Linguagem SQL

18 de abril de 2007

Sumário

Parte I

Sobre essa Apostila

Conteúdo

O conteúdo dessa apostila é fruto da compilação de diversos materiais livres publicados na internet, disponíveis em diversos sites ou originalmente produzido no CDTC em *http://www.cdtc.org.br*.

O formato original deste material bem como sua atualização está disponível dentro da licença *GNU Free Documentation License*, cujo teor integral encontra-se aqui reproduzido na seção de mesmo nome, tendo inclusive uma versão traduzida (não oficial).

A revisão e alteração vem sendo realizada pelo CDTC (*suporte@cdtc.org.br*) desde outubro de 2006. Críticas e sugestões construtivas são bem-vindas a qualquer tempo.

Autores

A autoria deste é de responsabilidade de Natália Cardoso do Rego Barros (natalia@cdtc.org.br) .

O texto original faz parte do projeto Centro de Difusão de Tecnologia e Conhecimento, que vem sendo realizado pelo ITI (Instituto Nacional de Tecnologia da Informação) em conjunto com outros parceiros institucionais, atuando em conjunto com as universidades federais brasileiras que tem produzido e utilizado Software Livre, apoiando inclusive a comunidade Free Software junto a outras entidades no país.

Informações adicionais podem ser obtidas através do email *ouvidoria@cdtc.org.br*, ou da *home page* da entidade, através da URL **http://www.cdtc.org.br**.

Garantias

O material contido nesta apostila é isento de garantias e o seu uso é de inteira responsabilidade do usuário/leitor. Os autores, bem como o ITI e seus parceiros, não se responsabilizam direta ou indiretamente por qualquer prejuízo oriundo da utilização do material aqui contido.

Licença

Parte II

Informações Básicas

Sobre o CDTC

Objetivo Geral

O Projeto CDTC visa a promoção e o desenvolvimento de ações que incentivem a disseminação de soluções que utilizem padrões abertos e não proprietários de tecnologia, em proveito do desenvolvimento social, cultural, político, tecnológico e econômico da sociedade brasileira.

Objetivo Específico

Auxiliar o Governo Federal na implantação do plano nacional de software não-proprietário e de código fonte aberto, identificando e mobilizando grupos de formadores de opinião dentre os servidores públicos e agentes políticos da União Federal, estimulando e incentivando o mercado nacional a adotar novos modelos de negócio da tecnologia da informação e de novos negócios de comunicação com base em software não-proprietário e de código fonte aberto, oferecendo treinamento específico para técnicos, profissionais de suporte e funcionários públicos usuários, criando grupos de funcionários públicos que irão treinar outros funcionários públicos e atuar como incentivadores e defensores de produtos de software não proprietários e código fonte aberto, oferecendo conteúdo técnico on-line para serviços de suporte, ferramentas para desenvolvimento de produtos de software não proprietários e de seu código fonte livre, articulando redes de terceiros (dentro e fora do governo) fornecedoras de educação, pesquisa, desenvolvimento e teste de produtos de software livre.

Guia do aluno

Neste guia, você terá reunidas uma série de informações importantes para que você comece seu curso. São elas:

- Licenças para cópia de material disponível
- Os 10 mandamentos do aluno de Educação a Distância
- Como participar dos foruns e da wikipédia
- Primeiros passos

É muito importante que você entre em contato com TODAS estas informações, seguindo o roteiro acima.

Licença

Os 10 mandamentos do aluno de educação online

- 1. Acesso à Internet: ter endereço eletrônico, um provedor e um equipamento adequado é pré-requisito para a participação nos cursos a distância.
- 2. Habilidade e disposição para operar programas: ter conhecimentos básicos de Informática é necessário para poder executar as tarefas.
- 3. Vontade para aprender colaborativamente: interagir, ser participativo no ensino a distância conta muitos pontos, pois irá colaborar para o processo ensino-aprendizagem pessoal, dos colegas e dos professores.
- 4. Comportamentos compatíveis com a etiqueta: mostrar-se interessado em conhecer seus colegas de turma respeitando-os e fazendo ser respeitado pelo mesmo.
- 5. Organização pessoal: planejar e organizar tudo é fundamental para facilitar a sua revisão e a sua recuperação de materiais.
- 6. Vontade para realizar as atividades no tempo correto: anotar todas as suas obrigações e realizá-las em tempo real.
- 7. Curiosidade e abertura para inovações: aceitar novas idéias e inovar sempre.
- 8. Flexibilidade e adaptação: requisitos necessário à mudança tecnológica, aprendizagens e descobertas.
- 9. Objetividade em sua comunicação: comunicar-se de forma clara, breve e transparente é ponto - chave na comunicação pela Internet.
- 10. Responsabilidade: ser responsável por seu próprio aprendizado. O ambiente virtual não controla a sua dedicação, mas reflete os resultados do seu esforço e da sua colaboração.

Como participar dos fóruns e Wikipédia

Você tem um problema e precisa de ajuda?

Podemos te ajudar de 2 formas:

A primeira é o uso dos fóruns de notícias e de dúvidas gerais que se distinguem pelo uso:

. O fórum de notícias tem por objetivo disponibilizar um meio de acesso rápido a informações que sejam pertinentes ao curso (avisos, notícias). As mensagens postadas nele são enviadas a

todos participantes. Assim, se o monitor ou algum outro participante tiver uma informação que interesse ao grupo, favor postá-la aqui.
Porém, se o que você deseja é resolver alguma dúvida ou discutir algum tópico específico do curso. É recomendado que você faça uso do Forum de dúvidas gerais que lhe dá recursos mais efetivos para esta prática.

