outubro de 2014

<http://www.din.uem.br/~ra55976/AlgebraRelacional.pdf> , capturado em novembro de 2014

SQL, Luis DIMAS, Editora FCA, Portugal, Maio-2005.

Banco de Dados- Vol IV, Sandra de Albuquerque SIEBRA, Recife, 2010

www.w3schools.com/sql capturado em novembro de 2014.

Unidade 0. Diagnóstico

Introdução à Unidade

O propósito desta unidade é verificar o seu o nível de conhecimentos prévios dos (as) estudantes para fazer este curso.

Formandos (as):

Nesta seção você vai encontrar questões de auto-avaliação que irão ajudá-lo (a) a testar sua prontidão para fazer este módulo. Você tem que encarar esta unidade como uma base para poder triunfar no estudo deste módulo.

Instrutores:

Questões de pré-avaliação fornecidas aqui levam os(as) estudantes a saber se estão preparados, no entanto, sugere-se que você incentive os (as) estudantes a avaliar-se, respondendo a todas as perguntas apresentadas abaixo. Caso os (as) estudantes dominem os conteúdos considerados pré-requisitos, eles têm a base necessária para iniciar o Módulo.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Relacionar os conceitos de outras disciplinas;
- Implementar conhecimentos de outras disciplinas para este módulo;
- Analisar o seu grau de competências para começar este módulo.

Perguntas: Escolha a opção correcta para cada questão.

1. O menor nível de representação de dados é:

- a . Word
- b . byte
- c . Bit
- d . long word

2. O sistema operacional é como um:

- a . Software Utilitário
- b . Processador de textos
- c . Gestor de vírus
- d . Nenhuma das alternativas acima

3. Qual dos seguintes dispositivos não é de armazenamento ?
- a . DVD
 - b . Disco rígido
 - c . Disquete
 - d . Mouse
4. O cérebro de um computador
- a . UAL
 - b . Memória
 - c . CPU
 - d . Sistema Operacional
5. Define-se Dados como sendo:
- a. Conjunto de Informação com significado.
 - b. Conjunto de símbolos articulados entre si com significado.
 - c. Conjunto de símbolos não articulados entre si sem significado.
 - d. Nenhuma das alternativas está correcta.
6. Define-se Informação como sendo:
- a. Conjunto de dados sem significado.
 - B. Conjunto de dados articulados entre si com significado.
 - c. Conjunto de dados articulados entre si sem significado.
 - d. Nenhuma das alternativas é correcta.
7. SQL significa?
- a. Structured Query Language
 - b. Sequential Query Language
 - c. Semantic Query Language
 - d. Solution-based Query Language

8. Qual ser[ia o resultado do seguinte programa?

```
#include<stdio.h>

int main()

int goto=5;

printf("%d",goto);
```

```
return 0;  
}
```

- a. 5
- b. **
- c. ***
- d. Erro de compilação
- e. Nenhuma das alternativas acima

9. Qual é o resultado de $31 \bmod 10$

- a. 3
- b. 1
- c. 3.5
- d. Nenhuma das alternativas

10. Java é um(a)

- a . Sistema Operacional
- b . compilador
- c . dispositivo de Entrada
- d . Linguagem de Programação

11. WAN significa

- a . Wap Area Network
- b . Wide Area Network
- c . Rede privada
- d . Rede local sem fio

12. Qual é o resultado de $3 \div 10$

- a. 3
- b. 1
- c. 0
- d. Nenhuma das alternativas

Soluções

1c; 2d; 3d; 4c; 5c; 6b; 7a ;

8 d; Explicação: Nome de variável invalido, pois, goto é uma palavra reservada na linguagem C e não pode ser declarado como variável.

9 b; 10 d; 11b; 12c.

Unidade 1. Introdução a Base de Dados

Introdução à Unidade

Você sabe o que é uma base dados? Você saberia dizer o quanto essa área está presente na sua vida? Sabia que pode ser muito mais do que você imagina? Sabia que ao efectuar compras num supermercado ou ao levantar dinheiro numa caixa automática bancária, você estaria fazendo uso directo ou indirecto de uma base de dados?

Afinal, você foi estudante de uma instituição e deve estar na base de dados deles. Se você tem uma conta bancária, você está na base de dados do banco ao qual sua conta pertence. Isso só para começo de conversa, porque deve haver muitas outras bases de dados dos quais você faz parte. Logo, só por ser uma área tão presente na sua vida, a área de Base de Dados já merece ser estudada, não acha? Fora isso, qualquer sistema que você pensar em desenvolver na sua vida, deverá fazer uso de uma base de dados e mesmo se você não desenvolver sistemas, mas utilizá-los no seu dia-a-dia, você será, com certeza, um utilizador de Base de Dados. Então, que tal conhecer essas tais Bases de Dados mais ao fundo? Começaremos a fazer isso nesta Unidade.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Identificar os principais conceitos relacionados à Base de Dados;
- Diferenciar um sistema de Ficheiros de um sistema Gestão de Base de dados;
- Enumerar diferentes tipos da Base de Dados;
- Listar os benefícios dos SGBD;
- Explicar as partes que compõem a arquitectura de SGBD.

Termos-chave

Dados, Informação, Base de Dados. SGBD

Dados: é um elemento que mantém a sua forma bruta e ele sozinho não levará a compreender determinada situação.

Informação: é o conjunto de dados contextualizados colectados de forma a se tornarem aplicáveis a determinada situação.

Base de Dados: De uma forma simples, podemos dizer que Base de dados é uma colecção de dados estruturados, organizados e armazenados de forma persistentes em um dispositivo computacional ou não.

SGBD: Sistema de Gestão de Base de Dados é um aplicativo informático que oferece uma interface entre os dados que são armazenados fisicamente na base de dados e o utilizador.

Actividades de Aprendizagem

Actividade 1.1 – Introdução a Base de Dados

Introdução

Nesta actividade, vamos abordar os conceitos básicos da área de Base de Dados, pois, essa é uma área presente no nosso dia-a-dia e que ganha ainda mais importância quando paramos para pensar que estamos vivendo na chamada “Era da Informação”, onde o conhecimento adquirido a partir das informações é o maior bem de qualquer empresa ou instituição e, as informações são obtidas a partir de dados que precisam estar armazenados em algum lugar (Base de Dados).

Para a realização esta actividade deve começar por ler o seguinte texto:

Base de dados (BD)

Existem muitas definições para base de dados e nós vamos ver mais do que uma definição, entretanto, de uma forma simples podemos dizer que, base de dados consiste numa colecção de dados estruturados, organizados e armazenados de forma persistente. Entretanto, uma BD não tem, necessariamente, de estar informatizada, pois, pode perfeitamente consistir num ficheiro manual como uma agenda com as moradas de pessoas conhecidas, uma lista de CDs, um livro, apontamentos tirados nas aulas, os dados guardados nos computadores das Finanças sobre os contribuintes e a World Wide Web.

Uma Base de Dados (também chamado de Banco de Dados) é uma colecção de dados relacionados, organizados e armazenados visando facilitar a manipulação dos dados armazenados, permitindo realizar alterações, inserções, remoções e consultas. Quaisquer aplicações do mundo real que possam ser representadas através de dados poderão ser armazenadas em um banco de dados.

“Um Banco de Dados é uma colecção de dados operacionais armazenados, sendo usados pelos sistemas de aplicação de uma determinada organização” (DATE, 2000).

“Um Banco de dados é um conjunto de dados armazenados, cujo conteúdo informativo representa a cada instante, o estado actual de uma determinada aplicação” (LAENDER, 1993).

Dados: é um elemento que mantém a sua forma bruta (texto, imagens, sons, vídeos, etc.) e ele sozinho não levará a compreender determinada situação. Ou seja, o termo “Dado” envolve fatos, imagens, sons que podem ou não ser úteis para determinado fim, eles apenas representam coisas do mundo real.

Informação: é o conjunto de dados contextualizados colectados de forma a se tornarem aplicáveis a determinada situação, ou seja, sua forma e conteúdo são apropriados para um uso específico.

Sistemas de Ficheiro: Antes de surgirem as base de dados

Antes de existirem as bases de dados, qualquer sistema, programa ou aplicação que precisasse armazenar e manipular dados fazia uso de um sistema de ficheiros (também chamados de sistemas de arquivos). Ou seja, cada sistema, programa ou aplicação desenvolvido tinha seus próprios arquivos de armazenamento dos dados. Geralmente, naquela época, os programas eram escritos em respostas às necessidades, ou seja, iam sendo desenvolvidos na medida em que as necessidades apareciam e muitas vezes, eram desenvolvidos por programadores diferentes.

Qual a implicação disso? Diferentes programadores pensam e programam de forma diferente. Desse modo, os arquivos de cada programa desenvolvido poderiam estar em formatos diferentes e poderiam ter sido usadas linguagens de programação diferentes. Qual o problema com isso? O problema com isso é que entre os diversos sistemas desenvolvidos gradualmente para uma determinada empresa ou instituição poderiam haver dados em comum.

Vamos dar um exemplo: suponha que uma fábrica possui um Sistema de Vendas, um Sistema de Produção e um Sistema de Engenharia (vide Figura 3). Cada um deles teria seus próprios arquivos e nesses arquivos poderia existir em comum, por exemplo, os dados de um produto, que por causa dessa organização precisaria ser replicado em cada sistema de arquivos. E qual a consequência dessa replicação de dados dentro de uma mesma fábrica?

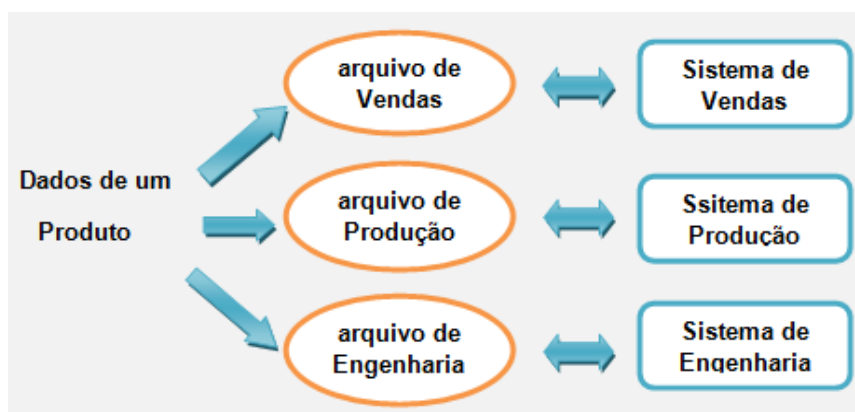


Figura 1: Exemplo de sistemas que fazem uso de um Sistema de Arquivos

A consequência é que essa redundância poderia acarretar inconsistência dos dados, uma vez que a mesma informação poderia estar duplicada em diversos arquivos (no exemplo, os dados do produto). Vamos dar um exemplo. Vamos supor que nos arquivos de Vendas, você actualizou o nome do produto 01 de Mesa para Cadeira. Se você desejasse manter a consistência, eu teria de entrar nos arquivos de Produção e de Engenharia e fazer a mesma actualização. Se não

fizesse, os dados do produto na fábrica ficariam inconsistentes, pois dependendo do sistema acedido o produto 01 poderia ser Mesa ou Cadeira. Ficou claro?

Além disso, a duplicação de dados em diversos sistemas de arquivos leva a um maior custo de armazenamento (lembre, você está armazenando diversas vezes os mesmos dados) e a necessidade de voltar a digitar os dados (e esse trabalho repetitivo pode levar a erros, que também geram inconsistências entre os dados). Adicionalmente, o uso de sistemas de arquivos possui outros problemas tais como:

Dificuldade do acesso a dados – a geração de informação pode surgir, durante o tempo em que o sistema está em produção, sob diferentes aspectos. Cada requisição de informação diferente, no sistema de arquivos, vai gerar a necessidade da criação de um programa aplicativo, de uma nova consulta ou de um novo relatório. Dessa forma, a recuperação de informação não é atendida de modo eficiente. Haveria dificuldade em apagar informações dos sistemas. Poderia novamente ocorrer casos de inconsistência, onde um produto poderia ser apagado dos arquivos de Vendas, mas não dos outros dois arquivos (Engenharia e Produção).

Isolamento dos dados – os dados estão armazenados em arquivos distintos, que não possuem qualquer tipo de relacionamento directo, e ainda, podem conter diferentes formatos para o mesmo dado. Por exemplo, o código do produto nos arquivos de venda poderia ser representado só por números e nos arquivos de Produção por letras e números.

Problemas de integridade – fica difícil manter restrições de integridade automaticamente. E o que são essas restrições de integridade? Seriam verificações de determinadas condições a serem feitas pelo sistema sobre os dados armazenados (por exemplo, a quantidade de produtos em stock não pode ser inferior a um valor X). Essas restrições teriam de ser mantidas pelo sistema, implicando em implementação do código apropriado para fazer esse tipo de verificação em CADA UM dos sistemas afectados. O problema seria que, a cada inclusão de uma nova restrição de integridade, um novo código deveria ser escrito EM CADA SISTEMA para poder mantê-la, já que cada sistema trabalha com um diferente sistema de arquivos.

Problemas de segurança - Nem todos os usuários do sistema devem estar autorizados a ver/aceder todos os dados armazenados. Porém, uma vez que os programas de aplicação são inseridos no sistema como um todo, de forma aleatória (à medida que a necessidade surge) é difícil implementar e garantir a efectividade de regras de segurança.

Anomalias no acesso concorrente- a melhoria de desempenho em um sistema pode ocorrer pela execução simultânea de diversas operações. Geralmente, nos sistemas de arquivos, esta melhoria seria difícil de implementar sem levar a danos na consistência dos dados. Ou seja, seria difícil permitir que múltiplos usuários possam ter acesso aos dados ao MESMO TEMPO. Vamos dar um exemplo.

Considere um sistema bancário (com a existência de contas correntes, agências, transacções bancárias, etc) e a seguinte situação: suponha que o saldo de uma conta bancária Conta_#1 seja 500 reais. Se dois clientes retiram fundos desta conta Conta_#1 AO MESMO TEMPO (acesso concorrente à conta Conta_#1), um estado inconsistente pode ocorrer se na execução das duas instâncias do programa de débito, ambos os clientes lerem o saldo inicial original e retirarem, cada um, seu valor correspondente, e seja então armazenado o valor restante. Instanciando o problema:

1. Ambos lêem o valor do saldo 500;
2. Um retira 50 (resultando 450) e o outro 100 (resultado 400) – lembre-se que ambos estão realizando a operação em cima dos 500 iniciais que foi lido;
3. Dependendo de qual execução do programa de débito registre o saldo restante primeiro, o valor do saldo da conta será 450 ou 400, quando, na verdade, deveria ser 350. Outros problemas existentes são:

A definição das estruturas dos arquivos está inserida no próprio código dos programas, sistemas ou aplicativos, dificultando a manutenção;

Compartilhamento de um arquivo por vários programas fica comprometido. Há a necessidade de duplicar a definição das estruturas dos arquivos nos programas;

Podem existir arquivos e programas de um mesmo sistema desenvolvidos, de forma isolada, por diferentes programadores de forma diferentes e, até mesmo, em linguagens de programação diferentes.

Justamente a necessidade de resolver diversos desses problemas motivou a criação das Base de Dados e dos SGBD. Com eles, os dados só necessitam ser armazenados uma única vez e passam a ser acedidos por todos os sistemas (vide Figura abaixo)

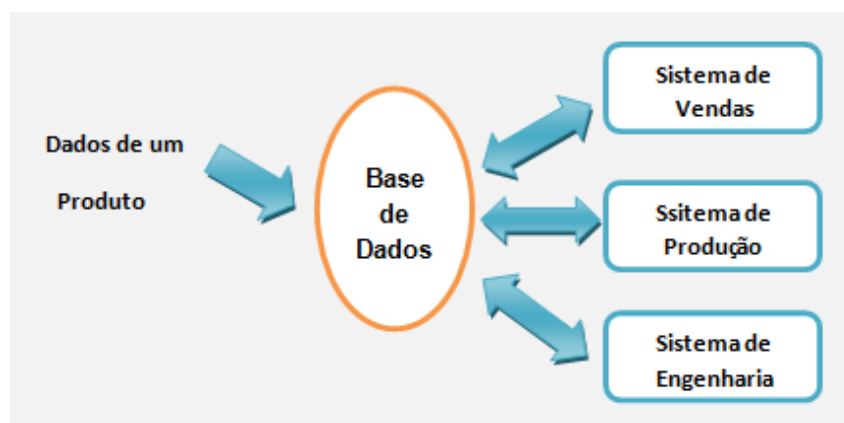


Figura 2:. Exemplo de Sistemas fazendo uso de um Banco de Dados

Conclusão

Nesta actividade introduzimos os sistemas de base de dados, onde se retratou os sistemas de ficheiros e as respectivas anomalias ou problemas, que teve como consequência o surgimento dos sistemas de gestão de base de dados, assunto a ser tratado na próxima unidade.

Avaliação da actividade

Verifique a sua compreensão!

1. Faça um resumo sobre Sistema de Ficheiro e SGBD.
2. Liste pelo menos cinco formas de interacção com base de dados no seu dia-a-dia.
3. Quais são as anomalias de um sistema de ficheiros?

Actividades de Aprendizagem

Actividade 1.2 - Sistemas de Gestão de Base De Dados (SGBD)

Introdução

Todas as organizações, por menor que sejam, possuem quantidades cada vez maiores de dados e informações a armazenar. Todavia, a manipulação destas informações se tornou impossível de ser realizada manualmente (via papéis, principalmente), pois sua utilização além de demorada (devido a catalogação dos dados) é passível de erros principalmente ocasionados pelo desgaste do operador em conseguir resgatar informações requisitadas. Nesse sentido, torna-se mais fácil encontrar a informação numa base de dados que recorre a uma das tecnologias de informação de maior sucesso e confiança. Ou seja, as bases de dados estendem a função do papel ao guardar a informação em computadores. Como iremos ver nesta actividade, o SGBD, veio melhorar as anomalias existentes no sistema de ficheiro.

Comece por ler o texto que se segue:

Sistema de Gestão de Base de Dados

Um SGBD é uma aplicação informática (ie, um software) que fornece a interface entre os dados que são armazenados fisicamente na BD e o utilizador. Desta forma, o utilizador deixa de ter de se preocupar com a forma como os dados são armazenados, pesquisados ou ordenados, pois é o SGBD que tem a responsabilidade dessa tarefa. Este utilizador pode ser alguém (pessoa) ou um aplicativo informático.

