Banco de dados

Módulo III – Linguagem SQL

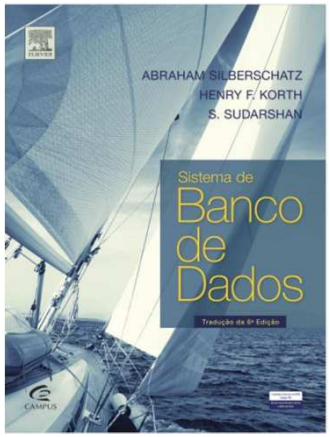
Curso Preparatório - ITnerante

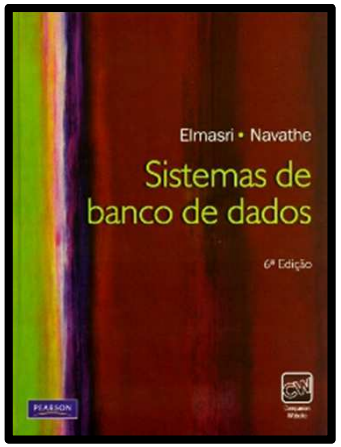
Prof. Thiago Cavalcanti

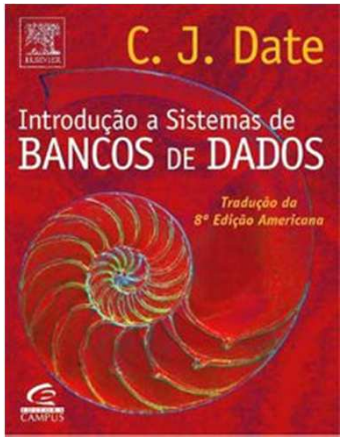
Ementa

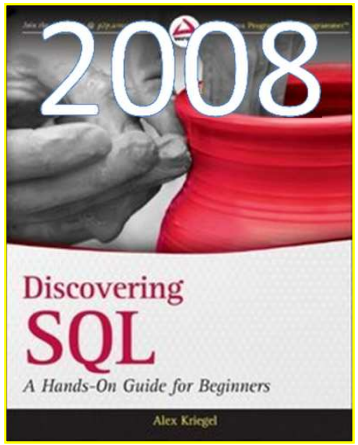
- SQL Básico
- Mais SQL
  - Consultas complexas
  - Triggers
  - Views
  - Modificações de esquema
- Bônus Track
  - Questões

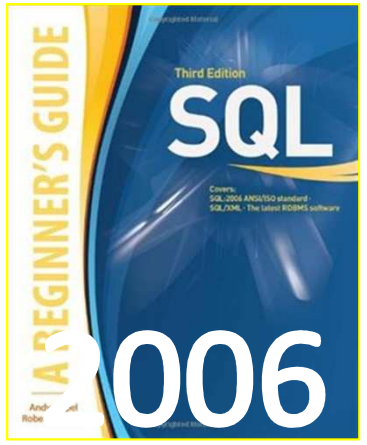
Bibliografia











ITnerante

Objetivos

- Apresentarmos as construções e conceitos fundamentais de SQL.
- Entender como as principais bancas abordam o assunto.

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

ITnerante

TIMASTERS

SQL

20% das questões de BD de concursos

ABSTRACT: In this paper we present the data manipulation facility for a structured English query language (SEQUEL) which can be used for accessing data in an integrated relational data base. Without resorting to the concepts of bound variables and quantifiers SEQUEL identifies a set of simple operations on tabular structures, which can be shown to be of equivalent power to the first order predicate calculus. A SEQUEL user is presented with a consistent set of keyword English templates which reflect how people use tables to obtain information. Moreover, the SEQUEL user is able to compose these basic templates in a structured manner in order to form more complex queries. SEQUEL is intended as a data base sublanguage for both the professional programmer and the more infrequent data base user.

A Linguagem SQL

- Pode ser considerada **uma das maiores razões para o sucesso** dos banco de dados relacionais
- Provou ser popular o suficiente para atrair a atenção do American National Standards Institute (ANSI)
  - Em resposta a proliferação dos dialetos do SQL publicou seu **primeiro padrão SQL em 1986** para trazer maior conformidade entre fornecedores.
  - 8 versões: 1986, 1989, 1992, 1999, 2003, **2006**, **2008**, 2011.

As partes do padrão SQL (ISO/IEC 9075)

Part	Topic	Status
1	SQL/Framework	Completed in 1999, revised in 2003, corrections published in 2007
2	SQL/Foundation	Completed in 1986, revised in 1999 and 2003, corrections published in 2007
3	SQL/CLI	Completed in 1995, revised in 1999 and 2003, corrections published in 2005
4	SQL/PSM	Completed in 1996, revised in 1999 and 2003, corrections published in 2007
5	SQL/Bindings	Established as a separate part in 1999, but merged back into Part 2 in 2003; there is currently no Part 5
6	SQL/Transaction	Project canceled; there is currently no Part 6
7	SQL/Temporal	Withdrawn; there is no Part 7
8	SQL/Objects and Extended Objects	Merged into Part 2; there is no Part 8
9	SQL/MED	Started after 1999, completed in 2003, corrections published in 2005
10	SQL/OLB	Completed as ANSI standard in 1998, ISO version completed in 1999, revised in 2003, corrections published in 2007
11	SQL/Schemata	Extracted to a separate part in 2003, corrections published in 2007
12	SQL/Replication	Project started in 2000, but subsequently dropped; there currently is no Part 12
13	SQL/JRT	Completed as ANSI standard in 1999, revision completed in 2003, corrections published in 2005
14	SQL/XML	Completed in 2003, expanded in 2006, corrections published in 2007

O que é SQL (Structured Query Language)?

- SQL é uma **linguagem** de programação reconhecida internacionalmente usada para **definição e manutenção de bancos de dados relacionais**
- Não existe fora do universo relacional
  - É Fundamentada no modelo relacional
- É **declarativa**:
  - Os detalhes de implementação são deixados para os SGBDs relacionais.

Questão 01 - CESPE/UnB – SERPRO/2013

[75] A linguagem de consulta estruturada, ou Structured Query Language (SQL), que serve para descrever estruturas de dados e esquemas, é uma linguagem de pesquisa declarativa padrão para banco de dados relacionais.

A origem e o padrão SQL

1974

- Versão original desenvolvida pela IBM - SEQUEL

1986

- Primeiro padrão SQL publicado pelo ANSI – SQL - 86

1989

- SQL – 89 – Padrão estendido da linguagem (2ª versão)

Histórico do padrão SQL/ANSI

- ANSI lançou uma atualização em 1992
  - Conhecida como SQL92 ou SQL2
- Outra em 1999
  - Chamada de SQL99 ou SQL3
    - Stored procedures e triggers
- A próxima atualização feita em 2003
  - Também é referida como (**SQL2003**)
    - Introduz características relacionadas a XML

Histórico do padrão SQL/ANSI

- SQL2003
  - Novos Tipos: BIGINT, MULTISET e XML
  - Adiciona um grande número de funções numéricas
  - Adiciona a cláusula TABLESAMPLE à cláusula FROM (estatísticas)
  - Alguns elementos do núcleo do SQL99 foram deletados no SQL3, incluindo:
    - Os tipos de dados BIT E BIT VARYING
    - A cláusula UNION JOIN
    - A instrução UPDATE ... SET ROW

Histórico do padrão SQL/ANSI

- SQL2006
  - Descreve como SQL e XML (eXtensible Markup Language) interagem
- SQL2008
  - Legaliza ORDER BY fora da definições do cursor
  - Adiciona gatilhos de INSTEAD OF
  - Adiciona a instrução TRUNCATE
- SQL2011
  - What´s new in SQL: 2011?

DDL, DML, DQL, DTL e DCL

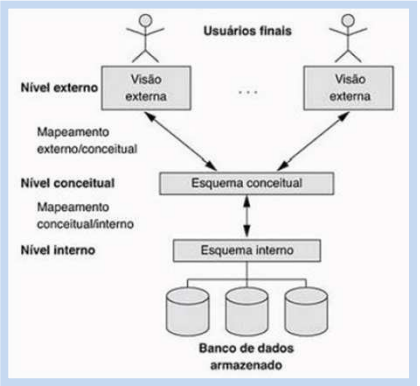
- **Linguagem de definição de dados** (DDL), em inglês, Data Definition Language
  - Contém comandos que criam, modificam ou destroem objetos de banco de dados, incluindo CREATE, ALTER, DROP ...
- **Linguagem de manipulação de dados** (DML), em inglês, Data Manipulation Language
  - Fornece comandos de manipulação de dados específicos como SELECT, INSERT, UPDATE e DELETE.
- Linguagem de Consulta de dados (DQL), em inglês, Data Query Language
  - É um subconjunto da DML, portanto também serve para manipular dados. Possui apenas o comando SELECT.

DDL, DML, DQL, DTL e DCL

- Linguagem de transação de dados (DTL)
  - Inclui comandos de COMMIT, ROLLBACK, or SAVEPOINT .
- Linguagem de Controle de dados (DCL), em inglês, Data Control Language
  - Contém os comandos relacionados com permissões ou privilégios. Os mais conhecidos são GRANT e REVOKE

Linguagem de Banco de Dados (Navathe)

- DDL - Linguagem de Definição de Dados (conceitual)
- SDL - Linguagem de Definição de Armazenamento (interno)
- VDL - Linguagem de Definição de Visões (externo)
- DML - Linguagem de Manipulação de Dados
- DCL - Linguagem de Controle de Dados
- DTL - Linguagem de Transação de Dados



ITnerante

Dialetos do SQL

- PL/SQL
  - Encontrado no Oracle. PL/SQL significa Procedural Language/SQL e contém muitas similaridades com a linguagem Ada.
- Transact-SQL
  - Usados por ambos Microsoft SQL Server e Sybase Adaptive Server como Microsoft e Sybase mudaram de uma plataforma comum que eles compartilhavam no inicio dos anos 90, suas implementações de Transact-SQL também divergiram.
- PL/pgSQL
  - Dialeto e extensões SQL implementadas no PostgreSQL. As iniciais significam Procedural Language/PostgreSQL.