. O fórum de dúvidas gerais tem por objetivo disponibilizar um meio fácil, rápido e interativo para solucionar suas dúvidas e trocar experiências. As mensagens postadas nele são enviadas a todos participantes do curso. Assim, fica muito mais fácil obter respostas, já que todos podem ajudar.
Se você receber uma mensagem com algum tópico que saiba responder, não se preocupe com a formalização ou a gramática. Responda! E não se esqueça de que antes de abrir um novo tópico é recomendável ver se a sua pergunta já foi feita por outro participante.

A segunda forma se dá pelas Wikis:

. Uma wiki é uma página web que pode ser editada colaborativamente, ou seja, qualquer participante pode inserir, editar, apagar textos. As versões antigas vão sendo arquivadas e podem ser recuperadas a qualquer momento que um dos participantes o desejar. Assim, ela oferece um ótimo suporte a processos de aprendizagem colaborativa. A maior wiki na web é o site "Wikipédia", uma experiência grandiosa de construção de uma enciclopédia de forma colaborativa, por pessoas de todas as partes do mundo. Acesse-a em português pelos links:

- Página principal da Wiki - http://pt.wikipedia.org/wiki/

Agradecemos antecipadamente a sua colaboração com a aprendizagem do grupo!

Primeiros Passos

Para uma melhor aprendizagem é recomendável que você siga os seguintes passos:

- Ler o Plano de Ensino e entender a que seu curso se dispõe a ensinar;
- Ler a Ambientação do Moodle para aprender a navegar neste ambiente e se utilizar das ferramentas básicas do mesmo;
- Entrar nas lições seguindo a seqüência descrita no Plano de Ensino;
- Qualquer dúvida, reporte ao Fórum de Dúvidas Gerais.

Perfil do Tutor

Segue-se uma descrição do tutor ideal, baseada no feedback de alunos e de tutores.

O tutor ideal é um modelo de excelência: é consistente, justo e profissional nos respectivos valores e atitudes, incentiva mas é honesto, imparcial, amável, positivo, respeitador, aceita as idéias dos estudantes, é paciente, pessoal, tolerante, apreciativo, compreensivo e pronto a ajudar.

A classificação por um tutor desta natureza proporciona o melhor feedback possível, é crucial, e, para a maior parte dos alunos, constitui o ponto central do processo de aprendizagem.' Este tutor ou instrutor:

- fornece explicações claras acerca do que ele espera, e do estilo de classificação que irá utilizar;
- gosta que lhe façam perguntas adicionais;
- identifica as nossas falhas, mas corrige-as amavelmente', diz um estudante, 'e explica porque motivo a classificação foi ou não foi atribuída';
- tece comentários completos e construtivos, mas de forma agradável (em contraste com um reparo de um estudante: 'os comentários deixam-nos com uma sensação de crítica, de ameaça e de nervossismo')
- dá uma ajuda complementar para encorajar um estudante em dificuldade;
- esclarece pontos que não foram entendidos, ou corretamente aprendidos anteriormente;
- ajuda o estudante a alcançar os seus objetivos;
- é flexível quando necessário;
- mostra um interesse genuíno em motivar os alunos (mesmo os principiantes e, por isso, talvez numa fase menos interessante para o tutor);
- escreve todas as correções de forma legível e com um nível de pormenorização adequado;
- acima de tudo, devolve os trabalhos rapidamente;

Parte III

Linguagem SQL

Capítulo 1

O que é a linguagem SQL

A SQL, Structured Query Language, é uma linguagem de pesquisa para banco de dados relacionais. A maioria das características originais da SQL foram inspiradas na álgebra relacional.

Capítulo 2

Plano de ensino

2.1 Objetivo

Qualificar usuários ao uso da linguagem SQL.

2.2 Público Alvo

Qualquer usuário que tenha uma noção básica de banco de dados.

2.3 Pré-requisitos

Os usuários deverão ser, necessariamente, indicados por empresas públicas e ter conhecimento básico acerca de banco de dados.

2.4 Descrição

O curso será realizado na modalidade Educação a Distância e utilizará a Plataforma Moodle como ferramenta de aprendizagem. Ele será dividido em tópicos e cada um deles é composto por um conjunto de atividades (lições, fóruns, glossários, questionários e outros) que deverão ser executadas de acordo com as instruções fornecidas. O material didático está disponível on-line de acordo com as datas pré-estabelecidas em cada tópico. A versão adotada do Audacity é a 1.2.3, caso possua outra versão instalada, podem ocorrer poucas diferenças.

Todo o material está no formato de lições, e estará disponível ao longo do curso. As lições poderão ser acessadas quantas vezes forem necessárias. Aconselhamos a leitura de "Ambientação do Moodle", para que você conheça o produto de Ensino a Distância, evitando dificuldades advindas do "desconhecimento"sobre o mesmo.

Ao final de cada semana do curso será disponibilizada a prova referente ao módulo estudado anteriormente que também conterá perguntas sobre os textos indicados. Utilize o material de cada semana e os exemplos disponibilizados para se preparar para prova.