O SGBD tem uma gama de funções pré-implementadas que gerem as operações de inserção, remoção, actualização e consulta dos dados armazenados. É o SGBD que vai fornecer um conjunto completo de serviços de acesso aos dados. Para isso, necessita de uma linguagem que permita ao utilizador (pessoa ou aplicação) interrogar ou operar sobre a base de dados. É aqui onde surge a linguagem SQL (Structured Query Language).

.Utilizadores de um SGBD

Uma BD pode apresentar diversos utilizadores cada qual com uma necessidade em particular, e com um envolvimento diferente com os dados do BD. Que tal conhecer um pouco mais? Os utilizadores podem ser classificados nas seguintes categorias:

Administrador de BD (DBA – Data Base Administrator) - em qualquer organização onde muitas pessoas compartilham diversos recursos, existe a necessidade de um administrador (sempre existe o que vai ficar à frente, não é?) para supervisionar e gerir estes recursos (em um ambiente de base de dados, o recurso primário é a própria base de dados e os recursos secundários são o próprio SGBD e software relacionados). A administração desses recursos é de responsabilidade do DBA. Ele é responsável, entre outras coisas, por autorizar acesso à base de dados, coordenar e monitorar seu uso. Adicionalmente, ele é responsável por problemas, tais como, quebra de segurança ou baixo desempenho do BD.

Analista de Base de Dados – possui a responsabilidade de identificar os dados a serem armazenados na BD e pela escolha da estrutura apropriada a ser utilizada para armazená-los. Ou seja, é a pessoa responsável pelo projecto de construção e utilização da BD, envolvendo as tarefas de definição de quais dados deverão ser construídos e como eles serão construídos. Pode existir um único analista ou um grupo deles e ele(s) irá(ão) interagir com outras classes de usuários, tanto os analistas e programadores, como os chamados utilizadores finais do sistema, a fim de obter a visão dos dados que cada um possui, integrando-as de forma a se obter uma representação adequada de todos os dados.

Utilizadores Finais: existem profissionais que precisam ter acesso à base de dados para consultar, modificar e gerar relatórios. A base de dados existe para estes utilizadores. Existem algumas categorias de usuários finais:

- › Utilizadores ocasionais: ocasionalmente fazem acesso à base de dados, mas eles podem necessitar de diferentes informações a cada vez que fazem acesso.

Eles podem usar uma linguagem de consulta sofisticada para especificar suas requisições e são, tipicamente, gerentes de médio ou alto nível;

- › Utilizadores comuns ou paramétricos: estes usuários realizam operações padrões de consultas e actualizações, chamadas transacções permitidas, que foram cuidadosamente programadas e testadas. Estes utilizadores constantemente realizam recuperações e modificações na base de dados;

- › Utilizadores sofisticados: incluem engenheiros, analistas de negócios e outros que procuraram familiarizar-se com as facilidades de um SGBD para atender aos seus complexos requisitos;

- › Analistas de Sistemas e Programadores de Aplicações: são os Engenheiros de Software, aquelas pessoas que determinam as necessidades dos utilizadores.

Exemplos de SGBDs

IBMInfor-mix	DB2	SQL-Ser-ver	Interbase
PostgreS-QL	mSQL	Oracle	Ingess
Firebird	MySQL	Sybase	etc

Tabela1: Exemplos de SGBD

Conclusão

Caro estudante, nesta actividade abordamos os sistemas de gestão de base de dados. Trata-se de uma tecnologia que veio revolucionar as anomalias do sistema de ficheiros. Portanto foi apresentado o conceito de SGBD, os diversos utilizadores de Base de dados e, alguns exemplos de sistemas de gestão de base de dados, base que podemos encontrar no mercado.

Avaliação da actividade

Verifique a sua compreensão!

1. Depois de ter lido o texto que acompanha a Actividade 1.2 (SGBD), diga qual é a importância de SGBD?
2. Quais são as responsabilidades de um administrador de BD?
3. Tendo em conta os exemplos de SGBD acima citados, investigue sobre a funcionalidade e historial de cada um deles e faça um resumo.

Actividades de Aprendizagem

Actividade 1.3.– Arquitectura de Base de Dados

Introdução

Nesta actividade, iremos abordar a arquitectura de base de dados, tendo em conta a ANSI/SPARC. Num primeiro nível as pessoas interagem com os programas de aplicação (programas desenvolvidos em uma linguagem de aplicação), que foram criados para os utilizadores finais utilizando-se uma linguagem de consulta (linguagem própria para acesso a base de dados). Esta aplicação interage com o SGBD, que possui programas responsáveis por processar as consultas e aceder aos dados armazenados, dentre outras funções. Por fim, num nível mais interno, encontra-se a base de dados, separada em dois arquivos distintos, um contendo a definição dos dados e outro contendo os dados propriamente ditos, ou seja, os dados armazenados.

Arquitectura ANSI/SPARC

Numa tentativa para estabelecer um padrão para toda industria de desenvolvimento de tecnologia de base de dados, foi definida a arquitectura ANSI1/SPARC2 com objectivo de permitir que um SGBD possa ser utilizado por vários tipos de utilizador, respeitando a particularidade e necessidade de cada um.

A arquitectura ANSI/SPARC é composta por três níveis independentes, cada um deles descrevendo a BD a um nível diferente de abstracção.



Figura 3. Arquitectura ANSI/SPARC

Nível Externo – Definição de views sobre os esquemas conceptual global. Consiste numa “janela” parcial que é criada sobre a totalidade da BD. Permite trabalhar apenas com uma parte dos dados que tenham interesse para uma determinada aplicação. Para essa aplicação, o view funciona como se estivesse a operar sobre a totalidade dos dados.

Nível conceptual - Representação do modelo conceptual de dados, independentemente de qualquer utilizador ou aplicação. Esta camada esconde os detalhes de implementação física dos ficheiros que armazenam os dados.

Nível interno - Armazenamento físico dos dados e definição das estruturas físicas que permitem obter um bom nível de desempenho.

Este tipo de arquitectura permite:

- Independência de Programas/Dados - Podemos dizer que é permitido efectuar alterações que envolvam a estrutura dos dados, ou a sua implementação física, não obrigam alteração no nível operacional.
- Independência lógica - permite alterar apenas o nível conceptual, não havendo nenhuma alteração no nível externo ou nas aplicações dos utilizadores.
- Independência física - permite alterar o nível interno sem ter que alterar o nível conceptual, nível externo ou as aplicações dos utilizadores.

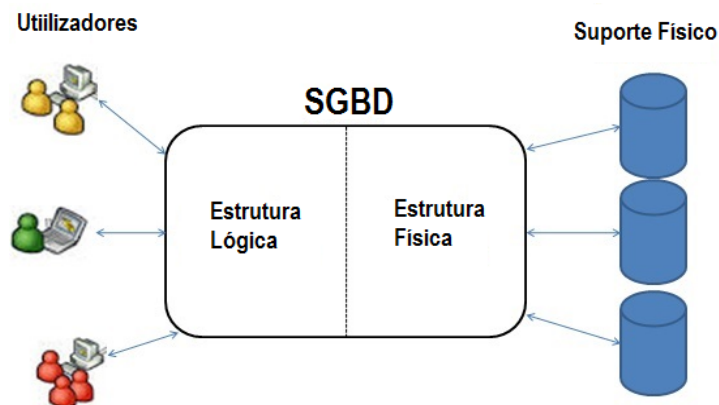


Figura 4: Arquitectura do SGBD

Conclusão

Nesta actividade abordamos a arquitectura de base de dados, com enfoque na proposta ANSI/SPARC, tendo se considerado as camadas ou níveis desta arquitectura, nomeadamente, Nível Externo; Nível Conceptual; Nível Interno e o Nível de Dados. A par destes assuntos, falamos do ganho que temos nesta arquitectura, tais como: Independência de física; Independência lógica E Independência DE PROGRAMA/DADOS.

Avaliação da actividade

Verifique a sua compreensão!

1. Quais são as camadas que compõem a arquitectura ANSI/SPARC?
2. Diferencie a independência lógica da independência física.
3. Relacione as duas arquitecturas abordadas nesta actividade.

Resumo da Unidade

Caro estudante, nesta unidade apresentamos três actividades, respectivamente, Introdução à Base de Dados, Sistemas de Gestão de Base De Dados e Arquitectura de Base de Dados. Esperamos que tenha compreendido os motivos que levaram a migração dos sistemas de ficheiros para os sistemas de gestão de base de dados, assim como a relação entre os utilizadores, o SGBD e os suportes físicos.

Avaliação da Unidade

Verifique a sua compreensão!

1. Defina o conceito dados.
2. Após definir dados, diga o que é informação.
3. O que entende por Nível conceptual em base de dados?
4. Qual é o papel dos utilizadores finais em base de dados?

Soluções

1. Dados é um elemento que mantém a sua forma bruta (texto, imagens, sons, vídeos, etc.) e ele sozinho não levará a compreender determinada situação. Ou seja, o termo "Dado" envolve fatos, imagens, sons que podem ou não serem úteis para determinado fim, eles apenas representam coisas do mundo real.
2. Informação é o conjunto de dados contextualizados colectados de forma a se tornarem aplicáveis a determinada situação, ou seja, sua forma e conteúdo são apropriados para um uso específico.

3. Nível conceptual representação do modelo conceptual de dados, independentemente de qualquer utilizador ou aplicação. Esta camada esconde os detalhes de implementação física dos ficheiros que armazenam os dados.

4. Utilizadores finais são profissionais que precisam ter acesso à base de dados para consultar, modificar e gerar relatórios. A base de dados existe para estes utilizadores.

Leituras e outros Recursos

Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431

Banco de Dados- Vol II, Sandra de Albuquerque SIEBRA, Recife, 2010

Unidade 2. Modelos de Base de Dados

Introdução à Unidade

Introdução

As primeiras BD surgiram nos anos 60, com o surgimento dos modelos de base de dados hierárquicos e em rede, com acesso sequencial aos dados e processamento em batch (em lote, várias instruções de uma vez), essa foi a chamada primeira geração. Os dois modelos da época eram o Hierárquico e o em Rede, ambos orientados a registos, isto é, qualquer acesso à base de dados – inserção, consulta, alteração ou remoção – era feita em um registo de cada vez. Vamos detalhar nesta unidade, cada um desses modelos.

No entanto, se você pretende desenvolver aplicações que usam BD relacionais, você, com certeza, deverá dominar os conceitos básicos sobre modelagem de dados. Não importa o tamanho da sua aplicação ou a complexidade da mesma, sempre será muito importante ter bem projectado o local onde os dados serão armazenados, de forma que eles possam ser recuperados de forma fácil, íntegra e correcta. É justamente o “como fazer” o projecto de banco de dados que veremos nesta unidade. Lembre-se, esse é um dos assuntos mais importantes da nossa disciplina, logo, leia esta unidade com mais atenção do que qualquer outra e não se esqueça de praticar a aplicação dos conceitos que serão aprendidos. Vamos lá?

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Identificar e Listar os diferentes modelos de base de dados;
- Dar um exemplo de cada modelo;
- Explicar a necessidade de modelagem de dados em projectos de BD.

Termos-chave

Modelos de dados, Modelação de dados, Entidades, Atributos e Relacionamentos.

Modelo de Dados: são as principais ferramentas que fornecem a abstracção a uma BD, visto que o modelo de dados representa, de forma abstracta, como aspectos do mundo real (fatos) estão relacionados, a fim de que possam ser representados no mundo computacional.

Entidades: é algo do mundo real que possui uma existência independente. Uma entidade pode ser um objecto com uma existência física.

Atributos: São propriedades descritivas de cada membro/propriedade de uma entidade ou relacionamento. Em outras palavras, uma entidade sempre é representada por um conjunto de atributos.

Relacionamentos: São associações entre uma ou várias entidades.

Actividades de Aprendizagem

Actividade 2.1 – Apresentação e descrição dos tipos de modelos de Base de Dados

Introdução

Caros estudantes, nesta secção, apresentaremos os diversos tipos de modelos de base de dados, respectivamente: Modelo de Dados Hierárquico; Modelo de Dados em Redes; Modelo de Dados Relacional; Modelo de Dados Orientado a Objectos e Modelo de Dados Objecto-Relacional. Comece esta actividade de aprendizagem lendo o texto que se segue:

Modelo de Dados Hierárquico

O modelo hierárquico foi o primeiro a ser reconhecido como um modelo de dados. Nesse modelo, os dados são estruturados em hierarquias ou árvores. Os nós das hierarquias contêm ocorrências de registos, onde cada registo é uma colecção de campos (atributos), cada qual contendo somente um valor.

Os registos são conectados uns aos outros por meio de uma ligação, também chamada de link (associação essa entre exactamente dois registos). O registo da hierarquia que precede a outros é o registo-pai, os outros são chamados de registos-filhos. O relacionamento entre um registo-pai e vários registos-filhos possui cardinalidade 1:N, ou seja, cada registo-pai pode possuir 1 ou mais registos-filhos, mas os registos filhos possuem apenas um único registo pai (Veja a Figura 5).

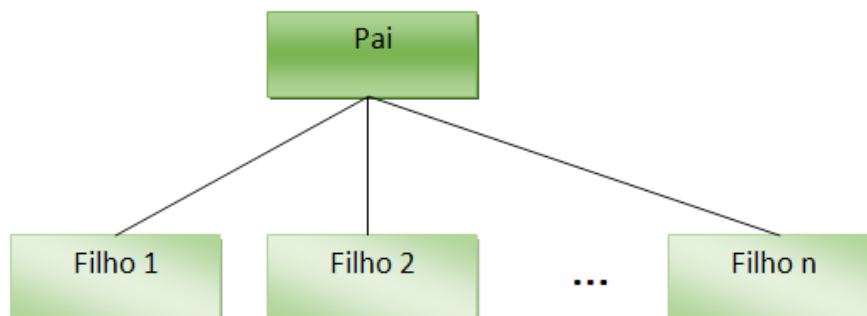


Figura 5: Modelo de Dados Hierárquico

Os dados organizados, segundo este modelo, podem ser acedidos segundo uma sequência hierárquica com uma navegação do topo (raiz) para as folhas (nós sem filhos) e da esquerda para a direita dentro de cada nó ou registo. Um mesmo registo pode estar associado a vários registos diferentes, desde que seja replicado. Porém, a replicação possui duas grandes desvantagens: pode causar inconsistência de dados quando houver actualização e causar desperdício de espaço.

Modelo de Dados em Redes

O modelo em redes surgiu como uma extensão ao modelo hierárquico, eliminando o conceito de hierarquia e permitindo que um mesmo registo estivesse envolvido em várias associações. E que benefícios trás isso? Bem, isso faz com que seja possível construir relações menos restritivas entre os registos.

No modelo em rede, os registos são organizados em grafos onde aparece um único tipo de associação (set) que define uma relação 1:N entre 2 tipos de registos: proprietário e membro. Desta maneira, dados dois relacionamentos 1:N entre os registos A e D e entre os registos C e D é possível construir um relacionamento M:N (muitos para muitos) entre A e D, o que não era possível no modelo hierárquico. Adicionalmente, ao contrário do Modelo Hierárquico, em que qualquer acesso aos dados passa pela raiz, o modelo em rede possibilita acesso a qualquer nó da rede sem passar pela raiz.

Dessa forma, no modelo em redes, os registos são organizados na base de dados por um conjunto arbitrário de grafos.



Figura 6: Modelo de Dados em Redes

Alguns problemas do modelo em redes são:

O modelo de rede é fortemente dependente da implementação.

As consultas são complicadas: o programador é forçado a pensar em termos de links e, em como percorrê-los para obter as informações de que necessita. Essa manipulação de dados é chamada navegacional.

Segunda Geração dos Bancos de Dados

Nos anos 70 e 80, com os dispositivos de acesso directo aos dados, surgem os sistemas indexados e de processamento transaccional (cada operação deve ser completamente realizada, se não o for, deve ser desfeita, esse é o conceito de transacção).

Nos anos 80, foram lançadas as bases do modelo relacional (dando origem à segunda geração) que impulsionou o uso e abriu caminho para todos os SGBDs actuais.

Modelo de Dados Relacional

O modelo relacional apareceu devido às seguintes necessidades:

- aumentar a independência de dados nos sistemas de gestão de base de dados;
- prover um conjunto de funções apoiadas em álgebra relacional para armazenamento e recuperação de dados;
- permitir processamento ad-hoc

O Modelo Relacional (MR) é considerado o primeiro modelo de dados efectivamente usado em aplicações comerciais. Foi introduzido por Codd em 1970. É o modelo que possui a base mais formal entre os modelos de dados, entretanto é o mais simples e com estrutura de dados mais uniforme. O modelo relacional tem como base a teoria dos conjuntos e a álgebra relacional.

A estrutura fundamental do modelo relacional é a relação (tabela). Na verdade, o modelo é composto por uma colecção de tabelas de nomes únicos. Cada relação ou tabela é constituída por uma ou mais colunas chamadas de atributos (campos) que são os tipos dos dados contidos na relação. O conjunto de valores passíveis de serem assumidos por um atributo será intitulado de domínio. Cada linha da relação é chamada de tupla (registro).

O esquema de uma relação nada mais é do que os campos (colunas) existentes em uma relação ou tabela. Já a instância da relação consiste no conjunto de valores que cada atributo assume em um determinado instante.

Diferentemente dos modelos que o precederam, o modelo relacional não tem caminhos pré-definidos para se fazer acesso aos dados. Os relacionamentos entre as tabelas são feitos através do uso de valores de atributos.

Para trabalhar com tabelas, algumas restrições precisaram ser impostas para evitar aspectos indesejáveis, como: Repetição de informação, incapacidade de representar parte da informação e perda de informação. Essas restrições são: integridade referencial, chaves e integridade de junções de relações.

O modelo relacional faz uso da linguagem SQL para definição e manipulação de dados. Alguns exemplos de base de dados que fazem uso desse modelo são: PostgreSQL, Oracle, SQLServer, MySQL, entre outros.