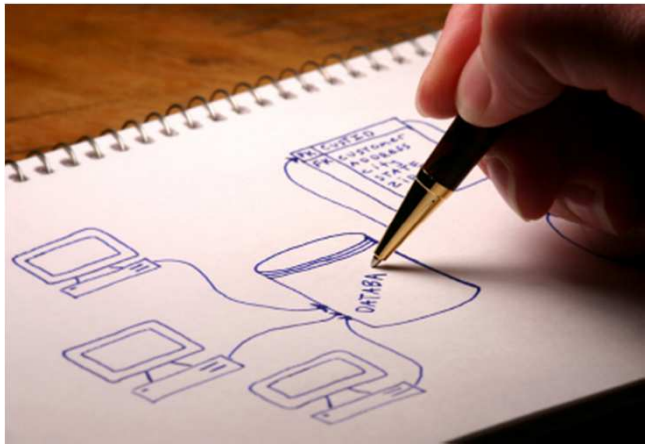
Partes da linguagem SQL (Silberchatz)

- Linguagem de definição de dados (DDL)
- Linguagem interativa de manipulação de dados (DML)
- Incorporação DML (*Embedded SQL)*
- Definição de Visões(DDL)
- Autorização (DDL)
- Integridade (DDL)
- Controle de Transações

ITnerante

TIMASTERS

SQL/ANSI




CREATE SCHEMA
SELECT
DROP TABLE
FROM
WHERE
CREATE TABLE
DELETE
DISTINCT
INSERT
UPDATE
ORDER BY
CREATE DOMAIN
GROUP BY
CREATE INDEX
HAVING
DROP INDEX

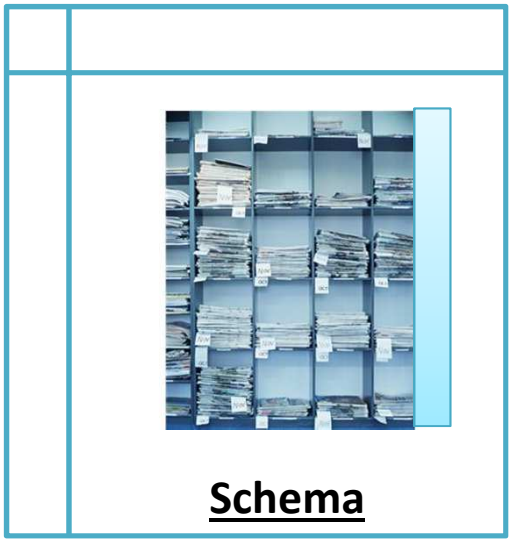
Data Definition Language

```
ALTER TABLE myLibrary ADD
COLUMN book_id INTEGER
```

```
CREATE TABLE myLibrary
(all_my_books VARCHAR(4000));
```

```
DROP SCHEMA library;
```

```
CREATE SCHEMA library;
```

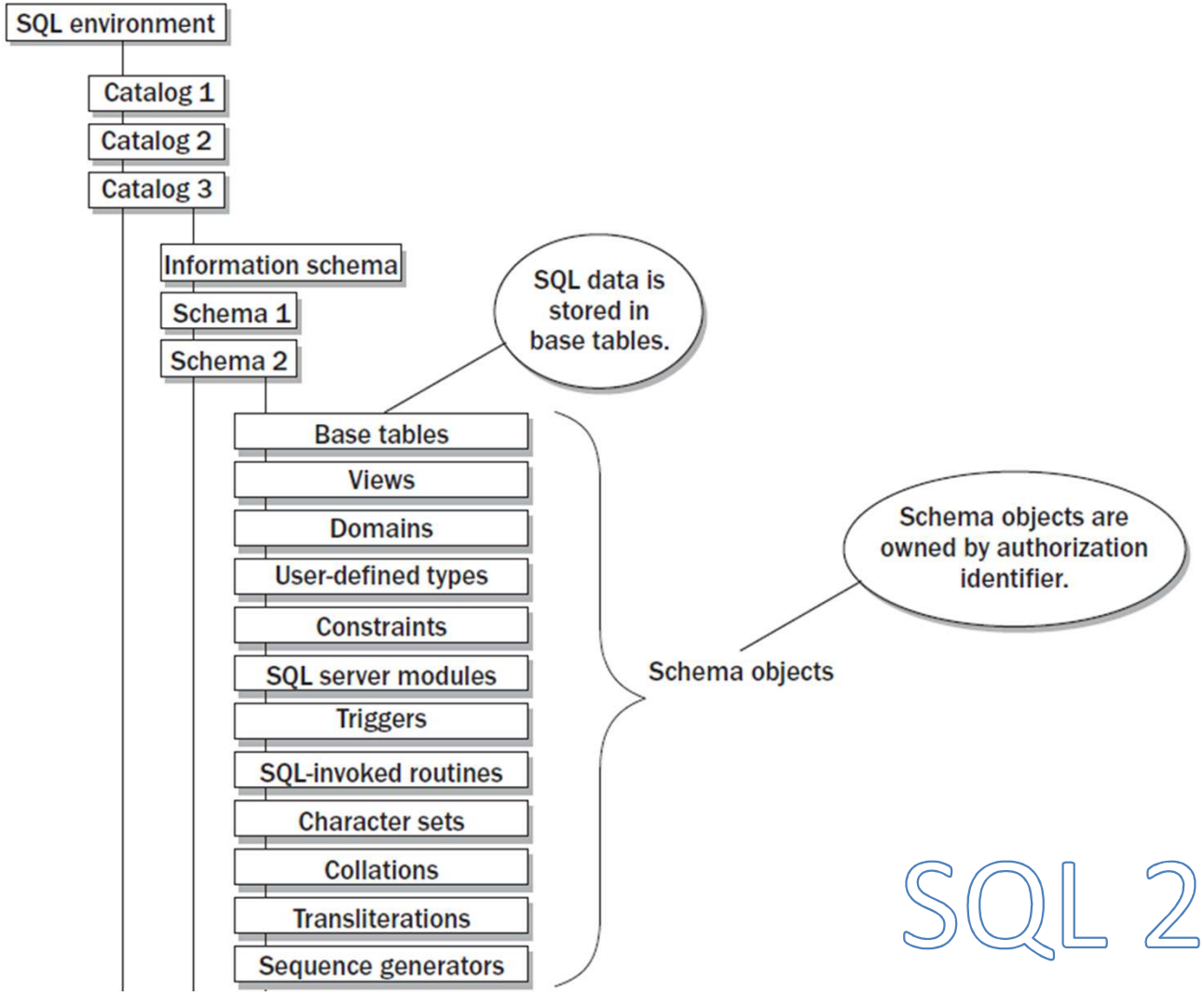


DDL

Conceito de catálogo

- Uma coleção de esquemas em um ambiente SQL que recebe um nome
- Catálogo: ITnerante
- Contém sempre um esquema especial chamado **INFORMATION_SCHEMA**
  - Proporciona informações sobre todos os esquemas do catálogo e todos os descritores de seus elementos
- **Uma instância do SGBD!**

Componentes de um catálogo



SQL 2006

ITnerante

Conceito de Esquema

- É um modo de agrupar as tabelas e outros construtores que pertencem à mesma aplicação de um banco de dados
- Identificado por um nome de esquema e inclui um identificação de autorização, que indica o usuário ou a conta a qual o esquema pertence
- Elementos: **tabelas**, **restrições**, **visões**, **domínios** e outros construtores (como concessão de autoridade)

Criando um SCHEMA

- É preciso de privilegio explicitamente concedido para criação de esquemas.
- Geralmente: usuários relevante, administradores de sistemas ou DBAs.

```
CREATE SCHEMA CONCURSO_PUBLICO
AUTHORIZANTION RCTHIAGO;
```

```
CREATE SCHEMA INVENTORY AUTHORIZATION MNGR
DEFAULT CHARACTER SET Latin1
CREATE TABLE ARTISTS
( ARTIST_ID INTEGER, ARTIST_NAME CHARACTER (20) ) ;
```

Questão 2 – CESPE/UnB - CNJ - Técnico Judiciário - Programação de Sistemas - 2013

Com relação a conceitos básicos de banco de dados, características dos bancos relacionais e linguagem SQL, julgue o item a seguir.

[112] Um esquema de um SGBD é identificado por um nome e uma identificação de autorização, que indica o usuário ou conta a qual o esquema pertence, bem como os descritores de cada elemento.

Alterações de esquemas SQL

DROP SCHEMA <schema name>
CASCADE | RESTRICT;

- Para remover o esquema com todas as suas tabelas, domínios e outros elementos, deve-se usar a opção **CASCADE.**
- Se a opção **RESTRICT** for usada o esquema será eliminado somente se não contiver nenhum(a) elemento/informação.

Criando um Banco de dados


```
CREATE DATABASE library;
```

- A declaração acima cria um banco de dados chamado *library* em seu SGBD (seja ele Microsoft SQL Server, IBM DB2, PostgreSQL e MySQL)
- Se você tem privilégios suficientes na instância SGBD, o comando vai criar um banco de dados
  - Uma **estrutura lógica** para armazenar seus dados, juntamente com todas as **estruturas de apoio**, **arquivos**, e outros objetos necessários para suas operações.
  - Você não precisa saber nada sobre isso, todos os espaços são preenchidos com valores padrão.
    - Eis o **poder de uma linguagem declarativa**!

Apagando um Banco de Dados

- Uma vez criado, um banco de dados pode ser destruído facilmente,
  - Usando instrução SQL: DROP
- Você não pode destruir objetos que não existem
  - SGBD irá avisá-lo(ERROR) sobre isso se você tentar

```
DROP DATABASE library;
```

Criando uma tabela

- A tabela é o lugar onde todos os seus dados serão armazenados
- Criar uma tabela é tão fácil quanto criar o banco de dados, vc precisa:
  - Especificar um nome para a coluna da tabela e seu tipo de dados

```
CREATE TABLE myLibrary
(all_my_books VARCHAR(4000));
```

Comando CREATE TABLE

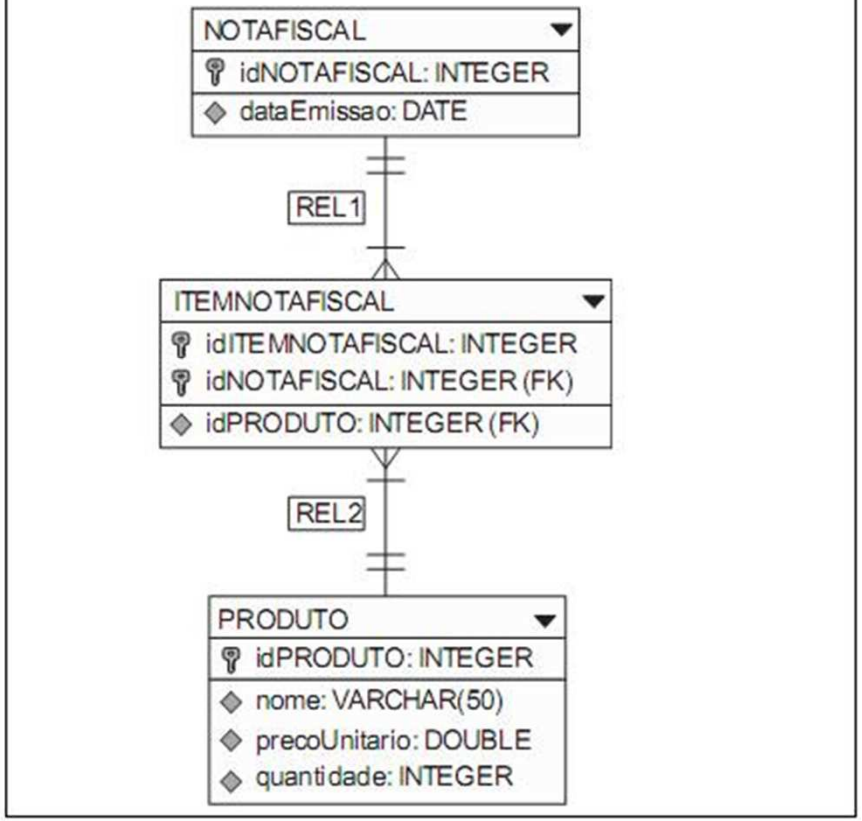
- Usado para especificar uma nova relação, dando-lhe um **nome** e especificando seus **atributos** e **restrições iniciais**.
- Os **atributos** são definidos **primeiro** e, a cada atributo, é dado **um nome**, **um tipo** para especificar o **domínio** de seus valores e alguma **restrição de atributo**.
- As restrições de **chave**, a **integridade** de entidade e a integridade referencial **podem** ser especificas no mesmo comando após os atributos forem declarados.