Os instrutores estarão a sua disposição ao longo de todo curso. Qualquer dúvida deve ser disponibilizada no fórum ou enviada por e-mail. Diariamente os monitores darão respostas e esclarecimentos.

2.5 Metodologia

O curso está dividido da seguinte maneira:

2.6 Cronograma

- Introdução a Linguagem SQL
- Comandos Básicos
- Alguns Comandos Avançados
- Modificação do banco de dados e criação de relações
- Definindo Dados

As lições contém o contéudo principal. Elas poderão ser acessadas quantas vezes forem necessárias, desde que esteja dentro da semana programada. Ao final de uma lição, você receberá uma nota de acordo com o seu desempenho. Responda com atenção às perguntas de cada lição, pois elas serão consideradas na sua nota final. Caso sua nota numa determinada lição for menor do que 6.0, sugerimos que você faça novamente esta lição.
Ao final do curso será disponibilizada a avaliação referente ao curso. Tanto as notas das lições quanto a da avaliação serão consideradas para a nota final. Todos os módulos ficarão visíveis para que possam ser consultados durante a avaliação final.
Aconselhamos a leitura da "Ambientação do Moodle"para que você conheça a plataforma de Ensino a Distância, evitando dificuldades advindas do "desconhecimento"sobre a mesma.
Os instrutores estarão a sua disposição ao longo de todo curso. Qualquer dúvida deverá ser enviada no fórum. Diariamente os monitores darão respostas e esclarecimentos.

2.7 Programa

O curso de linguagem SQL oferecerá o seguinte conteúdo:

- Uso das ferramentas mais comuns.
- Histórico
- Composição da linguagem
- Estruturas básicas
- Conjuntos
- Visões
- Modificação do banco
- Criação de Relações
- Restrições
- Triggers

2.8 Avaliação

Toda a avaliação será feita on-line.
Aspectos a serem considerados na avaliação:

- Iniciativa e autonomia no processo de aprendizagem e de produção de conhecimento;
- Capacidade de pesquisa e abordagem criativa na solução dos problemas apresentados.

Instrumentos de avaliação:

- Participação ativa nas atividades programadas.
- Avaliação ao final do curso.
- O participante fará várias avaliações referente ao conteúdo do curso. Para a aprovação e obtenção do certificado o participante deverá obter nota final maior ou igual a 6.0 de acordo com a fórmula abaixo:
- Nota Final = ((ML x 7) + (AF x 3)) / 10 = Média aritmética das lições
- AF = Avaliações

2.9 Bibliografia

- http://pt.wikipedia.org/wiki/SQL
- http://www.orafaq.com/faqsql.htm
- http://pt.wikipedia.org/wiki/Gatilhos_%28banco_de_dados%29
- http://pt.wikipedia.org/wiki/Banco_de_dados

Capítulo 3

Introdução a Linguagem SQL

3.1 Histórico

A SQL, Structured Query Language, é uma linguagem de pesquisa para banco de dados relacionais. A maioria das características originais da SQL foram inspiradas na álgebra relacional.

SQL é normalmente pronunciado em português como "esse-quê-ele", porém sua pronúcia correta deveria ser "síquel", do inglês "sequel", ou "alguma coisa que segue outra coisa". SQL é uma brincadeira com o nome da primeira linguagem de consulta QUEL.

Originalmente a SQL foi criado pela IBM mas logo outras visões baseadas nele foram criadas por outros produtores. Assim com essa expansão foi criado um padrão para controlá-la pela American National Standards Institute, ANSI em 1986, e o ISO em 1987.

A SQL ganhou várias versões conforme ia sendo revisada. Em 1992, 1999 e 2003 quando foi revista, a SQL foi chamada de SQL-92, SQL3 e SQL: 2003, respectivamente. Na SQL 3 foram inseridas ao contexto da SQL expressões regulares de emparelhamento, queries recursivas e triggers. A SQL:2003 introduziu características relacionadas ao XML.

3.2 Definição de um exemplo

Durante o nosso curso utilizaremos o exemplo de uma locadora. A seguir mostraremos as entidades utilizadas junto com seus atributos.

cliente(nome, endereco, CPF, data_nascimento, código)
associado(RG, nome, cidade, parentesco, codigo_titular)
aluguel(codigo_cliente, nome_ filme, preco, data_aluguel, data_devolucao)
filme(nome,ator_principal, diretor, genero, idade, copia, duracao)

3.3 Considerações Iniciais

Para começar o estudo dos comandos da SQL alguns conceitos devem ser lembrados.

3.3.1 Transação

É uma coleção de operações que desempenham uma determinada função. Como exemplo usemos uma locadora de vídeos. Para alugar um dvd é necessário escolher o filme, verificar a

disponibilidade no estoque e depois dar baixa no sistema. Ou seja, todas essas ações juntas são operações que precisam ser realizadas para que a transação de alugar um dvd seja realizada.

3.3.2 Entidade

É um objeto ou alguém que pode ser representado num banco de dados. No nosso exemplo da locadora, os clientes são entidades. Uma entidade tem várias propriedades, no nosso exemplo, o cliente pode ter um telefone, um código e outros dados pessoais.