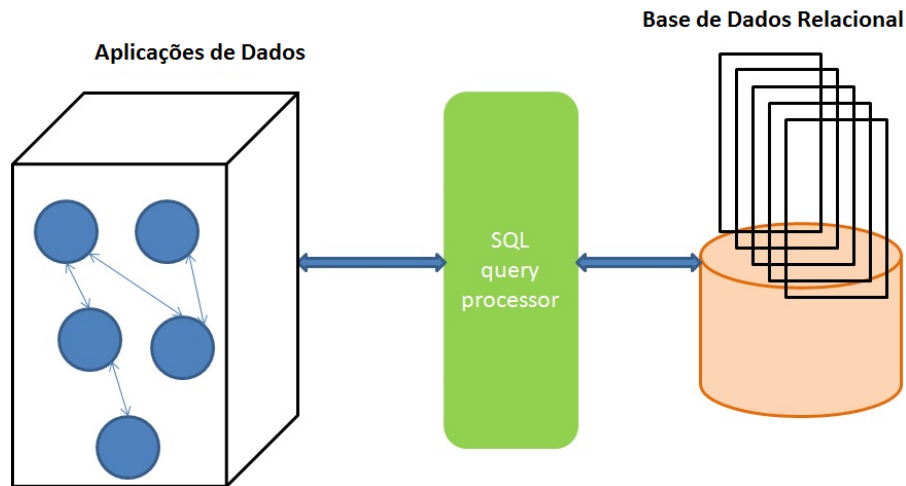


Figura 7: interface entre aplicações e base de dados relacional

ID_Item	Descrição	Preço
111	Livro – Base Dados I	2500
112	Livro – Sistema Operativo II	3000
113	DVD – Cartoon	1750
124	CD – Música Clássica	4000
130	CD – Música RNB	3500
131	Livro – Ciências da Computação	5000
135	Livro – Análise Matemática I	3250
140	Livro – Ciências da Educação	4500
141	Livro – Ciências Geográficas	5000

Tabela 2: tabela items

ID_autor	Nome
9	Jacqueline KONATE
10	Robert OBOKO
11	John KANDIRI
12	Aurelio RIBEIRO
15	Jacira BOMBA
19	Jessica SOGOBA
25	Kiara PIRES
26	Jacques DEMBELE

Tabela 3: tabela Autor

ID_Compras	ID_Autor	ID_Item	Data Compra	Quantidade
602	19	113	10/08/2014	1
603	10	131	10/08/2014	4
701	9	112	12/08/2014	2
702	15	124	13/09/2014	1
703	12	140	03/09/2014	5
704	11	135	01/10/2014	1
705	9	111	12/10/2014	10
708	25	130	13/10/2014	3
710	19	113	13/10/2014	1

Tabela 4: Tabela Compras

Terceira Geração dos Bancos de Dados

Devido às necessidades do mundo moderno de manipular dados complexos, surgiu a terceira geração dos bancos de dados, com os modelos de dados orientados a objectos e os modelos de dados objecto-relacionais. E o que eles trouxeram de novo? Vamos ver a seguir.

Modelo de Dados Orientado a Objectos

Os bancos de dados orientados a objecto começaram a se tornar comercialmente viáveis em meados de 1980. A motivação para seu surgimento está em função dos limites de armazenamento e representação semântica impostas no modelo relacional. Um exemplo de modelo orientado a objectos são os sistemas de informações geográficas (SIG) que são mais facilmente construídos usando tipos complexos de dados. A habilidade para criar os tipos de dados necessários é uma característica das linguagens de programação orientadas a objectos.

Contudo, estes sistemas necessitam guardar representações das estruturas de dados que utilizam no armazenamento permanente. A estrutura padrão para os bancos de dados orientados a objectos foi feita pelo Object Database Management Group (ODMG).

O termo Modelo de Dados Orientado a objectos é usado para documentar o padrão que contém a descrição geral das facilidades de um conjunto de linguagens de programação orientadas a objectos e a biblioteca de classes que pode formar a base para o Sistema de Banco de Dados. Quando os bancos de dados orientados a objectos foram introduzidos, algumas das falhas perceptíveis do modelo relacional pareceram ter sido solucionadas com esta tecnologia e acreditava-se que tais bancos de dados ganhariam grande parcela do mercado.

Hoje, porém, acredita-se que os Bancos de Dados Orientados a objectos serão usados em aplicações especializadas, enquanto os sistemas relacionais continuarão a sustentar os negócios tradicionais, onde as estruturas de dados baseadas em relações são suficientes.

Alguns problemas do modelo orientado a objectos são: ele não tem base teórica (formalismo) como os modelos anteriores e não existe linguagem padronizada para acesso e manipulação dos dados (tal qual o SQL). Isso fez com que esse paradigma não fosse bem aceite no mercado. Como solução surgiu o paradigma objecto-relacional.

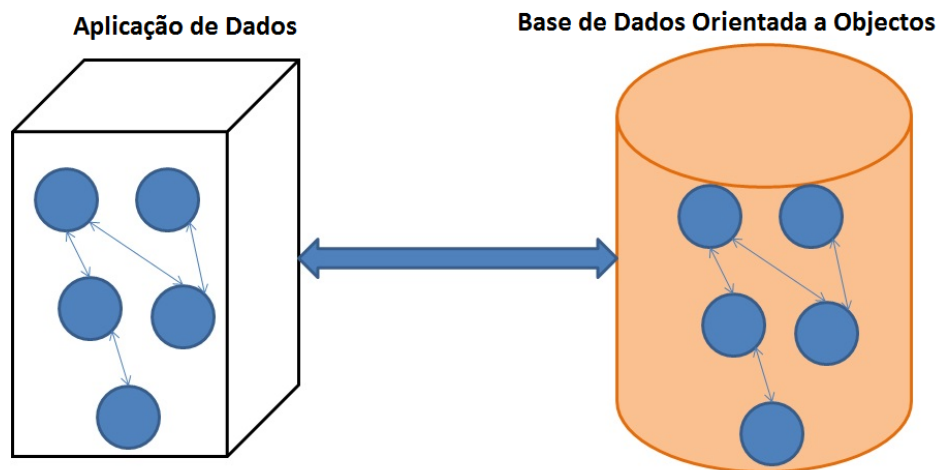


Figure 8: interface entre aplicações e base de dados orientada a objectos

Modelo de Dados Objecto-Relacional

A área de actuação dos sistemas Objecto-Relacional tenta suprir a dificuldade dos sistemas relacionais convencionais, que é o de representar e manipular dados complexos, visando ser mais representativos em semântica e construções de modelagens. Neste modelo de dados, toda a informação persistente está ainda nas tabelas, mas algumas das entradas da tabela podem ter uma estrutura de dados mais rica, denominada tipos abstractos de dados (TAD). Um TAD é um tipo de dado que é construído combinando tipos de dados alfanuméricos básicos. Um suporte para tipos de dados abstractos é atractivo porque as

operações e funções associadas com o novo tipo podem ser usadas para indexar, armazenar, e recuperar registos baseados no índice do novo tipo de dado (por exemplo, multimídia).

Diferente do modelo orientado a objectos, o modelo objecto-relacional foi

desenvolvido com base no modelo relacional. Dessa forma, foi possível unir conceitos de OO (tais como: super-tabelas, super-tipos, herança, reutilização de códigos de criação, encapsulamento, controle de identidade de objectos e referência a objectos) com conceitos do modelo relacional (tais como: capacidade de consulta avançada e a alta protecção a dados). Assim, os modelos Objecto-Relacional combinam os benefícios do modelo Relacional com a capacidade de modelagem do modelo OO.

A linguagem de consulta objecto relacional é uma extensão da linguagem SQL para suportar o modelo de objectos. As extensões incluem consultas envolvendo objectos, atributos multivalorados, TADs, métodos e funções como predicados de busca em uma consulta.

Alguns motivos que levam a utilização desse tipo de modelo são:

- A incapacidade do modelo relacional básico de resolver os desafios e atender as necessidades das novas aplicações;

- A capacidade de armazenar novos tipos de dados;

- Fornecem suporte para consultas complexas sobre dados complexos;

Atendem aos requisitos das novas aplicações e da nova geração de aplicações de negócios;

Algumas aplicações para as quais é necessário o uso desse tipo de modelo são:

Armazenamento de imagens (obtidas por satélite ou de alguma outra forma digital);

Dados complexos não-convencionais em projecto de engenharia;

Grandes informações sobre o genoma biológico;

Projectos de arquitectura;

Dados de séries temporais em transacções;

Dados sobre o espaço (regiões geográficas, criação de mapas);

Sistemas móveis e distribuídos, dentre outros.

Alguns bancos de dados que fazem uso do modelo objecto-relacional são Oracle 9i, DB2/6000, ILUSTRA e CA-Ingres.

Conclusão

Como vimos as primeiras BD surgiram nos anos 60, com o surgimento dos modelos de base de dados hierárquicos, ambos orientados a registos, isto é, qualquer acesso à base de dados – inserção, consulta, alteração ou remoção – era feita em um registo de cada vez. Vamos detalhar agora, cada um desses modelos.

A par destes modelos apresentamos também os Modelo de Dados Relacional; Modelo de Dados Orientado a Objectos e Modelo de Dados Objecto-Relacional.

Avaliação da actividade

Verifique a sua compreensão!

1. Após ter lido o texto apresentado na actividade 2.1 sobre modelo de BD, estabeleça uma diferença entre os dois primeiros modelos.
2. Esboce três tabelas para o modelo relacional.
3. Distribua cada modelo de acordo com a sua geração.

Actividades de Aprendizagem

Actividade 2.2–Modelação de Dados

Introdução

Modelar significa criar um modelo que explique as características de funcionamento e comportamento de um software a partir do qual ele será criado, facilitando seu entendimento e seu projecto, através das características principais que evitarão erros de programação, projecto e funcionamento. É uma parte importante do desenho de um sistema de informação e podem ser utilizadas ferramentas como o UML.

Modelo de Dados

Os modelos de dados são as principais ferramentas que fornecem a abstracção a uma BD, visto que o modelo de dados representa, de forma abstracta, como aspectos do mundo real (fatos) estão relacionados, a fim de que possam ser representados no mundo computacional. Mas o que é um modelo de dados? Ele é um conjunto de conceitos que podem ser usados para descrever a estrutura de uma base de dados. Por estrutura de uma base de dados entende-se os tipos de dados, relacionamentos e restrições pertinentes aos dados que serão armazenados na BD. Muitos modelos de dados também definem um conjunto de operações para especificar como recuperar e modificar a base de dados.

Já o processo pelo qual você planeia ou projecta a base de dados, de forma que possa construir uma base de dados consistente, de forma a exigir menos espaço em disco e aproveitar os recursos disponíveis no SGBD é chamado modelagem de dados. Ou seja, é o processo para a criação do modelo de dados. Mas por que modelar os dados? Qual o objectivo disso? É importante modelar os dados a fim de conhecer melhor as informações dos utilizadores e como elas se relacionam a fim de representar, da melhor forma, o ambiente observado criando, por consequência, bases de dados mais correctas e eficientes. Um projecto mal feito pode trazer diversos problemas, tais como: a base de dados e/ou a aplicação podem não funcionar adequadamente; os dados podem não ser confiáveis ou serão inexactos e a performance da BD pode ser degradada.

A modelagem de dados é uma das etapas do ciclo de Desenvolvimento de Sistemas de Base de Dados (Vide Figura 9).

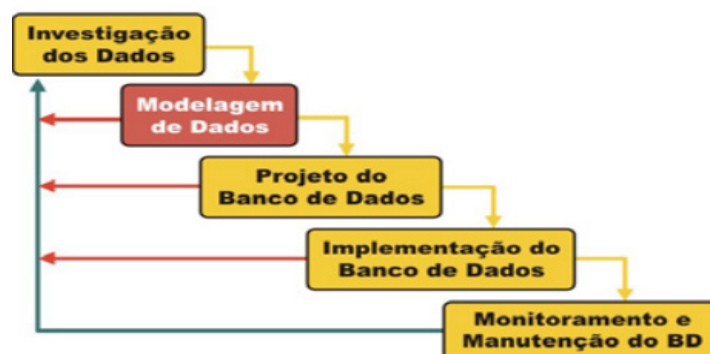


Figura 9. Ciclo de Desenvolvimento de SBDs

E quais são as outras etapas? Primeiro, para poder realizar a modelagem dos dados, você precisa fazer um levantamento de requisitos. Ou seja, precisa investigar quais dados deverão fazer parte do banco de dados, a fim de representar bem o problema a ser resolvido e poder atender as necessidades de armazenamento (persistência) dos dados da aplicação. Uma vez que se saiba os dados que devem ser manipulados, você deve analisar como esses dados podem ser representados e relacionados (olhando para o mundo real) através de um modelo de dados. Esse é o papel da segunda etapa, a modelagem dos dados. Uma vez que os dados estejam modelados a base de dados será projectada, transformando o modelo de dados criado em estruturas de mais baixo nível que possam ser criadas dentro do SGBD. Posteriormente, a BD é realmente implementada usando algum dos SGBDs disponíveis no mercado e, depois, mantida e monitorada pelo administrador de base de dados.

Modelo Conceptual

É a primeira etapa da modelagem de dados, sendo a descrição mais abstracta da realidade, modelando de forma mais natural os fatos do mundo real, suas propriedades e relacionamentos. É utilizado para entendimento, transmissão, validação de conceitos, mapeamento do ambiente e para facilitar o diálogo entre utilizadores e desenvolvedores. A modelagem conceptual da BD não depende da sua implementação, não contendo nenhum detalhe da mesma. Assim, a modelagem conceptual é independente do SGBD utilizado para implementação da BD. De facto, o modelo conceptual regista que dados podem aparecer na base de dados, mas não regista como estes dados estão armazenados em nível de SGBD. A modelagem conceptual da base de dados relacionais é feita através da criação do modelo entidade-relacionamento (MER). No caso de base de dados orientados a objectos ou objecto-relacional, é usado o modelo de classes da UML (Unified Modeling Language).

Modelo Lógico

Representa os dados em alguma estrutura (lógica) de armazenamento de dados, que vai depender do SGBD a ser utilizado. Ou seja, este modelo vai especificar a representação/ declaração dos dados de acordo com o SGBD escolhido, definindo assim a estrutura de registos da BD (onde cada registo define número fixo de campos (atributos) e cada campo possui tamanho fixo). Um exemplo de modelo lógico é o modelo relacional. O modelo relacional usa um conjunto de tabelas para representar tanto os dados como a relação entre eles. Cada tabela possui múltiplas colunas e cada coluna possui um nome único.

Modelo Físico

É usado para descrever os dados no nível mais baixo, tratando de aspectos de implementação do SGBD (como indexação e estruturação de arquivos, controle de concorrência, transacções, recuperação em casos de falhas, entre outros). As linguagens e notações para o modelo físico não são padronizadas e variam de produto a produto (são dependentes do SGBD). Além disso, a tendência dos produtos modernos é cada vez mais esconder o modelo físico.

Todos esses modelos, na verdade, são visões diferentes, com nível de profundidade diferente para os mesmos dados. E é importante saber que, a partir de um modelo, o modelo seguinte pode ser derivado.

Conclusão

Durante esta actividade vimos que modelar significa criar um modelo que explique as características de funcionamento e comportamento de um software a partir do qual ele será criado, facilitando seu entendimento e seu projecto, através das características principais que evitarão erros de programação, projecto e funcionamento. Vimos também que trata-se uma parte importante do desenho de um sistema de informação e podem ser utilizadas ferramentas próprias para se modelar, tais como UML. Ainda nesta unidade falaremos dos modelos de dados lógicos, físicos conceptual, modelo entidade relacionamento, atributos, entidades e relacionamentos.

Avaliação da actividade

Verifique a sua compreensão!

1. Após ter lido o texto apresentado na actividade 2.1 sobre modelo de BD, estabeleça uma diferença entre os dois primeiros modelos.
2. Esboce três tabelas para o modelo relacional.
3. Distribua cada modelo de acordo com a sua geração.

Actividades de Aprendizagem

Actividade 2.3–Modelo Entidade Relacionamento

Introdução

O modelo entidade-relacionamento (conhecido como MER) foi originalmente definido por Peter Chen em 1976 e é baseado na teoria relacional criada em 1970 por Codd. Posteriormente, na década de 80, o MER sofreu algumas extensões para tornar-se mais preciso na representação do mundo real. Actualmente, ele é o modelo de dados conceptual mais difundido e utilizado para modelagem de base de dados relacionais.

Para iniciar o projecto conceptual da BD, deve ter sido antes definido qual o problema a ser resolvido, ou seja, deve-se ter determinado as fronteiras que delimitam e restringem o mini-mundo a ser modelado e realizado a especificação desse mini-mundo. Tudo isso faz parte da etapa de análise dos requisitos. A partir justamente da especificação feita é que você irá extrair as informações necessárias para desenhar a primeira versão do MER.

Segundo Silberschatz (1999), o MER tem por base a percepção de que o mundo real é formado por um conjunto de objectos chamados entidades e pelo conjunto dos relacionamentos entre esses objectos. Na verdade, existem três noções básicas empregadas pelo MER: conjunto de entidades, conjunto de relacionamentos e conjunto de atributos. Só para ilustrar, na Figura 10 apresentamos um pequeno exemplo de MER onde estão representadas as entidades cliente e conta, cada uma com seus atributos. As duas entidades se relacionam através do relacionamento cliente-conta.



Figura 10: Um pequeno exemplo de MER

Componentes do Modelo Entidade-Relacionamento

Entidades

O objecto básico que o MER representa é a entidade. Uma entidade é algo do mundo real que possui uma existência independente. Uma entidade pode ser um objecto com uma existência física (por exemplo, uma pessoa, um DVD, um carro ou um livro), nesse caso é chamada entidade concreta. Ou pode ser um objecto com existência conceptual (por exemplo, uma locação, um curso, um empréstimo ou um projecto), nesse caso é chamada entidade abstracta. Em outras palavras, é um objecto concreto ou abstracto da realidade modelada, sobre o qual se deseja manter informações na BD. Graficamente, entidades são

representadas por um rectângulo com um nome dentro (vide Figura 6). Esse nome deve vir no singular e deve ser representativo.



Figura10 - Exemplos de entidades

É importante que toda entidade criada seja descrita em um dicionário de dados.

Mas o que é um dicionário de dados? Ele é um documento com a explicação de todos os objectos criados na base de dados (entidades, atributos e relacionamentos). Ele permite que os analistas obtenham informações sobre todos os objectos do modelo de forma textual, contendo explicações por vezes difíceis de incluir no diagrama. A maioria dos SGBDs

actuais já fornecem ferramentas para facilitar a inclusão de informações no dicionário de dados, deixando assim os objectos criados bem melhor documentados (você vai conseguir saber exatamente quem é quem e para quê serve). Nesta etapa de definição da entidades, a informação possível de colocar no dicionário é apenas a descrição da entidade. Na tabela 5, você pode ver exemplos de como poderia ser a descrição das entidades Empregado e Encomenda em um dicionário de dados.