Exemplo

```
CREATE TABLE DEPARTAMENTO
        (DNOME VARCHAR(15) NOT NULL ,
        DNUMERO INT NOT NULL ,
        GERSSN CHAR(9) NOT NULL ,
        GERDATAINICIO DATE,
PRIMARY KEY (DNUM) ,
UNIQUE (DNOME) ,
FOREIGN KEY (MGRSSN) REFERENCES
EMPREGADO(SSN) ) ;
```

Questão 03 - FCC - DPE-SP - Agente de Defensoria – Programador - 2013

Para responder às questões, utilize a figura abaixo, parte de um modelo Entidade-Relacionamento de um banco de dados.



Questão 03 - FCC - DPE-SP - Agente de Defensoria – Programador - 2013

Uma instrução SQL correta para criar a tabela NOTAFISCAL apresentada no modelo é:

a) CREATE TABLE NOTAFISCAL (idNOTAFISCAL INTEGER NOT NULL, dataEmissao DATE NULL, PRIMARY KEY(idNOTAFISCAL));

b) CREATE TABLE NOTAFISCAL (idNOTAFISCAL INTEGER NULL AUTOINCREMENT, dataEmissao DATE NULL, PRIMARY KEY(idNOTAFISCAL));

c) CREATE SCHEMA NOTAFISCAL (idNOTAFISCAL INTEGER NOT NULL AUTO_INCREMENT, dataEmissao DATE NOT NULL, PRIMARY KEY(idNOTAFISCAL));

d) CREATE TABLE NOTAFISCAL (idNOTAFISCAL INTEGER NULL, dataEmissao DATE NOT NULL, CONSTRAINT UNIQUE KEY(idNOTAFISCAL));

e) CREATE TABLE NOTAFISCAL (idNOTAFISCAL INTEGER NULL CONSTRAINT PRIMARY KEY, dataEmissao DATE NULL);

Alterações em tabelas

- DROP
  - **DROP TABLE** DEPENDENTE **CASCADE**;
- O comando drop table remove todas as informações de uma relação do banco de dados.

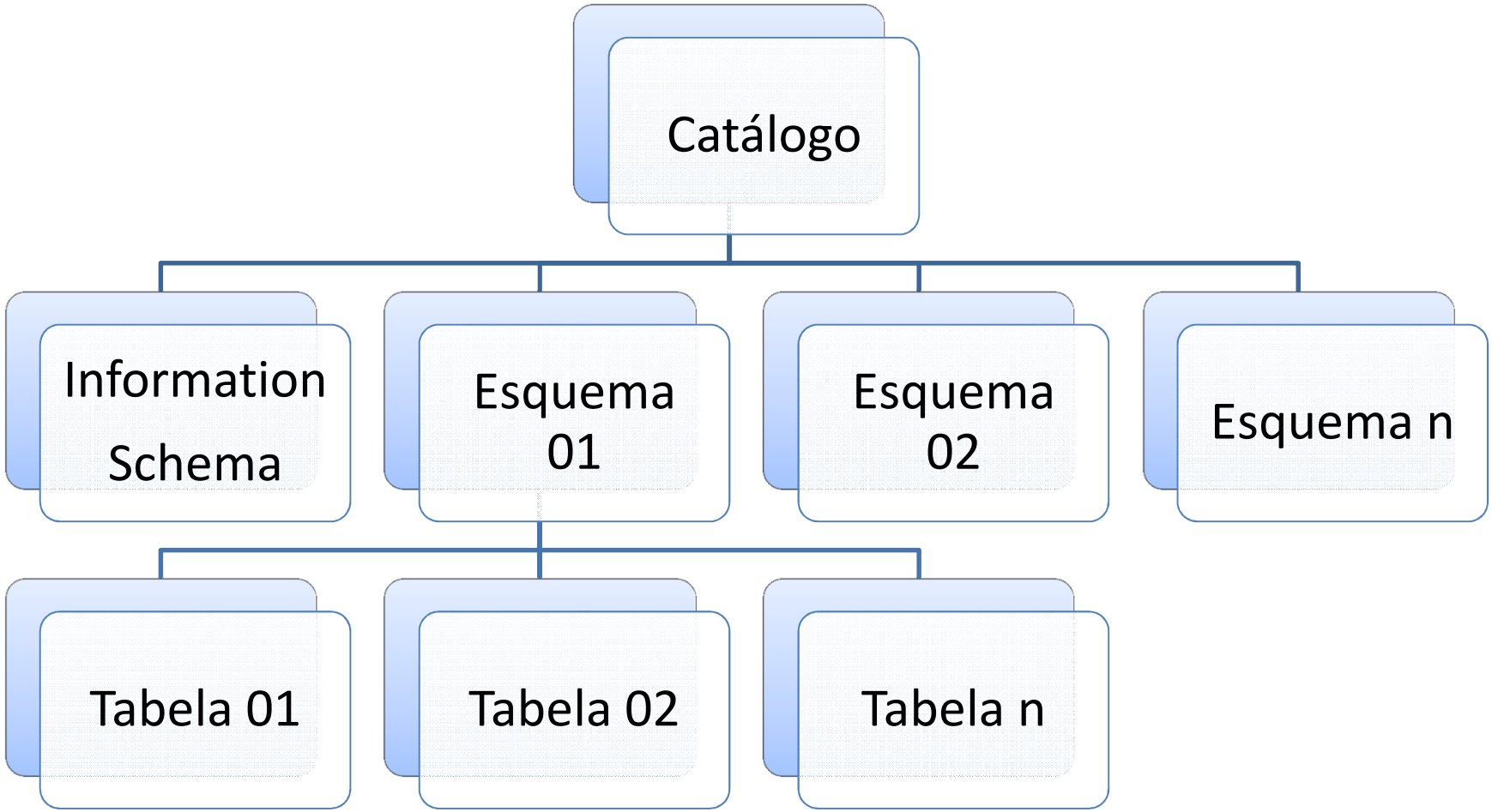
Alterações em tabelas

- O comando **ALTER TABLE** é usado para adicionar atributos a uma relação existente.
- Todas as tuplas da relação recebem valores nulo para seu novo atributo.

```
ALTER TABLE <table name>
            ADD [COLUMN] <column definition>
          | ALTER [COLUMN] <column name>
                  { SET DEFAULT <default value> | DROP DEFAULT }
          | DROP [COLUMN] <column name> { CASCADE | RESTRICT }
```

- Exemplos:
  - **ALTER TABLE** EMPRESA.EMPREGADO **ADD** FUNCAO **VARCHAR(12)**;
  - **ALTER TABLE** EMPRESA.EMPREGADO **DROP** ENDERECO **CASCADE;**

Catálogo, Esquema, Tabela



Considerações a respeito de nomes de objetos

- Um **identificador** é um nome dado a um objeto SQL.
  - Até 127 caracteres
  - **Regular identifiers** e **delimited identifiers**
- **Regular**
  1. Os nomes não são case-sensitive
     - ITnerante_table == ITNERANTE_TABLE
  2. Somente letras, números e underline são permitidos.
  3. Nenhuma palavra-chave reservadas de SQL pode ser usada.

Considerações a respeito de nomes de objetos

- **Delimited** (Delimitado)
  1. O identificador deve ser incluído dentro de um conjunto de aspas duplas, como o "ArtistNames" identificador.
  2. As aspas não são armazenadas no banco de dados, mas todos os outros caracteres são armazenados como eles aparecem na instrução SQL.
  3. Os nomes são case-sensitive.
  4. A maioria dos caracteres são permitidas, incluindo espaços.
  5. Palavras-chave reservadas de SQL podem ser usadas.

Qualified Names

- Todos os identificadores de objeto de esquema são qualificados pelo **caminho lógico** em que eles se encaixam **na estrutura hierárquica do ambiente SQL**.
- Um nome qualificado inclui o **nome do catálogo**, o **nome do esquema**, e o **nome do objeto** no esquema, cada um separado por um ponto.
  - Por exemplo:
    - A tabela: CD_ARTISTS.
    - Do esquema: COMPACT_DISCS
    - Do catálog: MUSIC
  - O fully qualified name dessa tabela será:

MUSIC.COMPACT_DISCS.CD_ARTISTS

Tipos de dados

character strings	binary strings	exact numeric	approximate numeric	datetime types
CHARACTER	BINARY	NUMERIC	FLOAT	DATE
CHARACTER VARYING	BINARY VARYING	DECIMAL	REAL	TIME
CHARACTER LARGE OBJECT	BINARY LARGE OBJECT	SMALLINT	DOUBLE PRECISION	TIMESTAMP
		INTEGER		
		BIGINT		

Cadeia de caracteres - Character Data

- Tamanho fixo ou variável?
  - Todas as cadeias de caracteres em SQL podem ser de comprimento fixo ou variável.
  - Como você pode imaginar, essa flexibilidade tem um preço de desempenho, pois o SGBD deve executar a tarefa adicional de alocação dinâmica.