3.3.3 Atributos

Um entidade é formada por um conjunto de atributos que são as propriedades que demos como exemplo no item anterior. Cada atributo tem um conjunto de valores possíveis, isso mostra que aquele atributo receberá valores daquele tipo.

3.3.4 Relacionamentos

O relacionamento é uma associação entre entidades, uma relação entre elas. Por exemplo, se tivéssemos, além da entidade clientes, a entidade empréstimos que armazenasse todos os dvds alugados. Existe um relacionamento entre elas que informaria quem alugou determinado dvd e caso necessitasse contatar a pessoa seria possível através dos dados pessoais encontrados nos atributos da entidade cliente.

3.3.5 Chaves

As chaves são a forma de ligação entre as entidades promovendo o relacionamento entre elas. Uma superchave é o conjunto de um ou mais atributos que tomados, juntos, identificam uma entidade. Na nossa locadora, o código do cliente seria uma superchave pois cada cliente teria seu próprio número. Se não existisse um código, o nome da pessoa e o telefone juntos poderiam ser uma superchave pois não se encontraria morando no mesmo local duas pessoas com o mesmo nome. Essa é a idéia de superchave, juntar atributos para conseguir relacionar entidades. Chave candidata é aquela que não possui nenhum subconjunto que possa ser uma superchave. A chave candidata escolhida pelo projetista é chamada de chave primária. A chave primária é o que diferenciará as entidades.

3.3.6 Tupla

O resultado de nossas pesquisas serão chamados de tuplas. Tupla é um conjunto de objetos com as mesmas características.

3.4 Composição da Linguagem SQL

A linguagem SQL tem várias partes:

- Linguagem de Definição de Dados (DDL) - Através da DDL tem-se comandos de tratamento de relações e criação de índices.

- Linguagem Interativa de manipulação de dados (DML) - É uma linguagem de consulta baseada na álgebra relacional e no cálculo relacional de tuplas. Veremos o que é tupla futuramente no curso. Através do DML temos também comandos de inserção, exclusão e modificação de tuplas no banco de dados.
- Incorporação DML - A SQL é usado também integrado com linguagem de programação como Cobol, Pascal e C, por exemplo.
- Definição de visões - Com a SQL DDL é possível definir visões.
- Autorização - A SQL DDL possui comandos que permitem ou não o acesso a relações e visões.
- Integridade - A SQL DDL possui comandos que criam regras que os dados devem obedecer para serem armazenados no banco.
- Controle de transações - Possui comando que especificam como será feita a inicialização e finalização de transações.

Capítulo 4

Comandos Básicos

4.1 Estruturas Básicas - select, from e where

Vamos agora discutir os comandos básicos da SQL. É importante saber nesse momento que as entidades envolvidas no banco de dados não podem ter nomes repetidos, o que é uma exigência do modelo relacional. Outra característica é a possibilidade de valores nulos para valores, se não houver restrição para isso pois na SQL é possível restringir que atributos não poderão receber valores nulos, como chaves primárias, por exemplo.

4.1.1 Select

O select é utilizado para realizar consultas. Vamos supor que queremos encontrar o nome de todos os associados da locadora, definida no nosso exemplo. A estrutura básica dessa consulta seria:

select nome
from associado
where cidade = "Brasília"

No select coloca-se o campo que se quer pesquisar e no from de que tabela que é a pesquisa. A cláusula where pode ser omitida e nela coloca-se alguma restrição para a pesquisa.

Na SQL é permitida a existência de tuplas duplicadas. Quando se quer no resultado de uma pesquisa apenas valores diferentes usa-se a palavra distinct da seguinte forma:

select distinct nome
from associado

Se você quiser deixar explícito de que a duplicação será aceita, pode-se usar all no lugar de distinct.

Quando se deseja mostrar todos os atributos de uma tabela usa-se a seguinte configuração:

select *
from cliente

Esse exemplo mostra todos os atributos de cliente.

Outra característica importante da cláusula select é que pode-se realizar operações nela, como mostrado logo abaixo:

select idade*10
from filme

Essa operação não alterará o atributo na tabela mas apenas mostrará como resultado da pesquisa a idade do filme vezes 10.

4.1.2 From

Na cláusula from, como foi dito anteriormente, coloca-se as tabelas que foram usadas. É importante ressaltar que mais de uma tabela pode ser utilizada. No exemplo a seguir queremos saber quais são os clientes que estão com filmes:

select cliente.nome
from cliente, aluguel
where cliente.codigo = aluguel.codigo_cliente

Observe que informamos de que tabela cada atributo é ao usar, por exemplo, a estrutura cliente.nome. Essa construção só é necessária se os atributos de diferentes tabelas tiverem nomes iguais, caso contrário essa construção se torna opcional. Costuma-se utilizá-la para deixar claro de que tabela é o atributo.

4.1.3 Where

Na cláusula where, coloca-se restrições para a pesquisa. No exemplo anterior fizemos uma pesquisa onde o código do cliente fosse igual ao código do cliente que alugou um determinado filme, assim poderíamos saber o nome de quem aluguel o filme pois na tabela de aluguel não existe o nome do cliente, apenas o código.