Entidade Descrição	Empregado	Encomenda
	Pessoa que mantém vínculo de empregado com a Empresa através de um contrato de trabalho de acordo com a legislação trabalhista.	Instrumento contratual de emissão unilateral pela empresa e aceitação, expressa ou tácita, pelo fornecedor do material

Tabela 5: Exemplo de Descrição de Entidade em um Dicionário de Dados

Atributos

São propriedades descritivas de cada membro/propriedade de uma entidade ou relacionamento. Em outras palavras, uma entidade sempre é representada por um conjunto de atributos. Cada instância de uma entidade ou relacionamento tem seu próprio valor para cada atributo. Por exemplo, o atributo nome do empregado pode ter os valores Ana, Maria, João, Igor, etc. O conjunto de valores permitidos para cada atributo é chamado de domínio (é a definição do tipo do atributo). Por exemplo:

nome = texto com 60 posições

conta = inteiro com 8 posições

consulta = texto com 8 posições

Os atributos podem ser dos seguintes tipos: simples, compostos, monovalorados, multivalorados, nulos e derivados.

Simples– os atributos simples não podem ser divididos, são atômicos. Por exemplo Salário, idade, entre outros.

Compostos – os atributos compostos podem ser divididos em partes (ou seja, podem ser quebrados em outros atributos mais simples/elementares). Por exemplo: endereço pode ser dividido em rua, número, bairro e Caixa postal. O atributo NomeCliente pode se dividido em prenome, nome_Intermediário e sobrenome. Atributos compostos são usados quando o usuário desejar se referir ao atributo como um todo em certas ocasiões e somente a parte dele em outras. Se o atributo composto for sempre referenciado como um todo, não existe razão para subdividi-lo em componentes elementares.

Monovalorados – Os atributos monovalorados possuem apenas um valor por entidade. Por exemplo: o atributo BI de uma entidade Pessoa refere-se apenas a um nº de BI é, então, monovalorado.

Multivalorados – Atributos multivalorados podem possuir um conjunto de valores para uma única entidade. Por exemplo: na entidade do Cliente pode existir um atributo telefone e um cliente pode ter cadastrado um, nenhum ou vários telefones (ex: telefone residencial, comercial e celular). Atributos multivalorados podem possuir uma multiplicidade, indicando as quantidades mínima e máxima de valores que ele pode assumir.

Nulos – Um atributo nulo é usado quando uma entidade não possui valor para um determinado atributo. Nulo significa que o valor do atributo é vazio (não-aplicável) ou ainda não foi informado (desconhecido). Por exemplo atributo complemento (do endereço) que, geralmente, é utilizado para colocar o número do apartamento onde alguém, tal como um cliente, mora. Porém, se a pessoa mora em casa, esse campo não é preenchido, ficando nulo.

O atributo nulo pode, também, ser utilizado para denotar que o valor é

desconhecido, como por exemplo, quando o cliente em um cadastro não responde o número da Caixa Postal da rua onde reside. A Caixa Postal existe e é aplicável, apenas o cliente pode não saber a Caixa Postal naquele momento. Logo, nos primeiros exemplos dados aqui, o significado do uso do nulo era “não aplicável” e, neste último exemplo, o significado do nulo é “desconhecido”.

Derivados – O valor desse tipo de atributo pode ser derivado de outros atributos ou entidades a ele relacionados. Por exemplo: suponha que se deseje colocar na entidade Pessoa o campo idade. Como ele está relacionado com o campo data de nascimento de uma pessoa, a idade não precisaria ser explicitamente preenchida. Seu valor poderia ser derivado da data de nascimento, fazendo a seguinte conta: data atual menos a data de nascimento. O uso de atributos derivados é uma decisão de projecto, dependendo da necessidade e do bom senso. Os atributos são representados no MER por elipses (contendo o nome dos atributos) ligadas às entidades ou relacionamentos (sim, relacionamentos também podem possuir atributos, falaremos sobre isso na próxima seção!) aos quais estes atributos pertencem. No exemplo da Figura 11, temos as entidades CLIENTE e CONTA. A entidade CLIENTE tem os atributos RG, nome, cidade e endereço e a entidade CONTA, os atributos número, saldo e data. Atributos Derivados é que possuem uma representação diferenciada, sendo representados por elipses tracejadas. E atributos multivalorados são representados por elipses duplas (uma elipse dentro da outra).

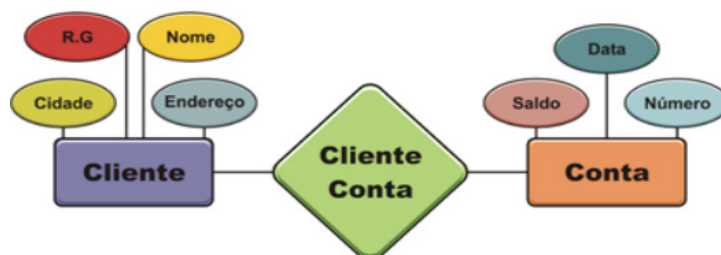


Figura 11: Exemplo de MER com os atributos sendo representados

Identificador de Entidade

Identificador de entidade é um (simples) ou mais (composto) atributos cujos valores identificam unicamente uma entidade. Ou seja, o identificador deve possuir um valor único para cada entidade, não admitindo valores repetidos do atributo (ou dos atributos) que o compõem.

Por convenção, o atributo identificador é representado sempre de forma diferenciada dos outros atributos. Na notação que vimos anteriormente, ele seria representado sublinhado. E na notação de Peter Chen usa-se um círculo preto para o atributo identificador e círculos brancos para o restante dos atributos. Por exemplo, na Figura 12 (lado esquerdo), vemos a entidade CLIENTE onde o atributo Nr. Cliente é o identificador da entidade (como é um único atributo, é um identificador simples). E na Figura 12 (lado direito), temos que a entidade ALUNO tem os atributos código e nome, sendo que o código é o atributo identificador da entidade.

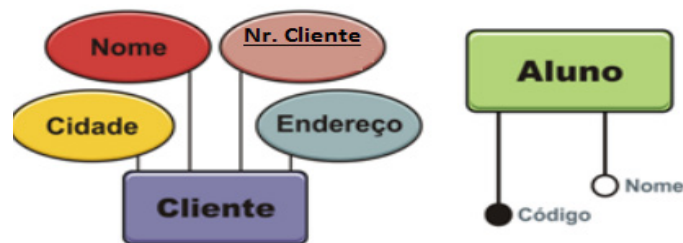


Figura 12: Atributos identificadores simples

Na Figura 13 vemos exemplos de identificadores compostos. Um identificador composto possui dois ou mais atributos como identificadores da entidade. Em ambos os desenhos da Figura 10 está representada a entidade PRATELEIRA que tem como identificador os atributos corredor e armário e um atributo extra chamado capacidade.

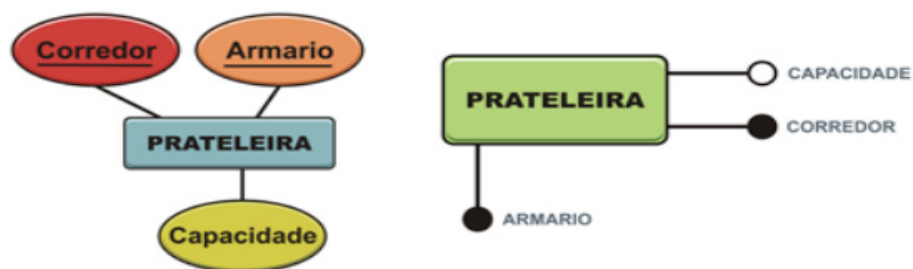


Figura 13: Exemplo de entidade PRATELEIRA com identificador composto

(notação convencional e notação Heuser)

Na prática, você verá que a maioria dos MER não apresenta os atributos. Por que será? Os atributos, em geral, não são representados no MER para não sobrecarregar (graficamente) a apresentação do diagrama. Isso deixa o diagrama entidade-relacionamento mais legível. Eles, geralmente, são apresentados em uma representação textual à parte do diagrama E-R.

Relacionamentos

São associações entre uma ou várias entidades. Em outras palavras, são funções que mapeiam um conjunto de instâncias de uma entidade em outro conjunto de instâncias de outra entidade

(ou da mesma entidade, como é o caso do “auto-relacionamento”). Vamos dar um exemplo: “departamento emprega funcionário” (vide Figura 14). Departamento e Funcionário seriam entidades ligadas através do relacionamento emprega, sendo representado na figura por um losango com o nome do relacionamento.

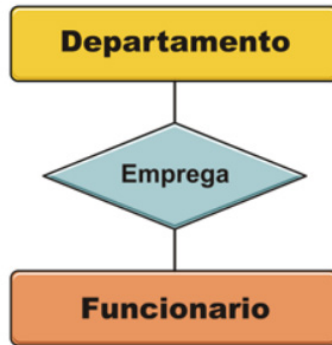


Figura 14: Exemplo de relacionamento

Cardinalidade do Relacionamento

A cardinalidade caracteriza o número mínimo e máximo de instâncias de cada entidade que podem estar associadas através do relacionamento. Dado um relacionamento R entre as entidades A e B (vide Figura 15), o grau ou cardinalidade especifica valores que vão responder às seguintes perguntas:

1. Com quantos elementos de B se relaciona cada um dos elementos de A?
2. Dado um elemento de B, com quantos elementos de A ele se relaciona?



Figura 15: Relacionamento de entidades A e B

Assim, as cardinalidades podem ser dos seguintes tipos:

Um para um (1:1): Uma entidade de A está associada, no máximo, a uma entidade de B, e uma entidade de B está associada a, no máximo, uma entidade de A. É como se cada instância da entidade A só encontrasse uma instância correspondente na entidade B (vide Figura 16).

Por exemplo, “Pessoa recebe Certidão de nascimento”. Cada pessoa só pode ter uma única certidão de nascimento (só se nasce uma vez, não é?) e uma certidão de nascimento só deve pertencer a uma única pessoa (vide Figura). Logo, esse relacionamento é dito um para um.



Figura 17: Exemplo de relacionamento um para um

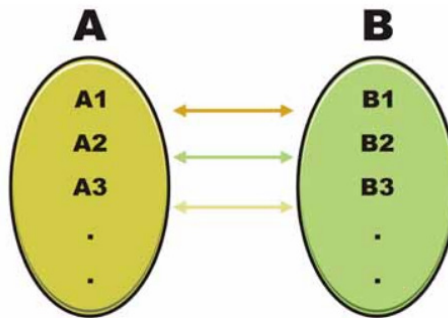


Figura 16- Esquema de um relacionamento 1:1

Um para muitos (1:N): Uma entidade em A está associada a várias entidades em B. Uma entidade em B, entretanto, deve estar associada a, no máximo, uma entidade em A. É como se cada instância da entidade A encontrasse zero ou mais instâncias correspondentes na entidade B, porém, cada instância da entidade B só encontrasse uma única instância correspondente em A (vide Figura 18). Por exemplo, “Empresa possui Filial”. Cada empresa pode possuir zero ou mais filiais (ou seja, N filiais, porque o N significa 0, 1, 2 ou mais). Mas uma filial só pertence a uma única empresa (vide Figura 19).

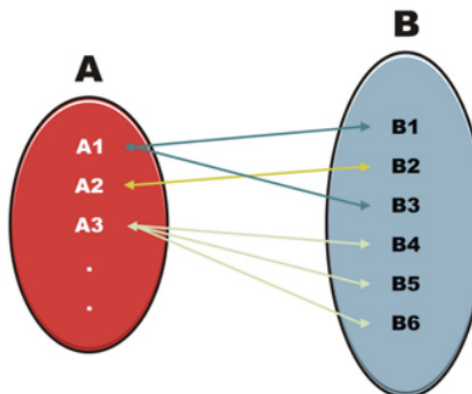


Figura 18: Esquema de relacionamento 1:N



Figura 19: Exemplo de relacionamento 1:N

Muitos para um: é o contrário da anterior. Aqui, uma entidade em A está associada a, no máximo, uma entidade em B. E uma entidade em B pode estar associada a zero, uma ou mais entidades (N entidades) em A. É como se cada instância da entidade A encontrasse apenas uma instância correspondente na entidade B. Mas, cada instância da entidade B encontrasse zero ou mais instâncias correspondentes na entidade A (vide Figura 20). Por exemplo, “Aluno é orientado por Professor”. Um aluno só é orientado por um professor (de repente, é regra da faculdade onde ele estuda), mas um professor pode orientar zero ou mais alunos (N alunos), conforme pode ser visto na Figura 21.

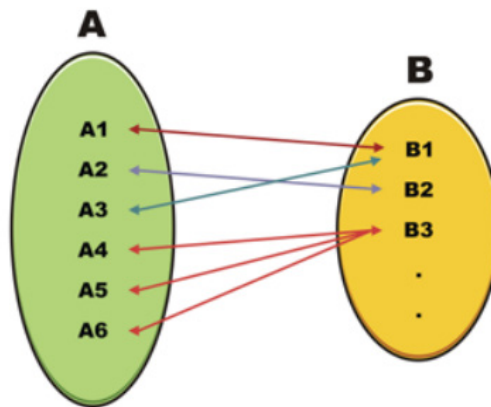


Figura 20: Esquema de relacionamento N:1



Figura 21: Exemplo de relacionamento N:1

Muitos para Muitos (M:N): Uma entidade em A está associada a qualquer número de entidades em B e uma entidade em B está associada a um número qualquer de entidades em A. É como se cada instância da entidade A encontrasse zero, um ou mais instâncias correspondentes na entidade B. E cada instância da entidade B encontrasse zero, uma ou mais instâncias correspondentes na entidade A (vide Figura 22). Por exemplo, "Aluno cursa Disciplina", um aluno cursa zero, uma ou mais disciplinas e uma disciplina é cursada por zero, um ou mais alunos.

Outro exemplo é "Médico consulta Paciente". Um médico consulta zero, um ou mais Pacientes. E um Paciente é consultado por zero, um ou mais médicos (por exemplo, a pessoa pode ter ido a um endocrinologista e em um cardiologista), conforme pode ser visto no diagrama exemplo da Figura 23.

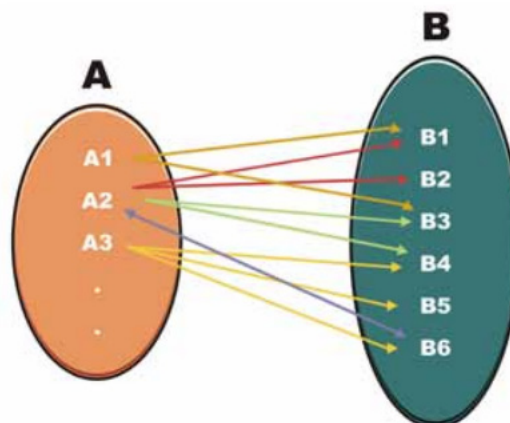


Figura 22: Esquema de relacionamento M:N



Figura 23: Relacionamento M:N

Notação de Peter Chen para Cardinalidade

A notação de Chen faz uso de dois tipos de cardinalidade: mínima e máxima. Essas cardinalidades são representadas por: (cardinalidade mínima, cardinalidade máxima).

A cardinalidade mínima expressa o número mínimo de ocorrências de determinada entidade associada a uma ocorrência da entidade em questão através do relacionamento.

Para fins de projecto define-se a cardinalidade mínima como 1 ou 0, onde :

= associação obrigatória (indica que o relacionamento deve obrigatoriamente associar uma ocorrência de entidade a cada ocorrência da outra entidade a qual se relaciona). Também chamada de relacionamento total.

0 = associação opcional (indica que o relacionamento não é obrigatório, ou seja, é opcional associar uma ocorrência de entidade a ocorrência da outra entidade a qual se relaciona). Também chamada de relacionamento parcial.

A cardinalidade máxima expressa o número máximo de ocorrências de determinada entidade, associada a uma ocorrência da entidade em questão, através do relacionamento. Para fins de projecto define-se como 1 ou N.

Por exemplo, na Figura 24, temos o relacionamento "Empregado possui Dependente".

A cardinalidade (0,N) representa que um empregado pode ter 0 ou mais dependentes, sendo 0 (zero) a cardinalidade mínima e N a cardinalidade máxima. Justamente porque essa cardinalidade representa que o empregado pode não ter nenhum dependente, dizemos que a associação é opcional. Já a cardinalidade (1,1) representa que um dependente deve ter no mínimo 1 e no máximo 1 empregado ao qual está filiado. Aqui, como se for criado um dependente deve existir no mínimo 1 Empregado, dizemos que a associação é obrigatória. Não pode existir dependente sem uma associação a um empregado. Ou seja, uma ocorrência de empregado pode não estar associada a uma ocorrência de dependente ou pode estar associada a várias ocorrências dele (determinado empregado pode não possuir dependentes ou pode possuir vários). E uma ocorrência de dependente está associada a apenas uma ocorrência de empregado (cada dependente possui apenas um empregado responsável).



Figura 24: Exemplo de uso da cardinalidade na notação de Peter Chen

Conclusão

Durante esta actividade vimos que modelar significa criar um modelo que explique as características de funcionamento e comportamento de um software a partir do qual ele será criado, facilitando seu entendimento e seu projecto, através das características principais que evitarão erros de programação, projecto e funcionamento. Vimos também que trata-se uma parte importante do desenho de um sistema de informação e podem ser utilizadas ferramentas próprias para se modelar, tais como UML.

Avaliação da actividade

Verifique a sua compreensão!

1. Crie uma relação com três entidades 1:N.
2. Dê exemplo de dois atributos multivalorados.
3. Qual é a solução para uma relação muito para muito.

Resumo da Unidade

Caro estudante, nesta unidade apresentamos duas actividades, respectivamente, Apresentação e descrição dos tipos de modelos de BD e Modelação de Dados. Portanto após ler o texto que acompanha estas actividades de aprendizagem, procure fazer um resumo em que deve incluir os termos chaves.

Avaliação da Unidade

Verifique a sua compreensão!

1. Defina o conceito Entidades.
2. O são Atributos?
3. Quando é que podemos considerar um atributo multivalorado?
4. O são Relacionamentos?