CHARACTER

- Especifica o número exato de caracteres que serão armazenados para cada valor.
  - Por exemplo, se você definir o número de caracteres de 10, mas o valor contém apenas seis caracteres os restantes quatro caracteres serão espaços.
  - O tipo de dados pode ser abreviado como CHAR.
  - Exemplo:
    - ARTIST_NAME **CHAR(60)**

CHARACTER VARYING

- Especifica o número máximo de caracteres que pode ser incluído em um dado.
- O número de caracteres armazenados é exatamente o mesmo número que o valor introduzido, de modo nenhum espaço é adicionado ao valor.
- O tipo de dados pode ser abreviado como VARYING CHAR ou VARCHAR.
- Exemplo: ARTIST_NAME **VARCHAR (60)**

Strings binárias

- A string binária é uma sequência de bytes da mesma forma que uma string de caracteres é uma sequência de caracteres
- Ao contrário de strings de caracteres que geralmente contêm informações na forma de texto
  - É usada para armazenar dados não tradicionais, tais como **imagens, áudio e arquivos de vídeo, executáveis de programa**, e assim por diante.
  - BINARY LARGE OBJECT → BLOB

CHARACTER x BINARY LARGE OBJECT

- CHARACTER LARGE OBJECT
  - Armazena grandes grupos de **caracteres**, até o valor especificado.
  - O número de bytes armazenados é exatamente o mesmo número que o valor introduzido
  - O tipo de dados também pode ser referido como CLOB.
    - Exemplo: ARTIST_BIO CLOB (200K)
- BINARY LARGE OBJECT
  - Armazena grandes grupos de **bytes**, até o valor especificado.
  - O tipo de dados também pode ser referido como BLOB.
    - Exemplo: ARTIST_PIC BLOB (1M)

Binary Data

- Os dados binários são de fácil compreensão e interpretação pelos computadores
  - Que podem transformá-los de forma que os seres humanos possam entender.
- Os principais exemplos de dados binários são imagens
  - Apenas o seu editor de imagem saber como organizar esses zeros e uns em uma imagem de sua(seu) namorada(o)
  - Documentos Adobe Acrobat, o conteúdo de um arquivo PDF, e assim por diante

NATIONAL CHARACTER

- NATIONAL CHARACTER (NCHAR(x))
- NATIONAL CHARACTER VARYING (Nome_variavel NCHAR VARYING (x))
- NATIONAL CHARACTER LARGE OBJECT (Nome_variavel NCLOB(200K))
  - Baseado em conjunto de caracteres definido

XML

- A Extensible Markup Language (XML) é uma linguagem de marcação de uso geral
  - Usada para descrever documentos em um formato que seja **conveniente para exibição** em páginas da web e **para a troca de dados** entre diferentes aplicações.
- As especificações para armazenar dados XML em bancos de dados SQL foram adicionados ao padrão SQL no SQL: 2003.
  - Exemplo: ARTIST_BIO XML(DOCUMENT(UNTYPED))

Caracteres em SGBDs comerciais

SQL STANDARD	ORACLE 11G	DB2 9.7	MS SQL SERVER 2008
CHARACTER	CHARACTER	CHARACTER	CHARACTER
VARYINGVARCHAR	VARYING VARCHAR VARCHAR2 LONG VARCHAR	VARYING VARCHAR LONG VARCHAR	VARYING VARCHAR TEXT
CLOB or CHARACTER LARGE OBJECT	CLOB	CLOB	VARCHAR(MAX)
NCHAR	NCHAR	GRAPHIC	NCHAR
NCHAR VARYING(n)	NCHAR VARYING NVARCHAR2	VARGRAPHIC LONG VARGRAPHIC	NVARCHAR
NATIONALCHARACTER LARGE OBJECT	NCLOB	DBCLOB	NVARCHAR(MAX)

Cadeia de caracteres - Tipos de dados

- Tamanho fixo:
  - CHAR(n) ou CHARACTER(n)
- Tamanho variável:
  - VARCHAR(n) ou CHAR VARYNG ou CHARACTER VARYING (n)
  - CLOB
- Binary String: BLOB

Dados Numéricos

- Depois dos caracteres vem os números.
- Um número é sempre um número, certo?
  - Talvez, não.
- Eles são divididos em duas grandes categorias:
  - Números exatos (exact numbers)
  - Números aproximados (approximate)

Números exatos

- Os números exatos podem ser números inteiros (lápis, pessoas, ou planetas) ou tem casas decimais (preços, pesos ou percentuais).
- Os números podem ser positivos e negativos.
- Eles podem ter precisão e escala.
  - **Precisão(p)** determina o número total **máximo de dígitos** decimais que podem ser armazenados (tanto para a esquerda e para a direita do ponto decimal).
  - **Escala(e)** especifica o número máximo de casas decimais permitidas.
- São acomodados por todos o SGBDS.

Números exatos em SGBDs proprietários

SQL STANDARD	ORACLE 11G	DB2 9.5	MS SQL SERVER 2008
INTEGER	NUMBER(38) INT	INTEGER BIGINT	INTEGER BIGINT
SMALLINT	SMALLINT NUMBER(38)	SMALLINT	SMALLINT TINYINT
NUMERIC	NUMERIC DECIMAL NUMBER	NUMERIC DECIMAL	NUMERIC DECIMAL MONEY SMALLMONEY

Números exatos no Postgresql e MySQL

SQL STANDARD	POSTGRESQL	MYSQL
INTEGER	INTEGER BIGINT	INTEGER BIGINT
SMALLINT	SMALLINT	SMALLINT TINYINT
NUMERIC	NUMERIC	NUMERIC

Números aproximados

- *Números aproximados são números que não podem ser representados com precisão absoluta (ou não ter um valor preciso).*

SQL STANDARD	ORACLE 11G	DB2 9.5	MS SQL SERVER 2008
FLOAT	FLOAT NUMBER	FLOAT	FLOAT
REAL	REAL NUMBER	REAL	REAL
DOUBLE PRECISION	DOUBLE PRECISION NUMBER	DOUBLE PRECISION	DOUBLE PRECISION

Tamanho em bytes e range

DATA TYPE	STORAGE SIZE (BYTES)	RANGE	NOTES
INTEGER	4	-2,147,483,648 to +2,147,483,647	Implemented in all RDBMSs
TINYINT	1	0 through 255	MS SQL Server only
SMALLINT	2	–32,768 to + 32,768	Implemented in all RDBMSs
BIGINT	8	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,808	Implemented in all RDBMSs
MONEY	8	– 922,337,203,685,477.5808 to + 922,337,203,685,477.5807	MS SQL Server only
SMALLMONEY	4	– 214,748.3648 to + 214,748.3647	MS SQL Server only
REAL	*4*	The range is from negative 3.402E + 38 to negative 1.175E – 37, or from positive 1.175E – 37 to 3.402E + 38. It also includes 0.	Implemented in all RDBMSs
FLOAT	*4 to 8*	The number can be zero or can range from –1.79769E + 308 to –2.225E – 307, or from 2.225E – 307 to 1.79769E + 308.	Implemented in all RDBMSs (if only as synonyms)
DOUBLE	*8*	The number can be zero or can range from –1.79769E + 308 to –2.225E – 307, or from 2.225E – 307 to 1.79769E + 308.	Implemented in all RDBMSs

Apresentamos os tipos numéricos e seu relacionamento com os SGBDs.

ITnerante

Questão 04 - ESAF - CVM - Analista de TIC - Infraestrutura - 2010

Na linguagem SQL

a) char(n): uma *string* de caracteres de tamanho fixo n, especificado pelo usuário.

b) floatchar(n): uma *string* de caracteres de tamanho variável máximo *n*, especificado pelo usuário.

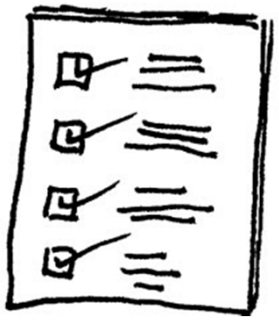
c) var(n): um número de ponto variável, com precisão de até *n* dígitos.

d) close(n): uma *string* de aproximação de caracteres numéricos de tamanho fixo n, definido pela linguagem.

e) doublefloat(n): um número de ponto flutuante duplo, com precisão modificada.

Tipos de dados Numéricos

- Inteiros
  - INT
  - SMALLINT
  - NUMERIC(p,e)
  - DECIMAL(p,e)
- Ponto flutuante(Real)
  - FLOAT(p)
  - REAL
  - DOUBLE PRECISION



DATE

- DATE é uma estrutura que consiste em três elementos: ano, mês e dia.
  - O ano é um número de 4 dígitos que permite que os valores de 0000 a 9999
  - O mês é um elemento de 2 dígitos com valores de 01 a 12, e
  - O dia tem 2 dígitos com um intervalo de 01 a 31.
- SQL/ANSI define a semântica de datas e hora usando a estrutura descrita acima, mas implementadores não são obrigados a usar essa abordagem, desde a implementação produza os mesmos resultados.
- Um *vendor* pode escolher algo semelhante às estruturas anteriores, outros podem implementar caracteres, números com escala diferente, e assim por diante.

2013-12-21

TIME

- TIME consiste de hora, minuto e segundo
  - A hora é um número de 00 a 23
  - O minuto é um número de dois algarismos, de 00 a 59
  - O segundo é um inteiro entre 00-59 ou um número decimal com uma escala mínima de 5 e precisão mínima de 3 que pode conter valores de 00.000 para 59,999.

DATETIME

- Combina data e hora em um único tipo, com intervalo de datas:
  - A partir de 01 de janeiro de 1753, até 31 de dezembro de 9999
  - Intervalo de tempo de 00:00:00 a 23:59:59:999, o armazenamento alocado é de 8 bytes.

ITnerante

Outros Tipos de Dados

- TIMESTAMP
  - Engloba os campos DATE e TIME, mais 6 posições para a fração decimal de segundos e uma qualificação opcional WITH TIME ZONE
- INTERVAL
  - Serve para calcular o intervalo entre dois objetos que representam tempos.

Outros tipos de dados

- BOOLEAN
  - Valores TRUE ou FALSE
- BIT
  - O tipo de dados BIT pode ser 0 ou 1, e como o nome sugere, que ocupa exatamente o bit 1 do armazenamento.
  - Este pode ser o tipo de dados subjacente para o tipo BOOLEAN no Microsoft SQL Server, Oracle e IBM DB2.
  - Usar BIT no lugar de BOOLEAN requer que uma lógica de interpretação seja implementada no aplicativo cliente.

Sobre valores nulos de atributos

- **Valor nulo** é um membro de **todos** os tipos de domínios.
- Declarando um domínio de atributo como sendo **NOT NULL** proíbe-se, assim, a inserção de valores nulos para esse tipo de atributo.