Algumas palavras-chave podem ser usadas junto com a cláusula where, elas são: between, and, not e or como nos exemplos a seguir, assim como operadores de comparação como <, >, = e <>.

select nome
from cliente
where codigo **between** 1000 **and** 1005

select aluguel.nome
from cliente, aluguel
where cliente.nome = "Maria Aparecida"
or cliente.nome = "João da Silva"

No primeiro exemplo seleciona-se o nome dos cliente que tem código entre 1000 e 1005. No segundo, o nome do filme que foi alugado pelo cliente com nome de Maria Aparecida ou João da Silva.

Para verificar se um valor é nulo, usa-se por exemplo:

where preco **is** null

4.2 Estruturas Básicas - rename, strings e ordenação

4.2.1 Rename

As vezes é interessante renomear um atributo na hora de apresentá-lo. Isso pode ser necessário quando, por exemplo, dois atributos de tabelas diferentes serão apresentados. Para realizar essa mudança usa-se a palavra as. Essa palavra pode ser utilizada tanto para renomear atributos como para renomear tabelas. Algumas vezes é necessário renomear tabelas para facilitar a escrita.

select aluguel.nome as nome_filme
from aluguel
where nome_filme = "Titanic"

select cliente1.nome, cliente2.nome
from cliente **as** cliente1, cliente **as** cliente2
where cliente1.data_nascimento = "05/05/1955"
and cliente2.data_nascimento = "03/12/1980"

4.2.2 Utilizando Strings

Para comparar strings utiliza-se a palavra like na cláusula where. Para comparar uma substring, utiliza-se o caracter "%"e para comparar qualquer caracter utiliza-se o caracter "_".

select nome
from cliente
where nome **like** "Maria%"

select nome
from cliente
where nome **like** "%ata%"

select nome
from cliente
where nome **like** "Juli_"

select nome
from cliente
where nome **like** "_ _ _ _ _"

No primeiro exemplo, são selecionados todos os cliente tem o nome começando por Maria. No segundo exemplo, todos os cliente que possuem no meio do nome a sequência de caracteres

"ata"no meio do nome, não pode ser nem no começo nem no fim, pois o caracter % informa que existem letras após e antes dos caracteres.

No terceiro exemplo, seleciona-se todos os nomes que começam com Juli e tem apenas um caracter depois. E no quarto exemplo, seleciona-se todos os nomes que possuem 5 caracteres.

É possível também o uso de de not like para procurar por diferenças ao invés de igualdades.

4.2.3 Ordenando o resultado

Para ordenar o resultado da pesquisa, utiliza-se o comando order by, que ordena de modo ascendente. Caso deseje ordenar de modo descendente acrecenta-se o desc.

```
select *
from cliente
order by nome

select *
from cliente
order by nome desc, codigo asc
```

No primeiro exemplo, ordena-se pelo nome de modo ascendente e no segundo descendente e ascendente. É possível ordenar duas colunas como mostrado no segundo exemplo, ordenando na ordem em que forem apresentadas.

4.3 Funções

A SQL permite o uso de funções pré-definidas para realizar operações de soma(sum), mínimo(min), máximo(max), média(avg) e contagem(count).

As funções avg e sum impõem que a entrada da função sejam números, já as outras funções aceitam strings como entrada.

As funções podem ser usadas de diferentes formas. Vamos citar um exemplo onde soma-se o preço de todos os alugueis realizados por um cliente.

```
select nome,sum(preco)
from cliente, aluguel
where codigo = codigo_cliente
group by nome
```

Quando deseja-se aplicar condições a grupos usa-se a cláusula having. Por exemplo, vamos reunir todos os clientes que possuem uma conta maior que 50 reais.

```
select nome,sum(preco)
from cliente, aluguel
where codigo = codigo_cliente
group by nome
having sum(preco) > 50
```

Capítulo 5

Alguns comandos avançados

5.1 Conjuntos

É possível realizar operações de conjuntos através da SQL, mas é necessário que os resultados das pesquisas sejam de tipos equivalentes.

5.1.1 União

(**select** nome
from cliente)
union
(**select** nome
from associado)

Nesse exemplo, têm-se como resultado o nome de todas as pessoas que são cliente ou dependentes (ou os dois). Veja que como numa operação de união de conjuntos, essa operação elimina automaticamente todas as repetições. Para que os resultados repetidos sejam apresentados deve-se acrescentar all após union.

5.1.2 Intersecção

(**select** codigo
from cliente)
intersect
(**select** codigo_titular
from associado)

Nessa operação, são selecionados os códigos de todos os clientes que possuem dependentes. Assim como a união, essa operação elimina repetições e para não eliminar as repetições deve-se usar all assim como na união.

5.1.3 Exceção

(**select** codigo
from cliente)

except
(**select** codigo_titular
from associado)

Nesse exemplo, são selecionados todos os clientes que não possuem dependetes. Assim como as operações anteriores repetições não são apresentadas, para apresentá-las usa-se, novamente, o all.

5.2 Subconsultas Aninhadas

Por meio da SQL podemos aninhar subconsultas, isto é, colocar uma consulta dentro de outra. Vamos mostrar um exemplo. Vamos supor que queremos pesquisar o nome dos clientes que alugaram filme em que o "Tom Cruise"é o ator principal. Primeiro temos que pesquisar todos os filmes que o "Tom Cruise"é o ator principal. Depois vemos quem são as pessoas que alugaram. Para isso vamos usar duas pesquisas.