Soluções

1. Uma entidade é algo do mundo real que possui uma existência independente. Uma entidade pode ser um objecto com uma existência física ou conceptual.
2. São propriedades descritivas de cada membro/propriedade de uma entidade ou relacionamento. Em outras palavras, uma entidade sempre é representada por um conjunto de atributos. Cada instância de uma entidade ou relacionamento tem seu próprio valor para cada atributo.
3. Quando possuem um conjunto de valores para uma única entidade. Por exemplo: na entidade do Cliente pode existir um atributo telefone e um cliente pode ter cadastrado um, nenhum ou vários telefones.

4. São associações entre uma ou várias entidades. Em outras palavras, são funções que mapeiam um conjunto de instâncias de uma entidade em outro conjunto de instâncias de outra entidade (ou da mesma entidade, como é o caso do “auto-relacionamento”).

Leituras e outros Recursos

Leituras e outros recursos obrigatórios:

Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431

Banco de Dados- Vol II, Sandra de Albuquerque SIEBRA, Recife, 2010

Unidade 3. Normalização de Dados

Introdução à Unidade

Projectar uma base de dados relacional significa agrupar atributos para formar bons esquemas de relações. Mas o que seria um bom esquema? Em nível geral, poderíamos dizer que seria um esquema fácil de entender e em que as tuplas da relação fossem armazenadas e acedidas de forma eficiente. Para isso, é preciso que sejam minimizadas, ao máximo, a redundância nos dados e as anomalias de inserção, actualização e exclusão. Além disso, é preciso garantir a integridade dos dados, evitando que informações sem sentido sejam inseridas e é preciso organizar e dividir as tabelas da forma mais eficiente possível. Uma forma de colaborar com essas necessidades é fazer a normalização dos dados. Esse é justamente o assunto desta unidade.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Explicar as dependências funcionais;
- Normalizar os dados pelo menos, até a 3ª Forma Normal;
- Diferenciar as três primeiras formas normais.

Termos-chave

Dependências funcionais, Normalização de dados, Formas normais.

Dependências Funcionais: Verifica-se quando um atributo A identifica um atributo B, dizemos que entre eles há uma dependência funcional. Temos, portanto, que A é o determinante e que B é o dependente.

Normalização de dados: é o processo que consiste em organizar uma base de dados com objectivo de eliminar anomalias e futuros problemas na Base de Dados.

Formas normais: São os passos que se seguem no processo de normalização. Cada forma normal tem os seus algoritmos.

Actividades de Aprendizagem

Actividade 3.1 – Dependências funcionais

Introdução

As dependências funcionais e as formas normais estabelecem critérios de qualidade de desenho no modelo Relacional. Permitem detectar e prevenir a redundância e garantir a coerência da informação.

Entretanto ao nível do modelo Relacional, deve ser usado para detectar eventuais problemas associados à redundância e coerência da informação. Ao nível do modelo Entidade Associação, deve ser usado para identificar entidades e determinar os seus atributos.

Para realizar esta actividade deverá ler o texto que se segue:

Dependências Funcionais

Sempre que um atributo X identifica um atributo Y dizemos que entre eles há uma dependência funcional. Temos, portanto, que X é o determinante e que Y é o dependente.

A representação é: $X \twoheadrightarrow Y$. Em outras palavras, se o valor de um atributo X permite descobrir o valor de um outro atributo Y, dizemos que X determina funcionalmente Y. Por exemplo, dada uma determinada Cidade sabemos a sua Província e com a Província temos o País. Isso é representado da seguinte forma:

Cidade \rightarrow Província

Província \rightarrow País

Em outras palavras, Província é funcionalmente dependente de Cidade e país é funcionalmente dependente da Província. Ou ainda, Cidade determina Província e Província determina País.

Logo, a dependência funcional é caracterizada pela existência de campos em uma determinada tabela relacional cuja ocorrência de valores está associada a valores que são preenchidos em outros campos na mesma tabela.

Por exemplo, suponha uma tabela EMPREGADO que possui dois atributos Cod_Empregado e Nome. O atributo Nome é funcionalmente dependente do atributo Cod_Empregado. Assim, $\text{Cod_Empregado} \twoheadrightarrow \text{Nome}$. Com isso, queremos dizer que Nome é função do Cod_Empregado, ou seja, se eu tiver um número de Cod_Empregado, poderei encontrar o Nome da pessoa correspondente. Em outras palavras, Cod_Empregado determina o Nome (vide Figura 25).

Cod_Empregado

Nome

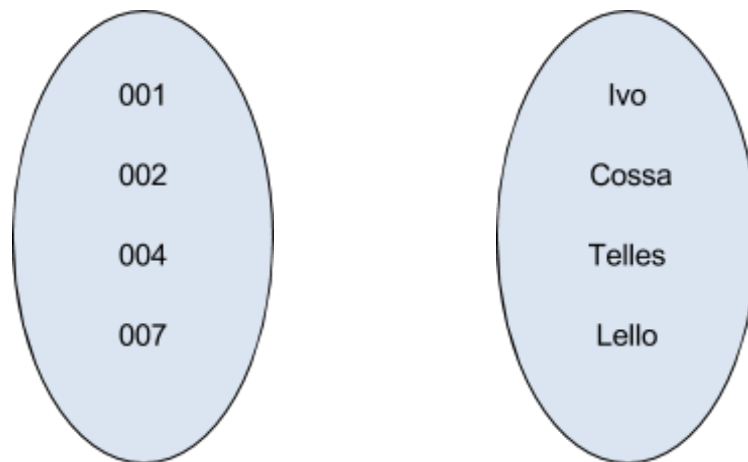


Figura 25: Cod_Empregado determina o Nome

Para efectuar a normalização de um modelo de dados relacional, como veremos daqui a pouco, alguns tipos de dependências funcionais são de extrema relevância:

Dependência Parcial – Ocorre quando a chave primária é composta e existe um campo da relação que depende somente de parte desta chave primária composta. Por exemplo, veja a Relação Alocação (vide Tabela 12). Ela possui a chave composta Cod_Empregado e Cod_Projecto. O ideal seria que todos os campos (atributos) da relação dependessem (fossem funcionalmente dependentes) da chave primária total, completa. Porém, o campo Nome_Empregado depende apenas de parte da chave primária (no caso, do Cod_Empregado). O mesmo ocorre com o atributo Nome_Projecto que só depende do atributo Cod_Projecto. Apenas o atributo Horas_Trabalhadas é funcionalmente dependente da chave primária completa (porque para determinar as horas trabalhadas, precisamos saber Cod_empregado e para qual projecto ele está trabalhando através do Cod_projecto).

<u>CPF_Empre-</u> <u>gado</u>	<u>Cod_Projecto</u>	<u>Nome_Empre-</u> <u>gado</u>	<u>Nome_Pro-</u> <u>jecto</u>	<u>Horas_Trabalha-</u> <u>das</u>
001	10	Ivo	GIS	23
002	11	Cossa	Floresta	24
003	12	Kiara	Pontes	30

Tabela 6 - Relação Alocação

Dependência Transitiva – Ocorre quando uma coluna depende não somente da chave primária da tabela, mas também de uma segunda coluna (ou conjunto de colunas) da tabela, que não fazem parte da chave primária. Em outras palavras, a dependência funcional transitiva é a dependência funcional indirecta existente entre dois ou mais atributos. Assim, se um atributoC depende funcionalmente de um atributoB e o atributoB depende funcionalmente de um atributoA, então diz-se que o atributoC depende indirectamente (transitivamente) do atributoA (vide Figura 26).

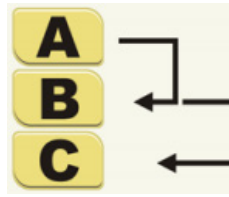


Figura 26 - Exemplo de dependência transitiva

Por exemplo, a relação Pedido_Venda (vide Tabela 13) tem como chave primária o atributo Cod_Pedido. Os atributos Data_Pedido, Situação_Pedido e Cod_Cliente dependem funcionalmente dessa chave primária. Porém, o atributo Nome_Cliente depende funcionalmente do Cod_Cliente (que não é a chave primária) e, apenas, indirectamente do Cod_Pedido, o que é uma anomalia, visto que, todos os atributos de uma relação deveriam depender funcionalmente apenas da sua chave primária.

<u>Cod_Pedi-</u> <u>do</u>	Data Pedido	Situação_Pedido	Cod_Clien-	Nome_Clien-
1	13/11/2014	Entregue	001	Aurélio
2	04/12/2014	Pendente	002	Jacira
3	14/05/2014	Entregue	003	Kiara

Tabela 7 - Relação Pedido_Venda

Dependência Multivalorada – Ocorre quando o valor de um atributo determina um conjunto de valores de um outro atributo. Por exemplo, até agora vimos que um atributo (que deve ser a chave primária da relação) determina outro atributo:

Cod_empregado → Nome (ou seja, o Cod_empregado determina o nome, sendo um nome para cada Cod_empregado).

Porém, se considerarmos: Cod_empregado → Dependente teremos um problema para expressar a realidade, porque através de um Cod_empregado poderia ocorrer o caos de mais de um dependente ser determinado e não apenas um. Isso caracteriza uma dependência multivalorada. Uma dependência multivalorada é representada por X ⇨ Y (que quer dizer, X multidetermina Y ou Y é multidependente de X).

Uma dependência funcional (DF) é uma propriedade do esquema da relação e não de uma instância particular da relação (tupla). Assim, uma DF não pode ser automaticamente inferida a partir de algumas tuplas da relação, mas deve ser definida por alguém que conheça a semântica dos atributos da relação. Isso, porque a DF deve ser válida para todas as tuplas de uma relação, ou seja, para a definição do esquema da relação como um todo.

Anomalias de Actualização

A mistura de atributos de várias entidades pode gerar problemas conhecidos como anomalias de actualização e essas anomalias podem, por sua vez, causar problemas tais como a ocorrência de:

- Grupos repetitivos de dados;
- Dependências parciais de chave;
- Redundâncias desnecessárias de dados;
- Perdas acidentais de informações;
- Dificuldade de representação de factos da realidade (modelos); e
- Dependências transitivas entre atributos.
- As anomalias de actualização podem ser de 3 tipos:
- Anomalias de inserção – Causam a repetição desnecessária de dados (redundância);
- Anomalias de alteração – Levam as inconsistências e aumentam o esforço para a actualização dos dados;
- Anomalias de exclusão - Causam a perda de informações associadas a um dado registo.

Conclusão

Nesta secção abordamos o conceito de dependências funcionais, isto é, os tipos de dependências que existem, as anomalias de actualização, por forma a podermos entrar nas formas normais e projectar uma base de dados ideal (possível de ser implementada com menos redundância quanto necessária).

Avaliação da actividade

Verifique a sua compreensão!

1. O que são dependências funcionais?
2. Defina dependência funcional transitiva.
3. Liste três anomalias de actualização.

Actividades de Aprendizagem

Actividade 3.2 – O que é Normalização?

Introdução

Em 1970, o Professor Dr. Edgar F. Codd, analista da IBM, desenvolveu uma série de estudos sobre como tratar os dados, a fim de eliminar as anomalias de actualização e as suas consequências desagradáveis para as organizações. Deste esforço surgiram duas inovações que revolucionaram a maneira de tratar os dados. A primeira destas inovações foi o Modelo Relacional, no qual se basearam os SGBD da metade da década de 1980 e início de 1990. A segunda inovação foi o processo de Normalização, sendo que ambas estão intimamente relacionadas.

O processo de normalização como foi proposto por Codd sujeita um esquema de relação a uma série de testes para certificar-se de que ele satisfaça certa forma normal.

Na verdade, a normalização de dados pode ser vista como o processo de análise de determinados esquemas de relações com base em suas dependências funcionais e chaves primárias para alcançar as propriedades desejáveis de: (1) minimização de redundâncias e (2)

minimização de anomalias de inserção, exclusão e alteração. Para aprender mais sobre normalização leia o que se segue e, como já é costume depois avalie a sua aprendizagem.

Assim, podemos dizer que o processo de normalização tem as seguintes funções:

- Analisar tabelas e organizá-las de forma que a sua estrutura seja simples, relacional e estável, para que a gestão delas possa ser também simples, eficiente e segura;
- Evitar a perda e a repetição da informação;
- Conseguir uma forma de representação adequada para o que se deseja armazenar;
- Oferecer mecanismos para analisar o projecto d2 BD (identificação de erros e possibilidades de melhorias) e oferecer métodos para corrigir problemas que, por ventura, sejam encontrados.

E, com essas funções, pretende-se alcançar os seguintes Objectivos:

- Garantir a integridade dos dados, evitando que informações sem sentido sejam inseridas na BD;
- Organizar e dividir as tabelas da forma mais eficiente possível, diminuindo a redundância e permitindo a evolução da BD com o mínimo de efeito colateral.
- Inicialmente Codd propôs três formas normais que ele chamou de primeira (1FN), segunda (2FN) e terceira (3FN) forma normal. Uma definição mais forte da 3FN (chamada forma normal BOYCE-CODD - FNBC ou BCNF) foi depois proposta por Boyce e Codd. Todas
- essas formas normais são baseadas nas dependências funcionais entre os atributos de uma relação. Posteriormente, uma quarta (4FN) e quinta (5FN) forma normal foram propostas.

As formas normais são aplicadas para evitar redundância de dados, inconsistências e anomalias de actualização. Isso porque, redundância de dados é um fato gerador de inconsistências. Já as anomalias de actualização criam dificuldades operacionais para manutenção da BD, quebrando a confiabilidade no dado ou ferindo a integridade dos dados.

Uma forma normal engloba todas as outras anteriores, ou seja, a hierarquia entre as formas normais indica que uma tabela só pode estar numa forma mais avançada se, além de atender as condições necessárias, já estiver na forma normal imediatamente anterior.

Por exemplo, para que uma tabela esteja na 2FN, ela obrigatoriamente deve estar na 1FN e assim por diante.

Na prática, o mais comum é normalizar relações apenas até a 3FN. Isso porque as três primeiras formas normais (1FN, 2FN e 3FN) atendem à maioria dos casos de normalização e já são suficientes para organizar as tabelas de uma BD.

Apresentaremos cada uma das formas normais nas seções a seguir, ilustrando cada uma delas com exemplos:

A exemplificação da teoria da normalização vai ter por base uma relação que pretenda representar uma relação que pretenda implementar as turmas a que um professor dá aulas.

Professor (#NrProfessor, Nome, Morada, Cpostal, Localidade, DirectorTurma, CodGrupoDisciplinar, NomeGrupo, CodTurma, DescriçãoTurma).

1ª FORMA NORMAL

Uma tabela encontra-se na primeira forma normal se:

- Todos os seus atributos ou colunas estiverem definidos em domínios que contenham apenas valores atômicos.
- Isto significa que um atributo só pode admitir valores elementares e não conjunto de valores.

UMA RELAÇÃO ESTÁ NA 1ª FORMA NORMAL SE NÃO TEM GRUPOS DE ATRIBUTOS REPETITIVOS.

A relação Professor não se encontra na 1FN porque os atributos DirectorTurma, CodGrupoDisciplinar, NomeGrupo, CodTurma, DescriçãoTurma constituem um grupo repetitivo, ou seja, sempre que um professor dá aulas a mais que uma turma é necessário repetir os atributos NrProfessor, Nome, Morada, Cpostal, Localidade.

Passagem para a 1FN:

Criar uma nova relação Turmas

Professor (#NrProfessor, Nome, Morada, Cpostal, Localidade,

CodGrupoDisciplinar, NomeGrupo)

Turma (#NrProfessor, #CodTurma, DirectorTurma, DescriçãoTurma)

2ª FORMA NORMAL

UMA RELAÇÃO ESTÁ NA 2FN SE ESTÁ NA 1FN E SE TODOS OS ATRIBUTOS NÃO CHAVE DEPENDEM DA TOTALIDADE DA CHAVE.

A relação Professor está na 2FN.

A relação Turma não se encontra na 2FN porque os atributos DirectorTurma e DescriçãoTurma não dependem do NrProfessor, mas só do CodTurma.

Passagem para a 2FN:

Criar uma nova relação Turma/Professor

Professor (#NrProfessor, Nome, Morada, Cpostal, Localidade,

CodGrupoDisciplinar, NomeGrupo)

Turma (#CodTurma, DirectorTurma, DescriçãoTurma)

Turma/Professor (#NrProfessor, #CodTurma)

As três relações encontram-se na 2FN.

3ª FORMA NORMAL

UMA RELAÇÃO ESTÁ NA 3FN SE ESTÁ NA 2FN E SE NENHUM ATRIBUTO NÃO CHAVE DEPENDER FUNCIONALMENTE DE ALGUM OUTRO ATRIBUTO QUE NÃO SEJA CHAVE.

As relações Turma e TurmaProfessor estão na 3FN.

A relação Professor não se encontra na 3FN porque o atributo NomeGrupo não depende do NrProfessor, mas apenas do CodGrupoDisciplinar.

Professor (#NrProfessor, Nome, Morada, Cpostal, Localidade,

CodGrupoDisciplinar)

Turma (#CodTurma, DirectorTurma, DescriçãoTurma)

Turma/Professor (#NrProfessor, #CodTurma)

GrupoDisciplinar (#CodGrupoDisciplinar, NomeGrupo)

As quatro relações anteriores estão na 3FN.

FNN	1FN	2FN	3FN
	Professor	Professor	Professor
#NrProfessor Nome Morada CPostal Localidade DirectorTurma CodGrupoDisciplinar NomeGrupo CodTurma DescriçãoTurma	#NrProfessor Nome Morada CPostal Localidade CodGrupoDisciplinar NomeGrupo	#NrProfessor Nome Morada CPostal Localidade CodGrupoDisciplinar NomeGrupo	#NrProfessor Nome Morada CPostal Localidade CodGrupoDisciplinar
	Turma/Professor	Turma/Professor	Turma/Professor
	#NrProfessor #CodTurma DirectorTurma DescriçãoTurma	#NrProfessor #CodTurma	#NrProfessor C#odTurma
		Turma	Turma
		#CodTurma DirectorTurma DescriçãoTurma	#CodTurma DirectorTurma DescriçãoTurma
			GrupoDisciplinar
			#CodGrupoDisciplinar NomeGrupo
Escolha da chave primária que identifica cada ocorrência de preenchimento do documento.	Remover o grupo repetitivo, acompanhados da chave primária.	Remover itens que dependem apenas de uma parte não plena da chave, acompanhados da parte parcial da chave, que os identifica.	Remover itens que dependem de outros itens e não da chave da relação.