ITnerante

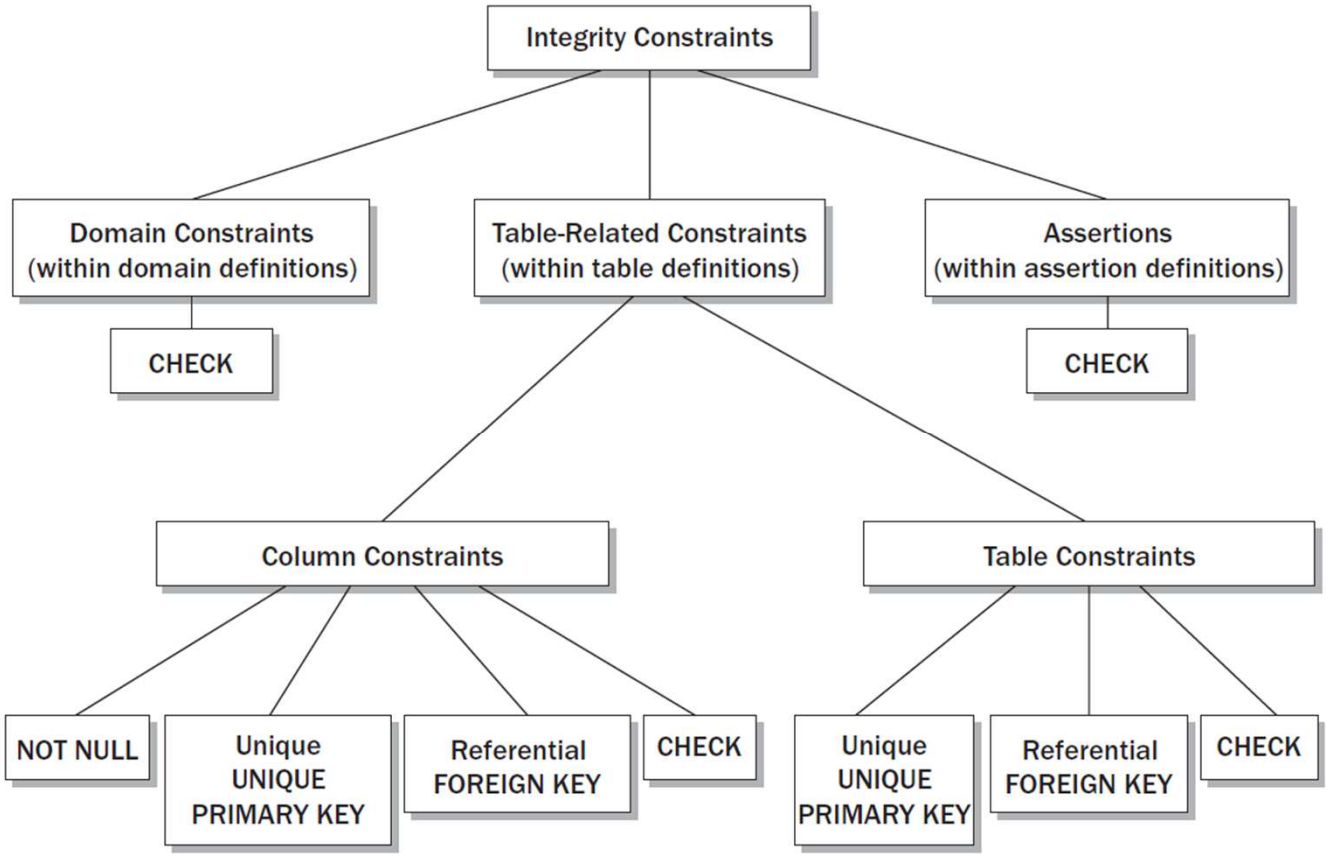
TIMASTERS

Restrições em SQL

Restrições em SQL

- Restrições de integridade SQL, que são geralmente referidos simplesmente como restrições, podem ser divididos em três categorias:
  - Restrições em tabelas
  - Assertions
  - Restrições de domínio.

Hierarquia das restrições de integridade

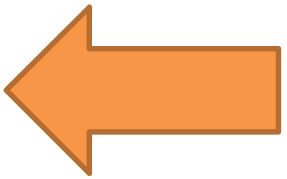


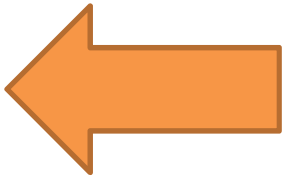
RESTRIÇÕES BÁSICAS EM SQL

- Restrições de chave
  - Chave simples:
    - DEP_NUMERO INT **PRIMARY KEY**;
  - Chave composta:
    - ... **PRIMARY KEY** (DNUM, DLOCALIZACAO)
- Integridade Referencial
  - **FOREIGN KEY** (DNU) **REFERENCES** DEPARTAMENTO (DNUM);
  - Ação referencial engatilhada
    - Referencial triggered action
    - **(CASCADE | SET NULL | SET DEFAULT | RESTRICT | NO ACTION) x (ON DELETE | ON UPDATE)**

Exemplo

```
CREATE TABLE EMPREGADO (...,
        DNO       INT       NOT NULL     DEFAULT  1,
CONSTRAINT EMPPK
        PRIMARY KEY  (SSN)  ,
CONSTRAINT  EMPSUPERFK
        FOREIGN KEY  (SUPERSSN)  REFERENCES
        EMPREGADO (SSN)  ON DELETE  SET NULL
        ON UPDATE  CASCADE,
CONSTRAINT  EMPDEPTFK
        FOREIGN KEY (DNO) REFERENCES
        DEPARTAMENTO (DNUMERO) ON DELETE
        SET DEFAULT ON UPDATE CASCADE );
```





RESTRIÇÕES BÁSICAS EM SQL

- NOT NULL
  - DNUMERO INT **NOT NULL**, ..
- UNIQUE
  - CD_NAME VARCHAR(60) **UNIQUE, ..**
  - CONSTRAINT UN_ARTIST_CD **UNIQUE** (ARTIST_NAME, CD_NAME)
- DEFAULT
  - TORCEDOR_DO VARCHAR(30) **DEFAULT** 'SPORT'
- CHECK
  - DNUMERO INT **CHECK** (DNUMERO > 0 AND DNUMERO < 21)
  - CREATE DOMAIN D_NUM AS INTERGER **CHECK** (D_NUM >0 AND D_NUM < 21)

Definindo a restrição CHECK

- É a restrição mais flexível de todas
  - Sua sintaxe básica é relativamente simples
  - Para criar uma restrição do tipo CHECK em uma **coluna** você deve utilizar a seguinte sintaxe:
    - <column name> { <data type> | <domain> } CHECK (<search condition>)
  - No caso da criação da restrição em uma **tabela**:
    - [CONSTRAINT <constraint name>] CHECK (<search condition>)

Restrições Genéricas

- SQL tem capacidade de especificar as restrições genéricas do tipo CHECK que podem ser aplicadas a várias tabelas, são as chamadas asserções (Assertion)
  - CREATE ASSERTION CHECK (condição de busca)

```
CREATE ASSERTION LIMIT_IN_STOCK CHECK
( ( SELECT SUM (IN_STOCK) FROM CD_TITLES ) < 5000 );
```

```
CREATE ASSERTION Picnic_Account_Check
  CHECK (NOT EXISTS (SELECT * FROM Picnics) OR
         EXISTS (SELECT * FROM Accounts WHERE balance > 0));
```

Domínios

- É possível especificar o tipo de dados de cada atributo diretamente, porém um domínio pode ser declarado e o nome do domínio usado com a especificação do atributo.

```
CREATE DOMAIN <domain name> [ AS ] <data type>
[ DEFAULT <default value> ]
[ CONSTRAINT <constraint name> ] CHECK ( <search condition> )
```

```
CREATE DOMAIN STOCK_AMOUNT AS INT
CONSTRAINT CK_STOCK_AMOUNT CHECK (VALUE BETWEEN 0 AND 30 );
```

Questão 05 - ESAF - ANA - Analista Administrativo - TI – Desenvolvimento - 2009

Em SQL, a cláusula check aplicada a uma declaração de domínio

a) permite especificar um predicado que deve ser satisfeito por qualquer valor atribuído a uma variável de determinado domínio.

b) especifica um predicado que deve ser satisfeito por uma tupla em uma relação.

c) proíbe a inserção de um valor nulo para as variáveis do domínio.

d) verifica se os atributos considerados formam uma chave candidata.

e) não tem efeito, pois não se aplica esta cláusula a declarações de domínio.

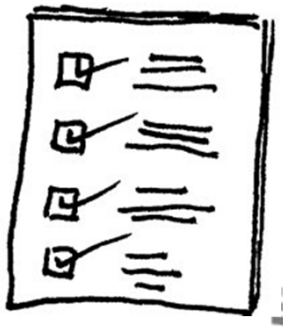
Resumindo: Restrições (constraint)

- O SQL contem 6 tipos de restrições:

PRIMARY KEY	UNIQUE
FOREING KEY	CHECK
NOT NULL	DEFAULT

- Estes podem ser criadas tanto no “crate table” ou no “alter table”