5.2.1 IN e NOT IN

select cliente.nome
from cliente, aluguel
where cliente.codigo=aluguel.codigo_cliente
and aluguel.nome_filme **in** (**select** nome
from filme
where ator_principal = "Tom Cruise")

Note que para igualar ao resultado de outro pesquisa usa-se a palavra in. Assim com essa pesquisa primeiro seleciona-se o nome de todos os filmes em que o "Tom Cruise"é o ator principal e depois pesquisa-se o nome de todos os clientes que alugaram um dos filmes que foram resultados da primeira pesquisa.

Quando deseja-se excluir um resultado, ou seja, se quiséssemos o nome dos clientes que não alugaram filme do "Tom Cruise"é só usar not in ao invés de in.

5.2.2 SOME E ALL

Podemos também comparar resultados de outros modos. A expressão "maior que algum"é representada pela expressão >some, comparativamente podem ser usadas as expressões <some, <=some, >=s
Um sinônimo de some que caiu no desuso é o any, foi abandonado por causar ambiguidade.

Quando a idéia é "maior que todos"usa-se o >all. Assim como com o some, outras variações podem ser usada como: <all, <=all, >=all, =all e <>all.

Vamos mostrar um exemplo de como o some e o all podem ser aplicados.

select cliente.nome
from cliente, aluguel
where cliente.nome = "Maria Aparecida da Silva"
and cliente.codigo=aluguel.codigo_cliente
and sum(preco)**<=some** (**select sum**(preco)
from aluguel

group by codigo_cliente)

Nesse exemplo, verifica-se se existe algum cliente que tenha a soma de todos os alugueis de dvds maior ou igual que o da Maria Aparecida da Silva.

select cliente.nome
from cliente, aluguel
where cliente.nome = "Maria Aparecida da Silva"
and cliente.codigo=aluguel.codigo_cliente
and sum(preco)**<=all** (**select sum**(preco)
from aluguel
group by codigo_cliente)

Esse exemplo é muito parecido com o anterior só que verifica se todos os clientes possuem a soma de seus alugueis maior ou igual ao da Maria Aparecida.

5.2.3 EXISTS e NOT EXISTS

Quando desejamos verificar se o resultado de uma pesquisa possui algum resultado específico podemos usar a cláusula exists.

Vamos pesquisar se existe algum cliente que possui dependentes. Para isso vamos usar a pesquisa a seguir:

select nome
from cliente
where exists (**select** *
from cliente,associado
where cliente.codigo = associado.codigo_titular)

Caso queira verificar se um determinado resultado não existe usamos o not exists.

5.2.4 UNIQUE

Quando queremos verificar se um resultado é único podemos usar a cláusula unique. No exemplo a seguir, acharemos todos os cliente que possuem apenas um dependente.

select nome
from cliente,associado
where cliente.codigo = associado.codigo_titular
and unique (**select** codigo_titular
from associado)

5.3 Visões

Visões podem ser entendidas como uma maneira alternativa de visualizar os dados de uma tabela. Pode ser considerada uma tabela virtual. As visões são utilizadas quando não é desejável

que o usuário tenha um acesso aos modelo lógico como um todos, quando você a intenção é que ele veja apenas o necessário.

Suponha que na nossa locadora os dependentes só podem ser esposa ou marido, que filhos não são mais aceitos como dependentes. Criaremos uma visão que agrupará as pessoas que se encaixam nessa restrição mas o banco de dados não será alterado. Como citamos anteriormente, é como se uma tabela virtual fosse criada, sem alterar as existentes.

create view dependetes_restritos **as**
(**select** RG, nome, cidade, parentesco, codigo_titular
from associado
where parentesco = "marido"
or parentesco = "esposa")

Criamos uma visão chamada dependentes_restritos onde só selecionou-se os dependentes que eram marido ou esposa do titular.

Após criar uma visão você pode acessá-la como uma tabela qualquer. Por exemplo:

select nome
from dependentes_restritos
where codigo_titular = 012345

Nesse exemplo, ele procurará na visão o nome do dependente do titular com código 012345.

Para destruir a visão que criamos basta executar o seguinte comando:

drop view dependentes_restritos

Capítulo 6

Modificação do banco de dados e criação de relações

6.1 Modificando o banco de dados

Em um banco de dados é permitida não só a consulta mas também a inserção, remoção e atualização dos dados.

6.1.1 Inserindo dados

Para inserir dados no banco você deve especificar qual tabela receberá os dados. Informar o atributo que receberá o dados não é obrigatório mas se não for informado os dados inseridos devem estar na ordem em que os atributos estão no banco.

Vamos analisar os exemplos seguintes:

insert into filme
values("Titanic", "Leonardo DiCaprio", "Fulano de Tal", "romance", 8, 1, 190)

insert into filme (duracao, copia, idade, genero, diretor, ator_principal, nome)
values(190, 1,8, "romance","Fulano de Tal", "Leonardo DiCaprio", "Titanic")

Veja que nos dois exemplos insere-se os mesmo dados mas no segundo exemplo não precisa-se respeitar a ordem dos atributos no banco pois estão sendo informados os atributos que receberão os dados.

Dentro de uma operação de inserção pode ser utilizada uma pesquisa para achar as tuplas que deve ser inseridos dados.