Tabela 8 – Resumo de Formas Normais

VANTAGENS / DESVANTAGENS DA NORMALIZAÇÃO

Vantagens

- Estruturas de dados mais estáveis
- Elimina a redundância
- Obtêm-se um modelo de dados mais natural e mais simples
- Evitam-se os efeitos laterais da alteração
- Evitam-se os efeitos laterais da inserção
- Evitam-se os efeitos laterais da remoção
- Facilita a exploração e manutenção de ficheiros

Desvantagens

- Favorece a proliferação no n.º de tabelas
- Favorece a fragmentação exagerada
- Perigoso de seguir cegamente

Conclusão

A normalização de dados é uma série de passos que se seguem no projecto de uma base de dados, que permitem um armazenamento consistente e um eficiente acesso aos dados em base de dados relacionais. Esses passos reduzem a redundância de dados e as chances dos dados se tornarem inconsistentes.

No entanto, muitos SGBDs relacionais não têm separação suficiente entre o projecto lógico da base de dados e a implementação física do banco de dados, e isso tem como consequência que as consultas feitas a um banco de dados totalmente normalizado tenham mau desempenho. Nestes casos, usa-se por vezes a desnormalização para melhorar o desempenho, com o custo de menores garantias de consistência.

Avaliação da actividade

Verifique a sua compreensão!

1. Como definiria “normalização de dados”?
2. Comente sobre a 3FN.
3. Em que ocasião podemos desnormalizar os dados?

Actividades de Aprendizagem

Actividade 3.3 – Boyce Codd até 5 Forma Normal

Introdução

Em situações muito particulares, a simples aplicação da 3FN pode levar à existência de anomalias significativas. Um refinamento da 3FN foi proposto e é vulgarmente denominado como sendo a Forma Normal de Boyce Codd (FNBC).

Leia o texto que se segue:

Forma Normal Boyce Codd

A razão deste estranho nome deve-se ao facto de se ter determinado que eram cinco as formas normais necessárias. (1FN, 2FN, 3FN, 4FN e 5FN).

No entanto, alguns anos mais tarde, detectou-se um caso especial da 3FN.

Ora, como não fazia nenhum sentido que algo parecido com a 3FN se chamasse 6FN, optou-se por lhe dar o nome dos seus criadores, Edgar F. Codd e Ray Boyce.

Enquanto a 3FN lida com dependência entre atributos chave e não chave, a FNBC lida com dependência entre chaves.

Esta forma normal foi especificada em especial para lidar com situações em que:

Existem múltiplas chaves candidatas;

As chaves candidatas são compostas;

As chaves candidatas sobrepõem-se.

Se algumas das situações acima não se verificar, então a BCNF reduz-se a 3FN.

Vejamos o seguinte exemplo que se assume as seguintes dependências funcionais:

Encarregado Cidade, Departamento

Cada empregado trabalha numa única cidade e gere um único departamento;

Cidade, Departamento Encarregado

Só existe um encarregado para cada cidade e departamento;

Projecto, Cidade Departamento

Podem existir vários encarregados para um mesmo projecto, mas têm de existir em cidades distintas;

Cada encarregado pode ser responsável por mais do que um projecto;

Em cada cidade um projecto está associado a um único departamento e só tem um encarregado responsável por ele.

Encarregado	Projecto	Cidade	Departamento
Aurélío	Informática	Mocuba	D1
Albano	Electrónica	Quelimane	D1
Albano	Informática	Quelimane	D1
Victorino	Pontes	Quelimane	D2
Victorino	Estradas	Quelimane	D2

Tabela 9 – Entidade Original

Esta situação pode ser resolvida com a criação de outra entidade, resultando no seguinte esquema sem duplicação de valores.

Encarregado	Cidade	Departamento
Aurélío	Mocuba	D1
Albano	Quelimane	D1
Victorino	Quelimane	D2

Tabela 10 – Entidade com os Encarregados

Projecto	Cidade	Departamento
Informática	Mocuba	D1
Electrónica	Quelimane	D1
Informática	Quelimane	D1
Pontes	Quelimane	D2
Estradas	Quelimane	D2

Tabela 11 – Entidade com os Projectos

Para que a FNBC seja aplicável é necessário que existam, pelo menos duas chaves candidatas com atributos comuns que se sobreponham.

Quarta Forma Normal

Uma relação está na Quarta Forma Normal (4FN) se não possuir dois ou mais atributos multi-valorados independentes. Em outras palavras, se não possuir casos de multi-dependência funcional. Mas o que é mesmo multi-dependência funcional? Se cada valor de um Atributo A permite descobrir um conjunto de possíveis valores de um outro Atributo B, dizemos que A determina multi-funcionalmente B ($A \twoheadrightarrow B$). Por exemplo, geralmente, em uma universidade um professor ministra mais de uma disciplina. Logo, Nome_Prof \twoheadrightarrow Disciplina (Nome_Prof determina multi-funcionalmente disciplina) e, assim, sempre que o nome do professor for pesquisado será possível obter os nomes de suas disciplinas.

Por exemplo, analise a relação Alocação_Professor (vide Tabela 12). Nela temos as seguintes multi-dependências funcionais: Nome_Prof \twoheadrightarrow Sigla_Disciplina (ou seja, a partir do nome do professor podemos saber as disciplina que ele ministra) e Nome_Prof \twoheadrightarrow Orientando (ou seja, a partir do nome do professor é possível saber os alunos que ele orienta).

Nome_Prof (PK)	Sigla_disciplina (PK)	Orientado (PK)
Aurélio Ribeiro	BD	Eugénio Macumbe
Aurélio Ribeiro	AS	Eugénio Macumbe
Valdinácio Paulo	REDES	Ambrósio Vumu
Valdinácio Paulo	REDES	Ambrósio Vumu
Martina Barros	SO	Gilberto Luis

Tabela 12 – Relação Alocação_Professor (fora da 4FN)

Para se verificar a 4FN a relação deve ter, no mínimo, três atributos.

Observe que as disciplinas que o professor lecciona devem ser independentes dos alunos que ele orienta. Se a relação acima significasse que o professor orienta determinado aluno em uma determinada disciplina, a relação estaria correcta e não necessitaria ser normalizada. Porém, estamos considerando multi-dependências funcionais entre atributos independentes. Relações que não estão na 4FN podem apresentar problemas de inconsistências devido à duplicidade dos dados. Por exemplo, na Tabela 12, o professor “Aurélio Ribeiro” só tem um orientando “Eugénio Macumbe”, mas ministra duas disciplinas. Devido a esse fato, o nome do orientando precisou ser duplicado sem necessidade. O mesmo ocorreu com o professor “Valdinácio Paulo” que só ministra uma disciplina “REDES” e possui dois orientandos (houve duplicação da sigla da disciplina). Adicionalmente, relações desse tipo podem apresentar anomalias de actualização, por exemplo:

- Anomalia de inserção – um orientando só pode ser inserido se o professor estiver ministrando alguma disciplina (veja que todos os atributos são chaves, nenhum deles pode ficar com valor null);
- Anomalia de remoção – apagar uma disciplina ou um orientando levaria à perda de informação;
- Anomalia de alteração – se necessitássemos alterar o nome de um orientando, precisaríamos alterar o mesmo em todas as tuplas em que ele aparecesse.

Para Normalizar para a Quarta Forma Normal, você deve retirar da relação o atributo multi-valorado e criar uma nova relação, tendo-o como parte da chave, a fim de separar as dependências. Ou seja, deve-se usar o mesmo procedimento feito para passar uma relação para a 1FN, visto anteriormente. Uma observação importante é que, para se normalizar em 4FN, a entidade tem que estar (obrigatoriamente) na 3FN. Por exemplo, a relação Alocação_Professor daria origem à nova relação Orientações_Professor (vide Tabela 13) e provocaria mudanças na relação original Alocação_Professor (vide Tabela 14).

Nome_Prof (PK)	Orientado (PK)
Aurélio Ribeiro	Eugénio Macumbe
Aurélio Ribeiro	Eugénio Macumbe
Valdinácio Paulo	Ambrósio Vumu
Valdinácio Paulo	Ambrósio Vumu
Martina Barros	Gilberto Luis

Tabela 13 - Relação Orientações_Professor

Nome_Prof (PK)	Sigla_disciplina (PK)
Aurélio Ribeiro	BD
Aurélio Ribeiro	AS
Valdinácio Paulo	REDES
Valdinácio Paulo	REDES
Martina Barros	SO

Tabela 14- Relação Alocação_Professor (na 4FN)

Veja que, agora, assuntos diferentes são tratados em relações diferentes, evitando duplicações e anomalias.

É importante lembrar que a necessidade de aplicar a 4FN ocorre apenas quando se tem tabelas que mantêm relacionamentos ternários ou superiores e se detectam dependências funcionais multivaloradas independentes.

Quinta Forma Normal

Não vamos dar uma exposição abrangente sobre a quinta forma normal, vamos apenas apresentar uma noção da sua ideia central. A quinta forma normal (5FN) é também chamada de Forma Normal de Projecção / Junção (FNPJ) e trata de casos bastante particulares que ocorrem na modelagem de dados: os relacionamentos múltiplos (ternários, quaternários e n-ários).

Uma relação R está na quinta forma normal (5FN) se está na 4FN e não possui nenhuma dependência de junção. Mas o que é uma dependência de junção?

Dependência de Junção: Dada uma relação R e certas projeções (subconjuntos da relação) R_1, \dots, R_n diz-se que há uma dependência de junção (de R em relação a R_1, \dots, R_n) se R pode ser reconstituída

com a junção de R_1, \dots, R_n .

Em outras palavras, uma relação na 4FN estará na 5FN, quando seu conteúdo não puder ser reconstruído (junção), porque há perda de informação, a partir das diversas relações menores (subconjuntos extraídos da relação principal). Ou seja, se ao particionar uma relação, sua junção posterior não conseguir recuperar as informações contidas na relação original, então esta relação está na 5FN.

Assim, se uma relação não está na 5ªFN, então, existe uma decomposição de R em um conjunto de projeções (subconjuntos) que estão na 5ªFN e cuja junção natural restabelece a relação original. Esta forma normal trata especificamente dos casos de perda de informação, quando da decomposição de relacionamentos múltiplos.

Para exemplificar, pode acontecer de uma tabela representar um relacionamento triplo que não pode ser simplificado, pois se for quebrado em relacionamentos binários poderá gerar informações incorrectas. Neste caso, a tabela já está na 5FN. Para verificar esse fato, pode-se utilizar a seguinte pergunta: é possível substituir o relacionamento ternário por relacionamentos binários?

Vamos a outro exemplo. Se um vendedor vende certo tipo de veículo e ele representa o fabricante daquele tipo de veículo, então ele vende aquele tipo de veículo para aquele fabricante (regra de simetria), tal como representado na relação Vendas (vide Tabela 15). Essa relação pode ser decomposta em três outras relações, portanto não está na 5NF (observe que há uma redundância de informações na relação, que irá requerer mais atenção na actualização de dados).

Vendedor (PK)	Fabricante (PK)	Veiculo(PK)
Aurélio Ribeiro	Ford	Automóvel
Aurélio Ribeiro	Ford	Camião
Aurélio Ribeiro	BMW	Automóvel
Aurélio Ribeiro	BMW	Camião
Valdinácio Paulo	Ford	Automóvel

Tabela 15- Relação Vendas

Assim, para transformar em 5FN podemos quebrar a relação Vendas em 3

relações diferentes, representadas nas Tabelas 16, 17 e 18. Essas relações não podem ser decompostas, estando assim na 5NF. Para conseguir recompor a relação Vendas originais seriam necessárias as três relações.

Vendedor (PK)	Fabricante (PK)
Aurélio Ribeiro	Ford
Aurélio Ribeiro	BMW
Valdinácio Paulo	Ford

Tabela 16 - Relação Vendedor-Fabricante

Vendedor (PK)	Veiculo(PK)
Aurélio Ribeiro	Automóvel
Aurélio Ribeiro	Camião
Valdinácio Paulo	Automóvel

Tabela 17 - Relação Vendedor-Veículo

Fabricante (PK)	Veiculo(PK)
Ford	Automóvel
Ford	Camião
BMW	Automóvel
BMW	Camião

Tabela 18 - Relação Fabricante-Veículo

Qualquer outra decomposição da relação original poderia ocasionar informações incorrectas (tuplas espúrias). Como pôde ser observado, a 4NF decompõe uma relação aos pares (substitui uma relação por duas de suas projecções – em pares), enquanto que a 5NF é utilizada quando uma decomposição aos pares não é possível, como no caso anterior (a relação foi decomposta em três outras, sem perdas).

Conclusão

A partir do modelo relacional gerado a partir do modelo entidade relacionamento, construído na fase de modelagem conceitual, proceda os seguintes passos:

Elimine os atributos repetidos (se houver), de modo a obter um modelo na 1FN;

Elimine as dependências parciais da chave primária em suas relações (se houver) para obter o modelo na 2FN;

Elimine as dependências transitivas nas tabelas (se houverem), obtendo um esquema na 3FN;

Avalie se vale a pena aplicar mais algum tipo adicional de normalização 4FN, 5FN e FNBC.

Avaliação da actividade

Verifique a sua compreensão!

1. Qual é a relação entre a 3FN e FNBC?
2. Defina “dependência de junção”?
3. Quais são os pressupostos da 5FN?

Resumo da Unidade

O conceito de normalização foi introduzido por Codd e, inicialmente ele criou as três primeiras formas de normalização chamando-as de: primeira forma normal (1NF), segunda forma normal (2NF) e terceira forma normal (3NF). Uma definição mais forte da 3NF foi proposta depois por Boyce-Codd, e é conhecida como forma normal de Boyce-Codd (FNBC).

Através do processo de normalização pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta “purificado” em relação às anomalias de actualização (inclusão, alteração e exclusão) as quais podem causar certos problemas, tais como: grupos repetitivos (atributos multivalorados) de dados; variação temporal de certos atributos, dependências funcionais totais ou parciais em relação a uma chave concatenada; redundâncias de dados desnecessárias; perdas acidentais de informação; dificuldade na representação de factos da realidade observada; dependências transitivas entre atributos, entre outros aspectos.

O processo de Normalização pode ser considerado como a aplicação de uma série de regras, que constituem as Formas Normais, que irão provocar a decomposição de esquemas de dados insatisfatórios de algumas relações, em novas relações.

O processo de normalização é aplicado em uma relação por vez. Durante o processo, a relação vai sendo “quebrada”, criando-se outras relações. Geralmente, o processo de normalização é realizado em etapas, começando através das formas normais menos rígidas e depois através de refinamentos sucessivos, até chegar a uma normalização considerada satisfatória para a aplicação.

A decisão de normalizar ou não uma relação é um compromisso entre garantir a eliminação de inconsistências no BD e alcançar a eficiência de acesso (pois normalizar demais pode diminuir a eficiência dos aplicativos). Por isso mesmo, deve ser realizada alguma ponderação antes de se tomar uma decisão.

Avaliação da Unidade

Verifique a sua compreensão!

1. Normalize as estruturas seguintes, apresentando a 1ª, 2ª e 3ª formas normais.

a) Factura (NrFactura, CodCliente, NomeCliente, MoradaCliente, CodProduto, DescriçãoProduto, Preço, Quantidade) .

b) Fornecedor (Cód-Fornecedor, Nome, N°-Telefone, Endereço, Cód-Postal, Localidade, Cód-Produto, Desc-Produto, Quantidade, Preço-Unitário).

Resolução:

Tabela Factura

1FN

Factura (#NrFactura, CodCliente, NomeCliente, MoradaCliente)

Produto (#NrFactura, #CodProduto, DescriçãoProduto, Preço, Quantidade)

2FN

Factura (#NrFactura, CodCliente, NomeCliente, MoradaCliente)

Produto (#CodProduto, DescriçãoProduto, Preço)

Produto/Factura (#NrFactura, #CodProduto, Quantidade)

3FN

Factura (#NrFactura, CodCliente)

Produto (#CodProduto, DescriçãoProduto, Preço)

Produto/Factura (#NrFactura, #CodProduto, Quantidade)

Cliente (#CodCliente, NomeCliente, MoradaCliente)

Tabela Fornecedor

1FN

Fornecedor (#Cód-Fornecedor, Nome, N°-Telefone, Endereço, Cód-Postal, Localidade)

Produto (#Cód-Fornecedor, #Cód-Produto, Desc-Produto, Quantidade, Preço-Unitário)

2FN

Fornecedor (#Cód-Fornecedor, Nome, N°-Telefone, Endereço, Cód-Postal, Localidade)

Produto (#Cód-Produto, Desc-Produto, Preço-Unitário)

Fornecedor/Produto (#Cód-Fornecedor, #Cód-Produto, Quantidade)

3FN

Fornecedor (#Cód-Fornecedor, Nome, N°-Telefone, Endereço, Cód-Postal)

CPostal (#Cód-Postal, Localidade)

Produto (#Cód-Produto, Desc-Produto, Preço-Unitário)

Fornecedor/Produto (#Cód-Fornecedor, #Cód-Produto, Quantidade)

Leituras e outros Recursos

- <https://fenix.tecnico.ulisboa.pt/downloadFile/3779571244058/1normform.pdf>
- https://pt.wikipedia.org/wiki/Normaliza%C3%A7%C3%A3o_de_dados
- Tecnologia de Bases de Dados (3ª Edição), José Luis Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431
- Banco de Dados- Vol III, Sandra de Albuquerque SIEBRA, Recife, 2010

Unidade 4. Introdução à Álgebra Relacional e SQL

Introdução à Unidade

Álgebra relacional é uma linguagem teórica com operadores que trabalham com uma ou mais relações para definir outra relação sem alterar a relação original. Ela é conhecida como linguagem de relação em tempo real, isso significa que todas as tuplas que possivelmente venham de diferentes relações, são manipuladas em uma declaração. Nesta unidade vamos apresentar as principais operações em álgebra relacional, que servirão de ponte para escrever códigos em SQL.

Objectivos da Unidade

Após a conclusão desta unidade, deverás ser capaz de:

- Listar e descrever as operações básicas de álgebra relacional;
- Criar novas relações derivadas das operações básicas de álgebra relacional;
- Diferenciar os comandos DLL, DML e DCL;
- Aplicar a DLL, DM e DCL na linha de comando SQL.
- Criar um projecto de base de dados usando SQL.