Assertions + Domain



ITnerante

ITnerante

TIMASTERS

DML
DATA MANIPULATION LANGUAGE

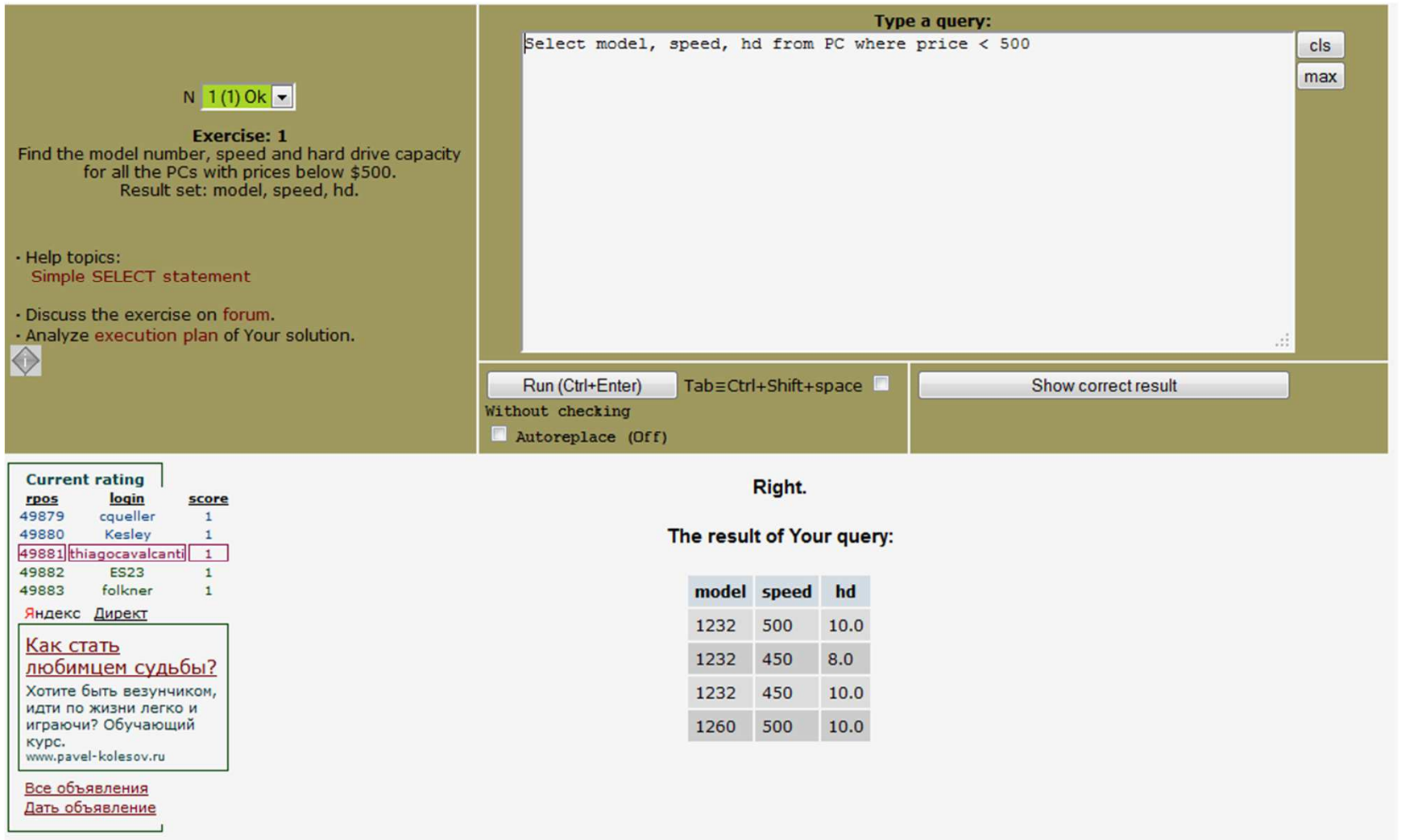
Prof. Thiago Cavalcanti

ITnerante

Aprendendo DML

ITnerante

Aprendendo DML



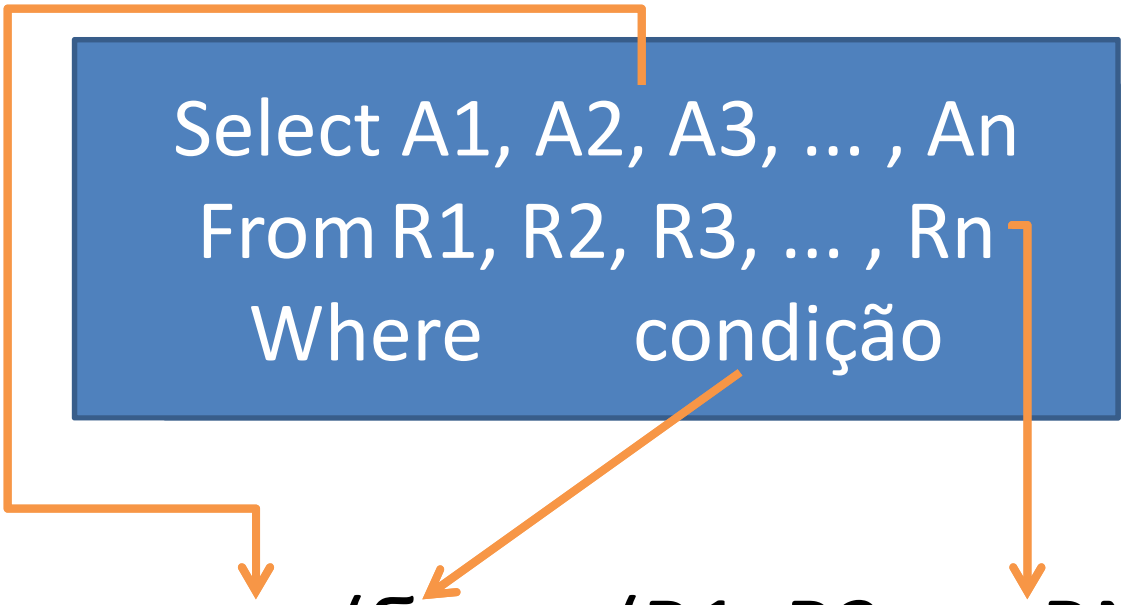
ITnerante

Outra opção

- http://sqlzoo.net/

ITnerante

Comando Select



Relembrando:
Π – pi - projeção
б – sigma – seleção
x – produto cartesiano

CONSULTAS EM SQL

SELECT [**DISTINCT** | **ALL**] { * | <select list> }
FROM <table reference> [{ , <table reference> } . . .]
[**WHERE** <search condition>]
[**GROUP BY** <grouping specification>]
[**HAVING** <search condition>]
[**ORDER BY** <order condition>]

Comentários

- Uso do asterisco (SELECT * FROM ...)
- Ausência da clausula WHERE
- Aliases (Pseudônimos)[1]
- Variáveis de Tuplas[1]

1- Veremos mais detalhes a seguir! ☺

Aliases (Pseudônimos)

- A SQL possui um mecanismo para renomear **tanto relações quanto atributos** através da cláusula **as**, da seguinte forma:

nome_antigo **AS** novo_nome

SELECT column_name **AS** alias_name
FROM table_name

<derived column> [[AS] <column name>]

SELECT column_name(s)
FROM table_name
AS alias_name

Variáveis de Tuplas

- **Variáveis de tuplas** são definidas na cláusula **FROM** através do uso da cláusula **AS** (Opcional).
- Exemplo: Encontre o nome dos clientes e seus números de empréstimo para todos os clientes que possuem um empréstimo em alguma agência.

```
SELECT DISTINCT  nome_cliente, T.numero_emprestimo
FROM  devedor AS T, emprestimo AS S
WHERE  T.numero_emprestimo =S.numero_emprestimo;
```

Questão 06 - ESAF - CGU - Analista de Finanças e Controle - Desenvolvimento de Sistemas - 2012

É correto afirmar que na *SQL*

a) uma constante de tupla precisa conter uma relação de atributos.

b) a tupla de um atributo contém relações entre variáveis.

c) uma variável de tupla precisa estar associada a uma relação.

d) uma tupla de variável precisa conter uma tupla de relação.

e) variáveis e tuplas são especificadas por objetos relacionados.

Tabelas como conjuntos: DISTINCT e ALL

- Para forçar a eliminação de duplicatas, deve-se inserir declaração **DISTINCT** depois do **SELECT**.
- Achar os nomes de todas as agências na relação empréstimo e remover as duplicatas

  SELECT DISTINCT nome_agencia

  FROM emprestimo
- A declaração ALL especifica que duplicatas não devem ser removidas.

  SELECT ALL nome_agencia

  FROM emprestimo

Usando operadores no Select

- A cláusula **SELECT** pode conter expressões aritméticas envolvendo os operadores +, - , * , e /, e operações em constantes ou atributos de tuplas.
- A consulta:

SELECT nome_agencia, numero_emprestimo,
total * 100 **FROM** emprestimo;

retornaria uma relação a qual é a igual a relação empréstimo, exceto que o atributo total é multiplicado por 100.

A Cláusula where

- Não é necessário que os atributos usados na clausula where estejam listados no select.
- A cláusula **where** corresponde ao predicado de **seleção** da álgebra relacional
- Ele consiste de um predicado envolvendo atributos das relações que aparecem na cláusula from.
- Operadores aritméticos
  - + , - , * , / , % (módulo)
- Operadores de comparação
- Conectivos ou operadores lógicos
  - AND, OR e NOT

Operadores de comparação

Operador	Simbolo	Exemplo
Igual	=	Qtd_Produto = 85
Diferente	<>	Qtd_Produto <> 85
Menor que	<	Qtd_Produto < 85
Maior que	>	Qtd_Produto > 85
Menor ou igual	<=	Qtd_Produto <= 85
Maior ou igual	>=	Qtd_Produto >= 85

Um pouco mais de comentários

- Operações com Strings
  - LIKE, NOT LIKE, % , _

```
SELECT * FROM Persons
WHERE City LIKE '%s‘;
```

```
SELECT * FROM Persons
WHERE City LIKE '%tav%‘;
```

```
SELECT * FROM Persons
WHERE City LIKE ‘_T_‘;
```

- Operadores aritméticos
  - BETWEEN (Inclusivo)

```
SELECT   numero_emprestimo
FROM    emprestimo
WHERE  total BETWEEN 90000 AND 100000;
```

Alguns exemplos de operações com Strings

Sample Value	Possible Query Results
'J%'	Jennifer Warnes, Joni Mitchell, Jose Carreras
'%Spark'	Court and Spark
'%Blue%'	Famous Blue Raincoat, Blue, Blues on the Bayou
'%Cline%Hits'	Patsy Cline: 12 Greatest Hits
'194_'	1940, 1942, 1947
'19__'	1900, 1907, 1938, 1963, 1999
'__ue'	Blue
'9__01'	90201, 91401, 95501, 99301, 99901
'9_3%'	9032343, 903, 95312, 99306, 983393300333

Ordenando Consultas

- Após a clausula WHERE
  - ORDER BY <nome(s)_coluna(s)> DESC ou ASC (default)
  - Separados por virgula
- Ordenação pode ser demorada! Só use quando for de fato necessários.

```
SELECT * FROM Pessoas
ORDER BY PrimeiroNome DESC
```

Vamos complicar um pouco

- Consultas um pouco mais complexas
  - Comparações envolvendo NULL e os Três Valores Lógicos
  - Consultas Aninhadas
  - Consultas Aninhadas Correlacionadas
  - Comparações de Tuplas e de Conjuntos/ Multiconjuntos
  - Função EXISTS e UNIQUE
  - Consultas Aninhadas Correlacionadas

Comparações envolvendo NULL

- É possível para as tuplas ter valor nulo, denotados por NULL, para alguns de seus atributos.
- Significa valor **desconhecido**, **não disponível** ou **inexistente**.
- O resultado de uma expressão aritmética envolvendo null é null.
- As operações envolvendo null retornam false.
- Resultado da cláusula where é tratado como false se a sua avaliação é desconhecida
  - "P is unknown" é true se a avaliação do predicado P é = unknown.

Operações aritméticas com NULL

- Nulos pode causar estragos com sua aritmética.
- Suponha que você queira calcular a diferença entre duas colunas numéricas, e uma das colunas tem um valor NULL.
  - Os resultados podem surpreendê-lo:
    - 19,99 + 0 = 19,99 (como esperado),
    - mas 19,99 + NULL = NULL.
    - O mesmo é verdadeiro para qualquer outro operador da matemática (multiplicação, divisão ou subtração).
  - Considerando a divisão por zero irá lançar um erro, a divisão por NULL vai retornar NULL.