Suponha que todos os dependentes serão transformados em clientes, então serão inseridos na tabela cliente todos os dependentes que se encontram na tabela associados.

insert into cliente
select nome, "Avenida Teste", CPF, NULL, NULL
from associado

6.1.2 Removendo dados

Para remover dados de uma tabela utiliza-se em SQL o delete from. No exemplo a seguir, removeremos todos os filmes cadastrados no banco que tenham duração maior que 120 minutos.

delete from filme

where duracao>120

Como no insert, é possível utilizar uma cláusula select embutida ao delete. Vamos excluir todos os filmes que foram alugados por clientes e que tinham como data de devolução o dia 12/10/2006.

delete from filme
where nome **in** (**select** nome_filme
from aluguel
where data_devolucao = "12/10/2006")

6.1.3 Atualizando dados

Quando deseja-se atualizar os dados de uma tabela usamos o update e o set para informar qual dado e será atualizado e como. Observe o exemplo a seguir:

update aluguel
set data_devolucao = "12/01/2007"
where data_devolucao = "11/01/2007"

Nesse exemplo modificaremos todos os alugueis que deveriam ser devolvidos no dia "11/01/2007"para o dia "12/01/2007".

6.2 Criando relações

Para compor relações, tomam-se duas relações e o resultado é outra relação. Para realizar uma junção selecionamos o tipo de junção, que representa como as tuplas que não tem relação com nenhuma outra tupla aparecerão na relação, e selecionamos também uma condição de junção, selecionando quais tuplas das duas relações tem correspondência e quais tuplas aparecerão no resultado na junção.

Vamos estudar os seguintes tipos de junção: inner join, left outer join, right outer join, full outer join. Como condição de junção temos: natural, on e using.

Nas junções do tipo outer é obrigatório o uso de uma condição de junção, já no inner é opcional. As condições on e using aparecem depois do tipo de junção, enquanto a condição natural aparece antes.

A condição on costuma ser a mais utilizada pois é bem parecida com a cláusula where. Um par de linhas será correspondente se a expressão da condição on for verdadeira.

A cláusula using é bem parecida com a on. Você informa a coluna e o nome das tabelas e ele vai pesquisar a coluna que tem o mesmo nome da que você informou. O resultado sai de forma reduzida, sem a repetição das colunas que são iguais.

Já a cláusula natural é mais resumida que a using. Essa cláusula comparará todas as colunas de nome igual das duas tabelas.

Vamos agora verificar os tipos de junção:

6.2.1 Inner join

Esta junção retorna todos os pares com correspondentes de linhas nas duas tabelas e descartam as linhas sem correspondentes de ambas as tabelas. Vamos utilizar como exemplo as seguintes tabelas:

cliente

nome	CPF	codigo
Ana Maria	002.123.234-45	0001
Joao da Silva	456.789.987-22	0002
Maria Costa	123.456.789-99	0003

aluguel

codigo_cliente	nome_filme
0001	Titanic
0001	Romeu e Julieta
0002	A Família Adams

select *
from cliente inner join aluguel **on**
cliente.codigo = aluguel.codigo_cliente

Ao realizar o inner join anterior teremos como resultado a seguinte relação:

nome	CPF	codigo	codigo_cliente	nome_filme
Ana Maria	002.123.234-45	0001	0001	Titanic
Ana Maria	002.123.234-45	0001	0001	Romeu e Julieta
Joao da Silva	456.789.987-22	0002	0002	A Família Adams

Ele relaciona as duas tabelas pelo código, e o cliente que não tá relacionado nas duas é desconsiderado na relação resultante.

6.2.2 Outer join

É a seleção em que são restritas as linhas que interessam em uma tabela, mas são consideradas todas as linhas de outra tabela. Ou seja, queremos ver quais linhas de uma tabela estão relacionadas com a outra tabela e quais as linhas não estão.

- left outer join - são incluidas todas as linhas da primeira tabela na expressão.
- right outer join - são incluídas todas as linhas da segunda tabela na expressão.
- full outer join - são incluidas todas as linhas de ambas as tabelas, as que satisfazem a expressão e as que não satisfazem.

Usando o right outer join, no exemplo anterior, não haveria mudança no resultado. Usando o left outer join e o full outer join o resultado seria o seguinte:

nome	CPF	codigo	codigo_cliente	nome_filme
Ana Maria	002.123.234-45	0001	0001	Titanic
Ana Maria	002.123.234-45	0001	0001	Romeu e Julieta
Joao da Silva	456.789.987-22	0002	0002	A Família Adams
Maria Costa	123.456.789-99	0003	NULL	NULL

Como pode-se ver observando o resultado, as relações do exemplo anterior continuam a aparecer mas é adicionado o cliente que não tinha nenhum aluguel e por isso não aparecia. Usando o right outer join não teríamos mudança no resultado pois nenhuma tupla da segunda tabela ficou de fora do resultado.

6.2.3 Outro exemplo

Vamos agora mostrar um exemplo usando using e natural para deixar os conceitos bem claros:

Considere as tabelas do exemplo anterior, mas agora a coluna codigo_cliente da tabela aluguel chama-se codigo. Assim usando using e natural teremos resultados.

select *
from cliente **inner join** aluguel **using**
(codigo_cliente)

select *
from cliente **natural inner join** aluguel

No primeiro exemplo mostramos como usar o using, informamos o nome das duas tabelas e o nome da coluna que será pesquisada. Já no segundo, você informa apenas o nome das tabelas e ele procura entre todas as tabelas as de mesmo nome.