Termos-chave

Álgebra relacional, SQL

A Álgebra Relacional é uma linguagem de consulta formal, porém procedimental, ou seja, o usuário dá as instruções ao sistema para que o mesmo realize uma sequência de operações na base de dados para calcular o resultado desejado. [Fonte: http://www.macoratti.net/13/06/sql_arcb.htm, acessado em 30 de Maio 2017]

SQL ou Linguagem de Consulta Estruturada é uma linguagem de consulta declarativa, não-procedural, fundamentada na álgebra e no cálculo relacional de tupla.

Actividades de Aprendizagem

Actividade 4.1 – Operações Básicas em Álgebra relacional

Introdução

Existem oito operações básicas: selecção, projecção, divisão, produto cartesiano, união, intersecção, diferença e junção. Os operadores têm como argumentos relações de entrada e devolvem uma relação como resultado.

Seleção (ou Restrição)

Indicada por σ (a letra grega sigma), é uma operação que para um conjunto inicial fornecido como argumento, produz um subconjunto estruturalmente idêntico, mas apenas com os elementos do conjunto original que atendem a uma determinada condição (também chamada de predicado). A selecção pode ser entendida como uma operação que filtra as linhas de uma tabela, e é também uma operação unária, já que opera sobre um único conjunto de dados.

Seleção ou Restrição: A operação de selecção trabalha em uma relação simples R e define a relação que contém apenas as *tuplas* (linhas) de R que satisfaz a condição (predicado).

- Predicado mais complexos podem ser constituídos por:

\wedge (AND), \vee (OR), e \sim (NOT).

- São utilizadas os operadores: =, \neq , \leq , \geq , $>$ e $<$.

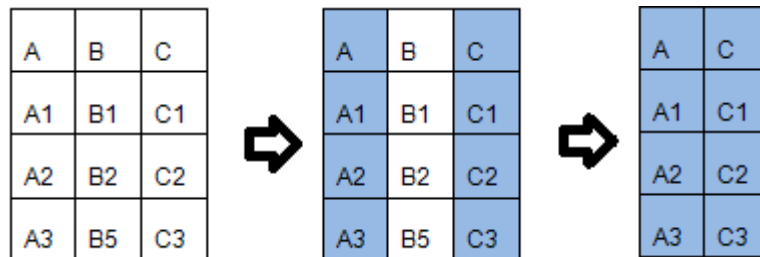
Projecção

Geralmente indicada na literatura por π (a letra grega pi) produz um conjunto onde há um elemento para cada elemento do conjunto de entrada, sendo que a estrutura dos membros do conjunto resultante é definida nos argumentos da operação. Pode ser entendida como uma operação que filtra as colunas de uma tabela. Por operar sobre apenas um conjunto de entrada, a projecção é classificada como uma operação unária.

A operação de **projecção** trabalha em uma relação simples R e define a relação que contém um subconjunto vertical de R, extraindo valores de atributos especificados e eliminando valores duplicados.

O resultado é a relação com *n-colunas* obtidas eliminando as colunas que não estão na lista. **Os tuplos duplicados são removidos.**

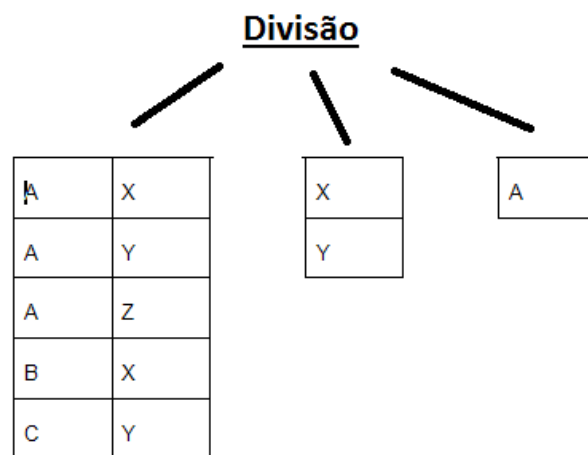
Exemplo: $\pi_{A,C}(R)$



Divisão: R/S

É uma operação adicional que produz como resultado a projecção de todos os elementos da primeira tabela que se relacionam com todos os elementos da segunda tabela.

Exemplo:

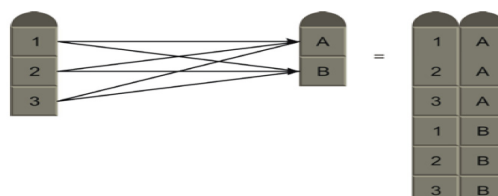


Produto cartesiano

A notação geralmente adoptada (na forma 'conjunto1 x conjunto2') para representar essa operação binária indica bem a sua natureza: o resultado do produto cartesiano de duas tabelas é uma terceira tabela contendo todas as combinações possíveis entre os elementos das tabelas originais.

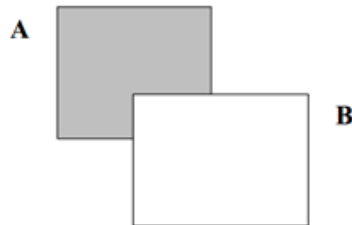
Produto Cartesiano: Define a concatenação de todas as tuplas da relação **R** com todas as tuplas da relação **S**. i.e. multiplica duas relações para definir outra consistindo em todos os pares de tuplas possíveis dentre duas relações.

Exemplo:



Diferença entre conjuntos: $A - B$

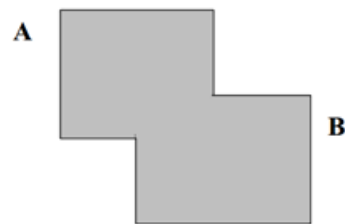
É uma operação primitiva que requer como operandos duas tabelas união-compatíveis, ou seja, estruturalmente idênticas. O resultado é uma tabela que possui todas as linhas que existem na primeira tabela e não existem na segunda.



Observe que $A - B$ é diferente de $B - A$.

União: $A \cup B$

Esta operação primitiva também requer como operandos tabelas união-compatíveis. Produz como resultado uma tabela que contém todas as linhas da primeira tabela seguidas de todas as linhas da segunda tabela. A tabela resultante possui a mesma quantidade de colunas que as tabelas originais, e tem um número de linhas que é no máximo igual à soma das linhas das tabelas fornecidas como operandos, já que as linhas que são comuns a ambas as tabelas aparecem uma única vez no resultado.



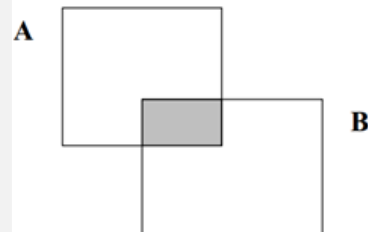
Observe que $A \cup B = B \cup A$.

Intersecção: $A \cap B$

Esta é uma operação adicional que produz como resultado uma tabela que contém, sem repetições, todos os elementos que são comuns às duas tabelas fornecidas como operandos.

As tabelas devem ser união-compatíveis.

O mesmo efeito pode ser obtido fazendo-se uma combinação de diferenças entre conjuntos
 $A \cap B = A - (A - B)$
 ou com uniões e diferenças
 $A \cap B = (A \cup B) - (A - B) - (B - A)$ ou ainda
 $A \cap B = (A \cup B) - ((A - B) \cup (B - A))$



Junção: $A \bowtie B$

É uma operação que produz uma combinação entre as linhas de uma tabela com as linhas correspondentes de outra tabela, sendo em princípio correspondente a uma selecção pelos atributos de relacionamento sobre um produto cartesiano dessas tabelas:

$$A \bowtie B = \sigma_{A.chave1 = B.chave2} (A \times B)$$

A operação de junção foi criada justamente porque esse tipo de combinação de tabelas é de uso muito comum, facilitando com isso a escrita de expressões. A tabela resultante de uma junção tem todas as colunas da primeira tabela e todas da segunda tabela. Isso faz com que os valores dos campos utilizados como critério para a correspondência entre as linhas apareça duplicado, já que um vem da primeira tabela e outro da segunda. Existe uma variação da junção, chamada junção natural, que fornece o mesmo resultado, mas sem essa repetição de valores: uma das colunas correspondentes aos atributos de relacionamento é descartada.

Resumo da actividade.

Nesta actividade apresentamos o conceito de Álgebra Relacional, tendo em conta as principais operações, nomeadamente, selecção, projecção, divisão, produto cartesiano, união, intersecção, diferença e junção. Os operadores têm como argumentos relações de entrada e devolvem uma relação como resultado.

Avaliação da actividade

Verifique a sua compreensão!

1. Seja dada a relação (TabelaR). Calcule:

a. $\sigma_{A=B \wedge D>5} (R)$

b. $\pi_{A,C} (R)$

c. Selecciona onde $D=7$

d. Projecta A e B onde $C>1$

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

Actividades de Aprendizagem

Actividade 4.2 –SQL

Introdução

SQL ou Linguagem de Consulta Estruturada é uma linguagem de consulta declarativa, não-procedural, fundamentada na álgebra e no cálculo relacional de tupla. Apesar de ser chamada linguagem de consulta (Query), ela não é apenas de consulta, ela inclui comandos para definição, manutenção e consulta em bancos de dados relacionais. Além disso, ela define

mecanismos para criação de visões, especificações de segurança, autorizações, definições de restrições e controle de transações. Adicionalmente, ela possui regras para embutir os comandos SQL em linguagens de programação genéricas como Java, PHP, C# ou C/C++.

Para realizar esta actividade deve ler o texto que se segue e no fim responder às questões colocadas para avaliar a qualidade da sua aprendizagem:

A SQL padrão é suportada por todos os SGBDs relacionais comerciais. Porém, mesmo padronizada, existem variações, ou seja, cada fornecedor pode incluir comandos próprios na SQL utilizada pelo seu SGBD. Em outras palavras, cada implementação do SQL de cada fornecedor possui os comandos do SQL padrão (também chamado SQL ANSI) e, também, algumas adaptações para resolver certas particularidades. Para conhecer o conjunto completo de comandos SQL de um determinado fornecedor (ex: Oracle), recomendamos a leitura do manual do fabricante. A vantagem de fazer uso apenas do SQL padrão é não ter problemas com migração de SGBD para SGBD. Por exemplo, se você fazia uso de SQL Server e, depois, migrou para o uso do Oracle, se fez uso apenas do SQL padrão, não haverá problemas ou necessidade de adaptações.

Trabalhando com SQL

O SQL é uma linguagem que permite ao utilizador:

Criar BD e suas estruturas relacionais correspondentes;

Administrar uma BD criando, eliminando e alterando os dados;

Realizar consultas simples e complexas.

Também tem dois componentes principais:

Linguagem de definição de dados "Data Definition Language" (DDL) para definir estruturas de bases de dados ;

Linguagem de Manipulação de dados "Data Manipulation Language" (DML) para recuperar e alterar os dados.

Mais adiante veremos esses e outros componentes. Para já vamos falar sobre tipos de dados.

Tipos de Dados

Antes de entrar nos comandos propriamente ditos da SQL, vale a pena comentar sobre tipos de dados. Para definir os atributos das tabelas, precisamos definir os domínios de cada um deles. Isso é feito através da especificação do tipo do dado. Nesse ponto é importante ressaltar que cada SGBD tem um conjunto próprio de tipos de dados. Mas, podemos dizer que, genericamente, vamos encontrar, na maioria dos SGBDs, tipos como:

Char(X): Para dados caracteres, onde X é o tamanho máximo permitido de caracteres e esse tamanho é fixo. Ou seja, se for especificado, por exemplo, um tamanho de 50 caracteres, sempre será ocupado na memória 50 posições, independente da palavra sendo armazenada.

Varchar (X): Idem ao anterior, mas o tamanho armazenado é variável. Só ocupará memória apenas para o que for digitado, tendo o X apenas como referência para tamanho máximo.

Integer : Para dados numéricos inteiros positivos ou negativos.

Decimal (X,Y): Para dados numéricos decimais, onde X é o tamanho máximo permitido da parte inteira e Y é o tamanho máximo da parte fraccionária.

Date: Para datas. Seu formato depende do SGBD relacional. E cada SGBD pode ter um tipo diferenciado para armazenamento de datas.

Logical: Para os valores lógicos TRUE ou FALSE.

Componentes ou subconjuntos do SQL

A linguagem SQL é dividida em subconjuntos de acordo com as operações que queremos efectuar sobre um banco de dados, tais como:

DML - O primeiro grupo é a DML (Data Manipulation Language - Linguagem de manipulação de dados). Este é um subconjunto da linguagem SQL que é utilizado para realizar inclusões, consultas, alterações e exclusões de dados presentes em registos. Estas tarefas podem ser executadas em vários registos de diversas tabelas ao mesmo tempo. Os comandos que realizam respectivamente as funções acima referidas são: Insert, Select, Update e Delete, etc.

Função	Comando SQL	Descrição do comando	Exemplo
Inclusões	INSERT	é usada para inserir um ou mais registos (tuplas/linhas) a uma tabela existente.	INSERT into Pessoa (id, nome, sexo) values (11,'Kiara', 'F');
Seleccção	SELECT	é usado para seleccionar linhas/ tuplas numa tabela ou vista.	SELECT * from Pessoa;
Alterações	UPDATE	para mudar os valores de dados em uma ou mais linhas da tabela existente.	UPDATE Pessoa SET ID=12 WHERE nome='Kiara';
Exclusões	DELETE	permite remover linhas existentes de uma tabela.	DELETE FROM pessoa WHERE id = 12;

Tabela 19 – Comandos DDL

DDL- Este é o segundo grupo (Data Definition Language - Linguagem de Definição de Dados). Uma DDL permite ao utilizador definir tabelas novas e elementos associados. A maioria das BD de SQL comerciais tem extensões proprietárias no DDL.

Os comandos básicos da DDL são: Create, Alter, Drop, etc.

DCL - O terceiro grupo é o DCL (Data Control Language - Linguagem de Controle de Dados). DCL controla os aspectos de autorização de dados e licenças de utilizadores para controlar quem tem acesso para ver ou manipular dados dentro DA BD.

Duas palavras-chaves da DCL:

GRANT - autoriza ao usuário executar operações.

REVOKE - remove ou restringe a capacidade de um usuário de executar operações.

DTL - Linguagem de Transacção de Dados

BEGINWORK (ou START TRANSACTION, dependendo do dialecto SQL) pode ser usado para marcar o começo de uma transacção de banco de dados que pode ser completada ou não.

COMMIT finaliza uma transacção dentro de um sistema de gestão de base de dados.

ROLLBACK faz com que as mudanças nos dados existentes desde o último COMMIT ou ROLLBACK sejam descartadas.

COMMIT e ROLLBACK interagem com áreas de controle como transacção e locação. Ambos terminam qualquer transacção aberta e liberam qualquer cadeado ligado a dados. Na ausência de um BEGIN WORK ou uma declaração semelhante, a semântica de SQL é dependente da implementação.

Palavras-chave em SQL

Cláusulas

As cláusulas são condições de modificação utilizadas para definir os dados que deseja seleccionar ou modificar em uma consulta.

FROM – Utilizada para especificar a tabela que se vai seleccionar os registos.

WHERE – Utilizada para especificar as condições que devem reunir os registos que serão seleccionados.

GROUP BY – Utilizada para separar os registos seleccionados em grupos específicos.

HAVING – Utilizada para expressar a condição que deve satisfazer cada grupo.

ORDERBY – Utilizada para ordenar os registos seleccionados com uma ordem específica (ascendente ou descendente).

DISTINCT – Utilizada para seleccionar dados sem repetição.

UNION - combina os resultados de duas consultas SQL em uma única tabela para todas as linhas correspondentes.

Operadores Lógicos

AND– “E lógico”. Avalia as condições e devolve um valor verdadeiro caso ambos sejam correctos.

OR– “OU lógico”. Avalia as condições e devolve um valor verdadeiro se algum for correto.

NOT – “Negação lógica”. Devolve o valor contrário da expressão.

Operadores relacionais

O SQL possui operadores relacionais, que são usados para realizar comparações entre valores, em estruturas de controle.

Operador	Descrição
<	Menor
>	Maior
<=	Menor ou igual
>=	Maior ou igual
=	Igual
<>	Diferente

Tabela 20 – Operadores básicos

BETWEEN – Utilizado para especificar um intervalo de valores.

LIKE – Utilizado na comparação de um modelo e para especificar registos de um banco de dados. “Like” + extensão % significa buscar todos resultados com o mesmo início da extensão.

IN - Utilizado para verificar se o valor procurado está dentro de uma lista. Ex.: valor IN (1,2,3,4).

Funções de Agregação

As funções de agregação, como os exemplos abaixo, são usadas dentro de uma cláusula SELECT em grupos de registos para devolver um único valor que se aplica a um grupo de registos.

AVG – Utilizada para calcular a média dos valores de um campo determinado.

COUNT – Utilizada para devolver o número de registos da selecção.

SUM – Utilizada para devolver a soma de todos os valores de um campo determinado.

MAX – Utilizada para devolver o valor mais alto de um campo especificado.

MIN – Utilizada para devolver o valor mais baixo de um campo especificado.

Comando Create

Este comando permite a criação de tabelas na base de dados ou mesmo de sua criação.

Sintaxe:

```
CREATE DATABASE <nome_db>;
```

onde:

nome_db - indica o nome d da Base de Dados a ser criado.

Sintaxe:

```
CREATE TABLE < nome_tabela >
( nome_atributo1 < tipo > [ NOT NULL ],
nome_atributo2 < tipo > [ NOT NULL ],
.....
nome_atributoN < tipo > [ NOT NULL ] ) ;onde:
nome_table - indica o nome da tabela a ser criada.
nome_atributo - indica o nome do campo a ser criado na tabela.
tipo - indica a definição do tipo de atributo (
integer(n), char(n), text, real(n,m), date..., etc ).
n- número de dígitos ou de caracteres
m- número de casas decimais
```

Agora, podemos partir para a definição da tabela Livro que faz uso das duas tabelas anteriormente definidas:

O comando CREATE TABLE especifica uma nova tabela (relação), dando o seu nome e especificando as colunas (atributos), cada uma com seu nome, tipo e restrições iniciais.

Onde : () Indica parte da sintaxe do comando e [] Indica opcionalidade do comando. Vamos explicar agora cada parte do comando completo.

Nome_Atributo- nome do atributo que está sendo definido.