Comparações envolvendo NULL

Conectivos lógicos na lógica de três valores.

(a)	**AND**	TRUE	FALSE	UNKNOWN
	TRUE	TRUE	FALSE	UNKNOWN
	FALSE	FALSE	FALSE	FALSE
	UNKNOWN	UNKNOWN	FALSE	UNKNOWN
(b)	**OR**	TRUE	FALSE	UNKNOWN
	TRUE	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE	UNKNOWN
	UNKNOWN	TRUE	UNKNOWN	UNKNOWN
(c)	**NOT**			
	TRUE	FALSE		
	FALSE	TRUE		
	UNKNOWN	UNKNOWN		

Unknown and true

True → True

False → False

Unknown and false

True → False

False → False

Comparações envolvendo NULL

- Encontre todos os números de empréstimo que aparecem na relação empréstimo com valores null para o total.
  - select numero_emprestimo
  - from emprestimo
  - where total is null
- Total de todos os empréstimos
  - select sum (total)
  - from emprestimo
- A declaração acima ignora valores null no total ; o resultado será null somente se não existir nenhuma tupla com o atributo total diferente de null
- Todas as operações agregadas, exceto count(*) ignoram tuplas com valores null nos atributos agregados.

Consultas Aninhadas

- SQL provê um mecanismo para aninhamento de subconsultas.
- Uma subconsulta é uma expressão select-from-where que é aninhada dentro de uma outra consulta.
- As aplicações mais comuns para as subconsultas são testes para membros de conjuntos, comparação de conjuntos e cardinalidade de conjuntos

Consultas Aninhadas

- Algumas consultas precisam que os valores existentes no banco de dados sejam buscados e depois usados em uma condição de comparação.
  - **Consulta interna x consulta externa**
- Sempre que uma condição na cláusula WHERE de uma consulta aninhada referencia algum atributo de uma relação declarada na consulta externa, as duas consultas são consideradas **correlacionadas**

Exemplo (Consulta Aninhada)

- Encontrar todos os clientes que possuem uma conta e um empréstimo no banco.

```
select distinct nome_cliente
from devedor
where nome_cliente IN (select nome_cliente
      from depositante)
```

- Encontrar todos os clientes que tenham um empréstimo no banco mas não tenham uma conta neste banco.

```
select distinct nome_cliente
from devedor
where nome_cliente NOT IN (select nome_cliente
    from depositante)
```

Exemplo: Cursores

Pnome	Unome
Mauricio	Mau

Pnome	Minicial	Unome	Cpf	DataNasc	Endereço	Sexo	Salario	Cpf_supervisor	Dnr
Thiago	T	Rcthiago	00	12-02-83	SQSW	M	10k	02	01
Flavia	F	Flavials	01	11-12-84	SQSW	F	10k	02	01
Mauricio	M	Mau	02	11-08-70	SQS	M	12k	03	01
Aroldo	A	Arold	03	11-05-65	SQN	M	15k	-	01
Karla	K	Karlinha	04	25-08-79	SQS	F	11k	05	02
Victor	V	Victor	05	29-09-78	SQS	M	13k	-	02

Fcpf	Nome_dep	Sexo	Data_Nascimento
00	Vinicius	M	10-02-2010
02	Daniel	M	20-02-2008
02	Mauricio	M	15-09-2011
03	Maria	F	20-08-1998

```
SELECT F.Pnome, F.Unome FROM
FUNCIONARIO AS F
WHERE EXISTS
(SELECT * FROM DEPENDENTE AS D
        WHERE F.CPF = D.CPF  AND
        F.SEXO = D.SEXO  AND
        F.PNOME = D.NOME_DEPENDETE)
```

Exemplo 2 - Cesgranrio

Funcionário

IDFUNCIONARIO	NOME	DEPENDENTES
1	Marta	3
2	Simone	2
3	Jose	4
4	Adriano	3
5	Laura	3
6	Paulo	2
7	Luiz	2
8	Everaldo	5
9	Jose	2
10	Sandra	2

Projeto

IDPROJETO	NOME	PRIORIDADE
1	COBIT	1
2	ITIL	1
3	CMMI	3
4	PMI	2
5	ISO	3
6	FSW	2

Alocado

PPROJETO	PFUNCIONARIO
1	1
1	2
1	4
1	5
2	3
2	7
2	9
2	10
3	2
3	3
3	5
3	7
3	10
4	1
4	2
4	3
4	5
4	6
4	8
4	9
5	3
5	7
5	8
5	9
5	10
6	1
6	2
6	3
6	5
6	8
6	9

Func(1)
IdProjeto = 4,6
Pprojeto = 1,4,6

∅

Func(4)
IdProjeto = 4,6
Pprojeto = 1

```
SELECT IDFUNCIONARIO, NOME FROM FUNCIONARIO
WHERE  NOT EXISTS ( (SELECT IDPROJETO
      FROM   PROJETO WHERE  PRIORIDADE = 2 AND
IDPROJETO NOT IN
        (SELECT PPROJETO FROM ALOCADO
                WHERE  PFUNCIONARIO =
IDFUNCIONARIO
        )));
```

Idprojeto = 4,6

Divisão (3 opções)

1) Not exists combinado com o not in

```
select distinct matr, namn
from student as R
where kon = 'K' and
      not exists (select kurskod
                  from kursanmalan
                  where matr = 40101 and
                        kurskod not in (select kurskod
                                        from kursanmalan as R2
                                        where R.matr = R2.matr));
```

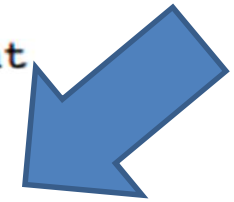
2) Usar duas vezes o not exists

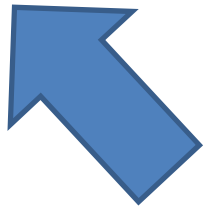
```
select distinct matr, namn
from student as R
where kon = 'K' and
      not exists (select kurskod
                  from kursanmalan as S
                  where matr = 40101 and
                        not exists (select kurskod
                                    from kursanmalan as R2
                                    where R.matr = R2.matr and
                                          S.kurskod = R2.kurskod ));
```

Divisão (3 opções)

3) “Contar e comparar”

```
select matr, namn
from kursanmalan natural join student
where kon = 'K' and
      kurskod in (select kurskod
                  from kursanmalan
                  where matr = 40101)
group by matr
having count(kurskod) = (select count(*)
                         from (select kurskod
                               from kursanmalan
                               where matr = 40101) as tab);
```





Conjuntos explícitos

- Conjuntos enumerados:
  - IN (1,2,3)

```
select distinct   nome_cliente
from   devedor
where   nome_cliente  not in  ('Smith', 'Jones')
```

Verificação de Relações Vazias

- O construtor exists retorna o valor true se a subconsulta usada como argumento é “não-vazia”.
- exists r $\Leftrightarrow$ r $\neq \emptyset$
- not exists r $\Leftrightarrow$ r = $\emptyset$

Exemplo (Construtor exists)

- Encontre todos os clientes que tenham uma conta em todas as agências localizadas no Brooklin

```
select distinct S.nome_cliente
from depositante as S
where not exists (
(select nome_agencia
 from agencia
 where cidade_agencia = 'Brooklyn')
except
 (select R.nome_agencia
  from depositante as T, conta as R
  where T.numero_conta = R.numero_conta and
  S.numero_cliente =T.numero_cliente))
```

- Note que $X - Y = \emptyset \Leftrightarrow X \subseteq Y$

Teste para Ausência de Tuplas Repetidas

- O contrutor unique testa se a sub-consulta tem alguma tupla repetida no seu resultado. Encontre todos os clientes que tenham apenas uma conta na agência Sudoeste.

```
select T.nome_cliente
from depositante as T
where UNIQUE (
        select R.nome_cliente
        from conta, depositante as R
        where T.nome_cliente = R.nome_cliente and
        R.numero_conta = conta.numero_conta and
        conta.nome_agencia = ' Sudoeste ')
```

Exemplo (Cláusula unique)

- Encontre todos os clientes que tenham pelo menos duas contas na agência Asa Sul.

```
select distinct T.nome_cliente
from depositante T
where NOT UNIQUE (
    select R.nome_cliente
    from conta, depositante as R
    where T.nome_cliente = R.nome_cliente and
R.numero_conta = conta.numero_conta and
conta.nome_agencia = ' Asa Sul ')
```

Funções Agregadas

- Essas funções operam nos multi-conjuntos de valores de uma coluna de uma relação e retornam um valor:
  - COUNT(), SUM(), MAX(), MIN(), AVG()
- GROUP BY
  - atributos na cláusula **select** fora das funções agregadas devem aparecer na lista **group by**.
- HAVING
  - predicados na cláusula **having** são aplicados após a formação dos grupos.

O uso do Group By

- ROLLUP

```
SELECT CATEGORY, PRICE, SUM(ON_HAND) AS TOTAL_ON_HAND
FROM COMPACT_DISC_STOCK
GROUP BY ROLLUP (CATEGORY, PRICE);
```

CATEGORY	PRICE	TOTAL_ON_HAND
Instrumental	14.99	5
Instrumental	15.99	23
Instrumental	16.99	50
Instrumental	NULL	78
Vocal	14.99	99
Vocal	15.99	73
Vocal	16.99	45
Vocal	NULL	217
NULL	NULL	295

O uso do Group By

- CUBE

```
SELECT CATEGORY, PRICE, SUM(ON_HAND) AS TOTAL_ON_HAND
FROM COMPACT_DISC_STOCK
GROUP BY CUBE (CATEGORY, PRICE);
```

CATEGORY	PRICE	TOTAL_ON_HAND
Instrumental	14.99	5
Instrumental	15.99	23
Instrumental	16.99	50
Instrumental	NULL	78
Vocal	14.99	99
Vocal	15.99	73
Vocal	16.99	45
Vocal	NULL	217
NULL	NULL	295
NULL	14.99	104
NULL	15.99	96
NULL	16.99	95

Funções Agregadas

- Encontre a média dos saldos em contas na agência Perryridge.

```
SELECT AVG (saldo)
FROM contas
WHERE nome_agencia = 'Perryridge‘;
```

- Encontre o número de tuplas na relação clientes:

```
SELECT COUNT (*)
FROM cliente;
```

- Encontre o número de depositantes no banco:

```
SELECT COUNT (DISTINCT nome_cliente)
FROM depositante;
```

Funções Agregadas

- Encontre o nome de todas as agências onde a média do saldo das contas seja maior que R$ 5.000,00

```
SELECT nome_agencia, AVG (saldo)
FROM conta
GROUP BY nome_agencia
HAVING AVG (saldo) > 5000;
```

Consultas Aninhadas Correlacionadas

- Encontre a média do balanço de contas das agências onde a média do balanço de contas é maior que $1200.

```
select nome_agencia, saldo_médio
from (select nome_agencia, avg (saldo)
  from conta
   group by nome_agencia)
     as result (nome_agencia,saldo_médio)
where saldo_médio > 1200
```

- Note que não é necessário usar a cláusula having, já que calculamos na cláusula from uma relação temporária e os atributos dessa relação podem ser usados diretamente no cláusula where.