Tome muito cuidado ao usar a cláusula natural pois se, por exemplo, na tabela aluguel, a coluna codigo se referisse ao codigo do filme alugado teriamos um grande problema. Seriam duas colunas de nomes iguais mas que se referem a coisas diferentes, uma ao codigo do cliente e a outra ao codigo do filme.

Capítulo 7

Definindo Dados

7.1 Domínio dos dados

Até agora consideramos que os dados foram dados. Agora construiremos nossas tabelas. Conheceremos os comandos responsavel em criar cada tabela, atribuir o tipo dos dados.

Os dados em SQL podem ser dos seguintes tipos:

7.1.1 char(n)

O char é uma cadeia de caracteres de tamanho fixo n que é definido pelo usuário.

7.1.2 varchar(n)

O varchar é uma cadeia de caracteres de tamanho variável sendo que tem como tamanho máximo n que é informado pelo usuário.

7.1.3 date

Dados com esse formato aceitam datas, dia, mês e ano(com quatro diígitos).

7.1.4 time

Dados com esse formato aceitam horário, horas, minutos e segundos.

7.1.5 int

Esse formato aceita números inteiros.

7.1.6 real

Esse formato aceita números reais.

7.1.7 float(n)

Aceita número flutuante com precisão n definida pelo usuário.

7.2 Definição de esquema

Para criar uma tabela utilizamos o comando create table e para definir quem será a chave primária usamos o comando primary key.

Observe o exemplo a seguir onde criaremos tabelas e definiremos a chave primária.

create table cliente (nome **varchar**(20) **not null**,
endereco **varchar**(50),
CPF **char**(11) **not null**,
data_nascimento **char**(10),
primary key(CPF))

create table filme (nome **varchar**(20) **not null**,
ator_principal **varchar**(20),
diretor **varchar**(20),
genero **varchar**(10),
primary key(nome, ator_principal),
check genero in ("Romance","Ação","Drama","Terror","Comédia"))

No primeiro exemplo criamos a tabela cliente, como você pode observar a estrutura do create table é composta do nome da tabela que será criada e dos atributos com seus devidos domínios. Depois de informar os atributos, informamos a chave primária, através do primary key.

Repare que usamos not null na criação do atributo nome, ele é usado quando queremos forçar o usuário a colocar o dado. O atributo CPF, por exemplo, é nossa chave primária, por isso não poderia ficar em branco, por isso também foi colocado um not null do lado.

No segundo exemplo, usamos a cláusula check, ela é usada para determinar uma condição ao atributo. No caso do nosso exemplo, ele define que o gênero do filmes só pode ser Romance, Ação, Drama, Terror ou Comédia.

7.3 Removendo e alterando tabelas

Para remover uma tabela temos duas opções possíveis o drop table e o delete from. A diferença entre os dois é que o primeiro é mais drástico, com ele remove-se a tabela e todos os seus atributos, já no segundo a tabela é mantida mas os atributos são apagados.

Utilizando o delete from atributos poderão ser inseridos futuramente pois a tabela não foi excluída, ela ainda existe.

Observe o exemplo abaixo:

delete from cliente

drop table cliente

Primeiro deletamos todos os atributos da tabela cliente e depois deletamos a tabela. Poderiamos ter usado apenas o drop table mas usamos os dois para mostrar a sintaxe das duas cláusulas.

Para adicionar um novo atributo a uma tabela utilizamos o alter table. As tuplas desse atributo inserido receberão o NULL para seus valores. Observe como utilizar o alter table.

alter table cliente **add** RG **varchar**(10)

Como podemos observar, informamos a tabela que terá um atributo adicionado, o nome do atributo e o domínio.

O alter table pode ser usado também para remover um atributo, ao invés de utilizar o add, usa-se drop, como no exemplo a seguir.

alter table cliente **drop** endereco

Informa-se o nome da tabela e o atributo a ser removido.

7.4 Restrições

Vimos anteriormente que ao criar uma tabela podemos criar restrições utilizando a cláusula check. Existe outro modo de criar uma restrição dando um nome à restrição, isso facilitará na hora de saber qual restrição foi violada.

Para criar essa restrição utilizaremos o creat domain. Observe o exemplo a seguir:

create domain CPF **char**(11)
constraint teste_CPF **check** (value **not null**)

create domain preco **numeric**(4,2)
constraint teste_preco **check** (**value** >=3,00)

No primeiro exemplo, criamos um teste_CPF que avalia se o CPF não é igual a null. Já no segundo verificamos se o preco, que tem 4 casas sendo que duas são depois da virgula, é maior ou igual a 3.

7.5 Triggers

Trigger, no português, gatilho, é uma ação que é realizada automaticamente quando ocorre uma mudança no banco de dados.

Logo abaixo temos uma trigger que é ativada quando acrescenta-se um novo preço de aluguel de dvd na tabela aluguel. Se for informado qualquer valor menor que três a trigger atualizará automaticamente o banco substituindo o valor por 3 que é o valor mínimo de um aluguel.

define trigger preco_menor **on update of** aluguel
(**if new** aluguel.preco < 3
then (**update** aluguel.preco
set aluguel.preco = 3)