Tipo: domínio do atributo ou seja o tipo do dado do atributo.

Tamanho: alguns tipos de dados necessitam de especificação do tamanho do dado. Por exemplo, o tipo CHAR

NOT NULL: expressa que o atributo não pode receber valores nulos.

DEFAULTvalor: indica um valor a ser atribuído ao atributo caso não seja determinado um valor durante a inserção.

PRIMARY KEY (Primária1, Primária2, ...) – serve para especificar a(s) chave(s) primária(s) da tabela.

UNIQUE: indica que o atributo tem valor único na tabela. Qualquer tentativa de se introduzir uma linha na tabela contendo um valor igual ao do atributo será rejeitada. Serve para indicar chaves secundárias (chaves candidatas). Em Candidata1, Candidata2 devem ser especificados os atributos que terão esse valor único na tabela.

FOREIGN KEY (Estrangeira1[, Estrangeira2 [, ...]]) **REFERENCES** TabelaExterna [(AtributoExterno1 [, AtributoExterno2 [, ...]])] – serve para especificar os atributos que são chaves estrangeiras na relação, já relacionando-os às tabelas onde eles são chave primária (Integridade Referencial). Em Estrangeira1, Estrangeira2, ... especificam-se os atributos que são chave estrangeira. Em TabelaExterna se especifica o nome da tabela onde o atributo é chave primária e, por fim, o nome desse atributo nessa TabelaExterna (porque os atributos na relação e na tabela externa original podem ter nomes diferentes). Se os atributos da relação e da tabela externa tiverem o mesmo nome, esses AtributoExterno1, AtributoExterno2, ... não precisam ser especificados.

CHECK (condição) – aqui são especificadas condições que devem ser verificadas na inserção de dados na tabela (validações).

Resumo da actividade

Nesta actividade apresentamos a linguagem SQL, que foi desenvolvida pelo laboratório da IBM, nos anos 70, como parte do sistema System R (o primeiro SGBD relacional). Ela foi, inicialmente, chamada de SEQUEL (Structured English Query Language), mas teve seu nome alterado para SQL por razões Jurídicas. Em 1986, em um esforço conjunto da ANSI (American National Standards Institute) e da ISO (International Standards Organization) criou-se a primeira versão padrão da SQL, a SQL-86 (SQL1), substituída posteriormente pela SQL-92 (SQL2) e depois pela SQL-99 (SQL3).

Avaliação da actividade

Verifique a sua compreensão!

1. O que um utilizador pode fazer com o SQL?
2. Quais são os comandos DDL?
3. Todas as chaves primárias têm dados UNIQUE. Comente a afirmação.

Actividades de Aprendizagem

Actividade 4.3 – Comandos SQL

Introdução

Caros (as) estudantes, nesta actividade vamos abordar alguns comandos SQL, tendo em conta a linguagem de definição de dados (DDL), a linguagem de manipulação de dados (DML) e a linguagem de controlo de dados (DCL).

1. DDL - Data Definition Language (CREATE, ALTER, DROP, etc)

Seja dada a tabela PESSOA.

Id	Nome	Idade	Sexo
11	Kiara	10	F
12	Jacira	24	F
13	Albano	37	M
20	Vitorino	34	M
21	Aline	15	F
22	Rica	11	F

Criação da tabela. (Sintaxe SQL CREATE TABLE)

O comando create pode ser utilizado para criar BD assim como as vistas (view) e etc.

<pre>CREATE TABLE Nome_Tabela (nome_coluna1 tipo_dado (tamanho), nome_coluna2 tipo_dado (tamanho), nome_coluna3 tipo_dado (tamanho),);</pre>	<pre>CREATE TABLE Pessoa (Id int primary key, Nome char(20), Idade int not null check (Idade between 0 and 150), Sexo char not null check (sexo IN ('M','F')));</pre>
--	---

Alterar a estrutura da tabela (Sintaxe SQL de ALTER TABLE)

ALTERTABLE nome_tabela ADD nome_coluna tipo_dado	ALTERTABLE Pessoa ADD Telefone integer
ALTERTABLE nome_tabela DROP nome_coluna	ALTERTABLE Pessoa DROP Sexo;
ALTERTABLE nome_tabela MODIFY nome_coluna tipo_dado	ALTERTABLE Pessoa MODIFY Nome Text;

Apagar Tabela (DROP)

O comando drop apaga toda estrutura da tabela. Ele também é utilizado para apagar a base de dados e vistas.

Droptable nome_tabela	Droptable pessoa;
------------------------------	--------------------------

2.DML-Data Manipulation Language (SELECT, INSERT, UPDATE, DELETE)

a) Sintaxe SELECT

Este comando é utilizado para seleccionar colunas e linhas numa ou várias tabelas.

SELECT nome_coluna, nome_coluna FROM nome_tabela;	SELECT Id, Nome, Idade, Sexo FROM Pessoa;
SELECT * FROM nome_tabela;	SELECT * FROM Pessoa;

Exemplo de SELECT com cláusula WHERE

SELECT nome_coluna, nome_coluna FROM nome_tabela WHERE nome_coluna valor_operador;	SELECT Id, Nome, Idade, Sexo FROM Pessoa WHERE Sexo='F';
---	---

Exemplo de SELECT com cláusula ORDER BY

SELECT nome_coluna, nome_coluna FROM nome_tabela ORDER BY nome_coluna, nome_coluna ASC DESC ;	SELECT Id, Nome, Idade, Sexo FROM Pessoa ORDER BY Idade ASC ;
--	---

Exemplo de SELECT com cláusula GROUP BY

SELECT nome_coluna, função_agregadora(nome_coluna) FROM nome_tabela WHERE nome_coluna valor_operador GROUP BY nome_coluna;	SELECT sexo, COUNT(sexo) as Total FROM Pessoa GROUP BY sexo;
--	---

Exemplo de SELECT com cláusula HAVING

SELECT nome_coluna, função_agregadora(nome_coluna) FROM nome_tabela WHERE nome_coluna valor_operador GROUPBY nome_coluna HAVING função_agregadora(nome_coluna) valor_operador;	SELECT sexo, COUNT(sexo) as Total FROM Pessoa GROUP BY sexo HAVING SEXO='F';
---	---

Sintaxe INSERT

INSERTINTOtable_name VALUES (value1,value2,value3,...);

Ou

INSERTINTOtable_name (column1,column2,column3,...) VALUES (value1,value2,value3,...);

Este comando permite inserir apenas uma linha/tupla da tabela pessoa. Insertinto PESSOA VALUES (11, 'Kia-ra', 10, 'F');	Este comando permite inserir varias linha/ tuplas da tabela pessoa. Insertinto PESSOA VALUES (12, 'Jacira', 24, 'F'), (13,'Albano', 37, 'M'), (20, 'Vitorino', 34, 'M'), (21, 'Aline', 15, 'F'), (22, 'Rica',11,'F');
--	--

Sintaxe UPDATE

UPDATE table_name SET column1=value1,column2=value2,... WHERE some_column=some_value;	UPDATE Pessoa SET idade=idade+1 WHERE sexo='M';
--	--

Sintaxe DELETE

DELETEFROM table_name WHERE some_column=some_value;	DELETEFROM Pessoa WHERE idade>10;
--	--

3.DCL-Data Control Language (GRANT, REVOKE, etc)

Comando GRANT

O comando GRANT permite atribuir um determinado privilégio a um utilizador. Se for especificada a opção WITH GRANT OPTION, o receptor do privilégio fica com a possibilidade de atribuir a terceiros.

Os privilégios que se podem atribuir são:

Privilégios	Descrição
SELECT	Permite Seleccionar todas colunas de uma tabela
SELECT (LISTA)	Permite Seleccionar as colunas especificadas em lista
INSERT	Permite Inserir registos em todas colunas da tabela

INSERT (LISTA)	Permite Inserir valores na lista das colunas de uma tabela
UPDATE	Permite Alterar conteúdos em todas as colunas de uma tabela
UPDATE (LISTA)	Permite Alterar conteúdo da lista de colunas de uma tabela
DELETE	Permite Apagar registos de uma tabela
REFERENCES	Permite Referenciar todas as colunas de uma tabela
REFERENCES (LISTA)	Permite Referenciar a lista das colunas de uma tabela
EXECUTE	Permite a Execução do STORED PROCEDURES
ALL [PRIVILEGES]	Permite atribuir todos os privilégios acima especificados (que façam parte do sistema)

Tabela 21 – Descrição de Comandos SQL

Exemplo:

SQL>CREATE USER Kiara IDENTIFIED BY lello;

Utilizador criado

SQL>GRANT INSERT, UPDATE, DELETE ON Pessoa TO Aurélio;

Privilégio atribuído

SQL>GRANT SELECT ON Pessoa TO Kiara WITH GRANT OPTION;

Privilégio atribuído

Comando REVOKE

O comando REVOKE permite retirar privilégios concedidos através do comando GRANT.

A sua sintaxe é:

REVOKE [GRANT OPTION FOR] privilégio (s) ON Objecto

FROM Utilizado(res) {RESTRICT | CASCADE}

O conjunto de privilégios é igual aos do comando GRANT.

Exemplo:

SQL>REVOKE INSERT, UPDATE, DELETE ON Pessoa FROM Aurélio;

SQL>REVOKE INSERT, ON Pessoa FROM kiara, Jacira, Silvério;

SQL>REVOKE ALL ON Pessoa TO Aline;

Resumo da actividade.

Nesta actividade apresentamos as principais sintaxes SQL de acordo com a linguagem de definição de dados, a linguagem de manipulação de dados e a linguagem de controlo de dados. Procure praticar todos esses comandos usando uma janela de consola SQL.

Avaliação da actividade

Verifique a sua compreensão!

1. Na linha de comando do seu SGBD crie a tabela acima (Pessoa).
 - a) Teste todos os comandos exemplificados nesta actividade.
 - b) Seleccione o nome e a idade de todas pessoas de sexo feminino.
 - c) Aumente dois anos a todos do sexo masculino.
 - d) Introduza uma coluna chamada telefone.

Resumo da Unidade

Nesta última unidade apresentamos os conceitos de álgebra relacional e SQL. A álgebra relacional é uma derivação descendente da lógica de primeira ordem e da álgebra de conjuntos em relação das operações sobre a relação, que auxilia o trabalho ao identificar os componentes de uma tupla por nome (chamado o atributo) ao invés de uma coluna de chaves numéricas, o qual é chamado a relação na terminologia de banco de dados.

A principal aplicação da álgebra relacional é sustentar a fundamentação teórica de base de dados relacional, particularmente linguagem de consulta para tais base de dados, entre os maiores o SQL.

Avaliação da Unidade

Verifique a sua compreensão!

- Em que consiste a Álgebra relacional?
- Como definiria SQL?
- Qual é o propósito de DML?
- Qual é a função do comando GRANT

Soluções

1. Álgebra relacional é uma linguagem teórica com operadores que trabalham com uma ou mais relações para definir outra relação sem alterar a relação original. Ela é conhecida como linguagem de relação em tempo real, isso significa que todas as tuplas possivelmente venham de diferentes relações, são manipuladas em uma declaração

2. SQL ou Linguagem de Consulta Estruturada é uma linguagem de consulta declarativa, não-procedural, fundamentada na álgebra e no cálculo relacional de tupla.
3. DML (Data Manipulation Language - Linguagem de manipulação de dados). Este é um subconjunto da linguagem SQL que é utilizado para realizar inclusões, consultas, alterações e exclusões de dados presentes em registros.. Os comandos que realizam respectivamente as funções acima referidas são; Insert, Select, Update e Delete, etc.
4. O comando GRANT permite atribuir um determinado privilégio a um utilizador. Se for especificada a opção WITH GRANT OPTION, o receptor do privilégio fica com a possibilidade de atribuir a terceiros.

Avaliação do Curso

Avaliação Sumativa

Esta avaliação visa fazer um balanço das aprendizagens e competências adquiridas no final deste módulo, portanto, ela tem um fim de o (a) classificar. Preste atenção ao resolver.

Escolha a opção correcta:

1. Quando é preferível usar um Sistema de Ficheiros?

Sempre

Nunca se deve usar

Quando a Base de Dados e aplicações forem muito simples, bem definidas e não se espera que mude muito.

Quando a base de dados estiver online.

Nenhuma das anteriores

2. Para que serve uma Base de Dados

Para aumentar a rentabilidade

Para aumentar a velocidade do computador

Para gerir grandes conjuntos de informação

Para eliminar redundância

Nenhuma das anteriores

Qual é Comando para criar tabelas numa base de dados?

DELETE.

CREATE

ALTER

DO

Nenhuma das anteriores

Qual deles não é um SGBD

Oracle

Excel

Mysql

Interbase

Postgres

Em base de dados uma relação equivale a:

Campo

Linha

Tabela

Nenhuma das anteriores

6... é a associação de um ou mais atributos que em conjunto identificam cada um dos tuplos (linhas da tabela)

Chave candidata

Superchave

Chave primaria

Chave estrangeira

7. Estabeleça a correspondência

Select		DCL
Create		
Drop		
Revoke		
Insert		DDL
Grant		
Update		
Alter		
Delete		DML

8. Que solução deve ser adoptada no modelo relacional para relacionamentos M:N? 9.

Explique quando uma tabela está em conformidade com cada uma das Formas Normais:

Calcule $R \times S$ [B = C]

R		S	
A	B	C	D
a1	b1	b2	d2
a2	b2	b1	d3

Seja dada a tabela :

PESSOA (Id, Nome, Idade, Salário, Telefone, Cod_Postal, Localidade)

Mostre apenas o nome e o telefone das pessoas que são de Marracuene cujo salário está abaixo de 1000

4 VAL

Álgebra Relacional:	
SQL:	

Quais são as pessoas que não são de Mocuba?

4 VAL

Álgebra Relacional:	
SQL:	

Correção da Avaliação

Solução (C)

Solução (C)

Solução (B)

Solução (B)

Solução (C)

Solução (B)

Solução: DML (SELECT, INSERT, UPDATE, DELETE, ...),

DDL (CREATE, ALTER, DROP, ...),

DCL (GRANT, REVOKE, ...)

Solução: No modelo relacional não é possível efectuar este tipo de relacionamento de forma directa. Neste caso, deve-se construir uma terceira tabela (tabela de associação). Essa tabela deve possuir chave primária composta de dois campos e as chaves estrangeiras provenientes das duas tabelas originais. Concluindo, um relacionamento de muitos-para-muitos deve ser dividido em dois relacionamentos de um-para-muitos com uma terceira tabela.

Solução: -- Uma tabela está na 1FN (Primeira Forma Normal) quando não possui tabelas aninhadas

-- Uma tabela está na 2FN quando, além de estar na Primeira Forma Normal, não contém dependências parciais

-- Uma tabela está na 3FN quando, além de estar na 2FN (Segunda Forma Normal), não contém dependências transitivas.

Solução:

R X S [B = C]			
A	B	C	D
a1	b1	b1	d3
a2	b2	b2	d2

Solução - Alínea (A)

Álgebra Relacional:	$\pi_{Nome, Telefone} (\sigma_{Localidade = 'Marracuene' \wedge salario < 1000} (PESSOA))$
SQL:	SELECT Nome, Telefone FROM PESSOA WHERE Localidade = ' Marracuene' AND salario <1000;

Solução - Alínea (B)

Álgebra Relacional:	$\pi_{Nome}(\sigma_{Localidade \neq 'Marracuene'}(PESSOA))$
SQL:	<pre>SELECT Nome FROM PESSOA WHERE Localidade <> 'Marracuene';</pre>

Sobre o Autor

Sou Aurélio Armando Pires Ribeiro, docente da Universidade Pedagógica, na Escola Superior Técnica, Departamento de Informática, onde lecciono várias disciplinas, particularmente Tecnologias de Base de Dados e Laboratório de Base de dados. Também trabalho no Centro de Educação Aberta e à Distância, da mesma universidade, sendo o responsável pelas tecnologias de informação e comunicação para o a educação à distância, sendo neste momento o chefe do departamento de Tecnologias de Informação e Comunicação.

Fiz os cursos de bacharelato e Licenciatura em Ensino de Informática na Universidade Pedagógica, assim como o mestrado em Informática Educacional, curso administrado pela Universidade Pedagógica em parceria com a Universidade Federal do Rio Grande do Sul do Brasil.

Tenho certa experiencia em educação à distância, pois por além de ser tutor nesta modalidade de ensino, desempenho actividades de edição e maquetização de materiais auto-instrucionais, faço pesquisa em tecnologias para apoio a educação à distância assim como oriento formação a docentes e estudantes em ambientes virtuais de aprendizagem.

Este manual foi desenvolvido de acordo com o currículo da Africano Universidade Virtual (UVA), numa equipe composta de falantes francófonos e lusófonos. Espero que goste do assunto aqui abordado.

Bom estudo!

Aurélio Ribeiro, M MEd

Desenvolvedor e editor de conteúdo em Português

Email: lellopires@gmail.com

Referências do Curso

- Tecnologia de Bases de Dados (3ª Edição), José Luís Pereira, Editora: FCA - Editora Informática, 1998, ISBN: 9789727221431
- Introdução a Sistemas de Banco de Dados, DATE, C. J.. Elsevier Editora, 2004.
- Banco de Dados- Vol I, Sandra de Albuquerque SIEBRA, Recife, 2010
- Banco de Dados- Vol II, Sandra de Albuquerque SIEBRA, Recife, 2010
- Banco de Dados- Vol III, Sandra de Albuquerque SIEBRA, Recife, 2010
- HECTOR GARCIA-MOLINA, JEFFREY D. ULLMAN, JENNIFER D. WIDOM: Database Systems: The Complete Book, , Prentice Hall; 1st edition (October 2, 2001), ISBN: 0130319953
- RAGHU RAMAKRISHNAN, JOHANNES GEHRKE, MCGRAW-HILL: Database Management Systems, 3 edition (August 14, 2002), ISBN: 0072465638.
- C.J. Date, " An introduction to database systems", (3rd ed Narosa publishers, 1985), 1997 (reprint)
- Henry F.korth, Abraham, " database system concepts", McGraw hill Inc., 1997.
- Naveen Prakash, Introduction to database management", TMH, 1993.
- Bobrowski, " client server architecture and introduction to oracle 7", 1996

Sede da Universidade Virtual africana

The African Virtual University
Headquarters

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oeer@avu.org

Escritório Regional da Universidade Virtual Africana em Dakar

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org