Funções Matemáticas

SQL FUNCTION	DESCRIPTION	RDBMS SUPPORT
ABS	Returns the absolute value of a numeric input argument.	All
POWER	Returns the argument X raised to the power Y.	Microsoft Access uses '^' operator, and SQL Server has both the function and the operator
SQRT	Returns the square root of the argument X.	Microsoft Access uses function SQR
RAND	Generates some random numbers between 0 and 1.	Microsoft Access uses function RND
FLOOR	Rounds numeric arguments *down* to the nearest integer value.	All
CEIL	Rounds numeric arguments *up* to the nearest integer value.	All; some support also CEILING synonym
ROUND	Returns the numeric argument rounded to the integer number of decimal places.	All

Mais funções sobre caracteres

- Concatenação: ||
- SUBSTRING <left paren> <character value expression> FROM <start position>
- OVERLAY <left paren> <character value expression> PLACING <character value expression>
- { UPPER | LOWER } <left paren> <character value expression> <right paren>
- TRIM(str)

DML (PARTE 2)

Operações de Conjuntos

- As operações de conjuntos union, intersect e except operam em relações e correspondem às operações ∪ , ∩ e - (diferença) da álgebra relacional
  - Estas operações eliminam as duplicatas.
- Se desejarmos obter as repetições, devemos explicitar através da forma union all, intersect all e except all.
- Suponha uma tupla que ocorra “m” vezes em r e “n” vezes em s, então temos :
  - m + n vezes em r union all s
  - min (m, n) vezes em r intersect all s
  - max (0, m-n) vezes em r except all s

Operações de Conjuntos

- Encontre todos os clientes que possuam um empréstimo, uma conta ou ambos:
  - (**SELECT** nome_cliente **FROM** depositante) **UNION** (select nome_cliente **FROM** devedor)
- Encontre todos os clientes que possuem ambos uma conta e um empréstimo:
  - (**SELECT** nome_cliente **FROM** depositante) **INTERSECT** (**SELECT** nome_cliente **FROM** devedor)
- Encontre todos os clientes que possuem uma conta mas não possuem empréstimo;
  - (**SELECT** nome_cliente **FROM** depositante) **EXCEPT** (**SELECT** nome_cliente **FROM** devedor)

Composição de Relações (JOIN)

- Operações de Junção tomam duas relações e retornam como resultado uma outra relação.
- As operações de Junção são normalmente usadas na cláusula from.
- Condições de Junção – define quais tuplas das duas relações apresentam correspondência, e quais atributos serão apresentados no resultado de uma junção.
- Tipos de Junção – define como as tuplas em cada relação que não possuam nenhuma correspondência (baseado na condição de junção) com as tuplas da outra relação devem ser tratadas.

Tipos e condições de junção

Tipos de junção
inner join
left outer join
rigth outer join
full outer join

Condições de junção
natural
on <predicate>
using (A_1, A_2,..., A_n)

DML – Select e Joins

Marcas	
marca	nome
VW	Volkswagem
GM	General Motors
Ford	Ford

Carros			
modelo	ano	cor	marca
Fox	2010	amarelo	VW
Ecosport	2007	vermelho	Ford
Uno	2003	branco	Fiat
Stilo	2009	preto	Fiat
KA	2005	prata	Ford

INNER JOIN

- select m.nome, c.modelo from marcas as m inner join carros as c

Marcas	
marca	nome
VW	Volkswagem
GM	General Motors
Ford	Ford

Carros			
modelo	ano	cor	marca
Fox	2010	amarelo	VW
Ecosport	2007	vermelho	Ford
Uno	2003	branco	Fiat
Stilo	2009	preto	Fiat
KA	2005	prata	Ford

nome	modelo
Volkswagem	Fox
Ford	Ecosport
Ford	KA

LEFT OUTER JOIN

- select m.nome, c.modelo from marcas as m left join carros as c

Marcas	
marca	nome
VW	Volkswagem
GM	General Motors
Ford	Ford

Carros			
modelo	ano	cor	marca
Fox	2010	amarelo	VW
Ecosport	2007	vermelho	Ford
Uno	2003	branco	Fiat
Stilo	2009	preto	Fiat
KA	2005	prata	Ford

nome	modelo
Volkswagem	Fox
Ford	Ecosport
Ford	KA
General Motors	NULL

RIGHT OUTER JOIN

- select m.nome, c.modelo from marcas as m right join carros as c

Marcas	
marca	nome
VW	Volkswagem
GM	General Motors
Ford	Ford

Carros			
modelo	ano	cor	marca
Fox	2010	amarelo	VW
Ecosport	2007	vermelho	Ford
Uno	2003	branco	Fiat
Stilo	2009	preto	Fiat
KA	2005	prata	Ford

nome	modelo
Volkswagem	Fox
Ford	Ecosport
NULL	Uno
NULL	Stilo
Ford	KA

FULL OUTER JOIN

- select m.nome, c.modelo from marcas as m full outer join carros as c

Marcas	
marca	nome
VW	Volkswagem
GM	General Motors
Ford	Ford

Carros			
modelo	ano	cor	marca
Fox	2010	amarelo	VW
Ecosport	2007	vermelho	Ford
Uno	2003	branco	Fiat
Stilo	2009	preto	Fiat
KA	2005	prata	Ford

nome	modelo
Volkswagem	Fox
Ford	Ecosport
NULL	Uno
NULL	Stilo
Ford	KA
General Motors	NULL

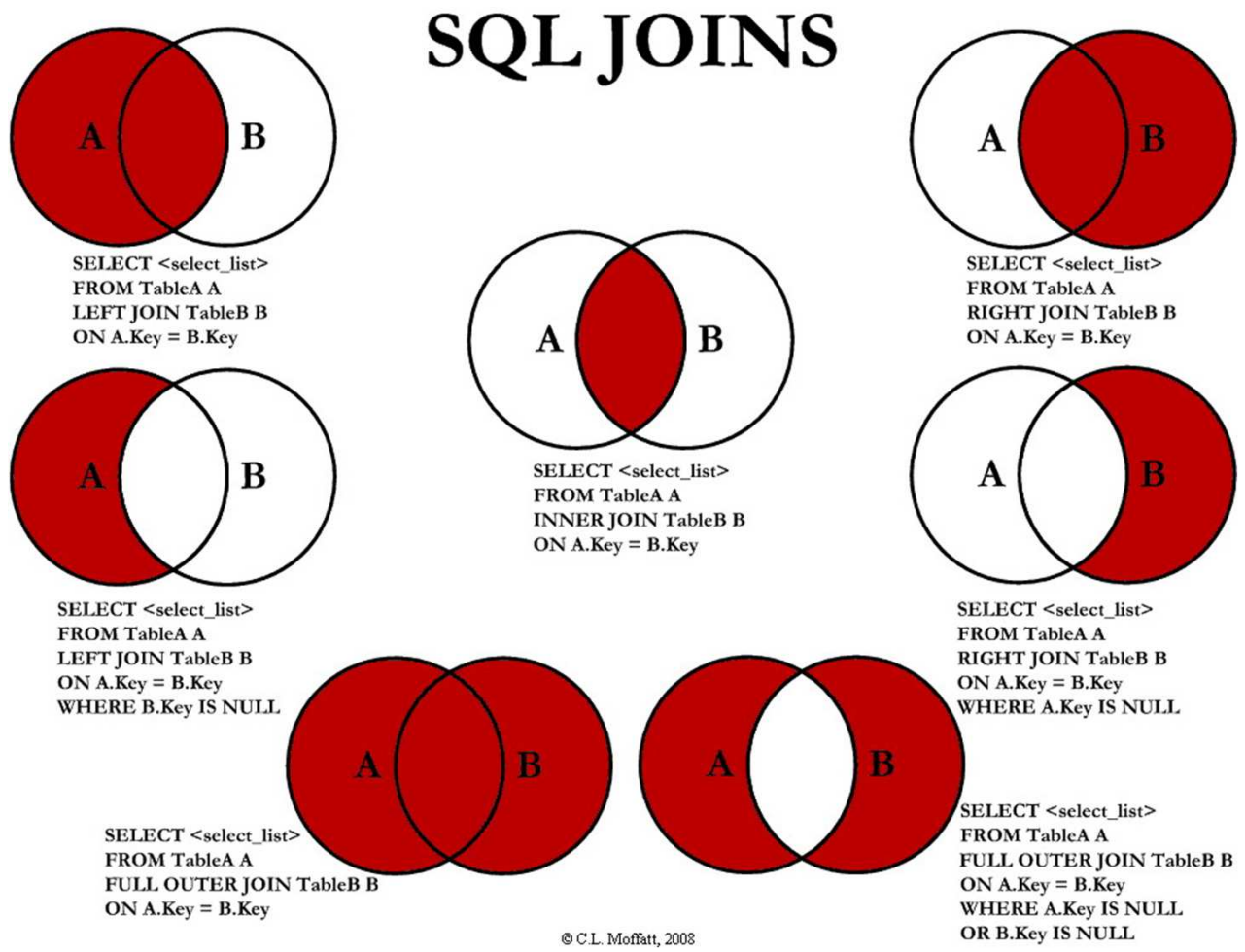
Cross join (Produto Cartesiano)

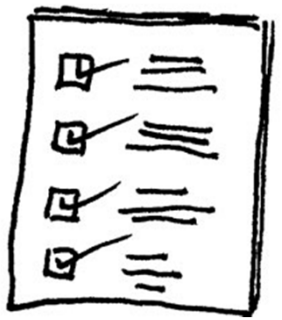
- select m.nome, c.modelo from marcas as m **cross join** carros as c

nome	modelo
Volkswagem	Fox
Volkswagem	Ecosport
Volkswagem	Uno
Volkswagem	Stilo
Volkswagem	KA
General Motors	Fox
General Motors	Ecosport
General Motors	Uno
General Motors	Stilo
General Motors	KA

nome	modelo
Ford	Fox
Ford	Ecosport
Ford	Uno
Ford	Stilo
Ford	KA

Resumindo





ITnerante

Resumo das consultas SQL

SELECT <ATRIBUTOS E LISTA DE FUNCOES>

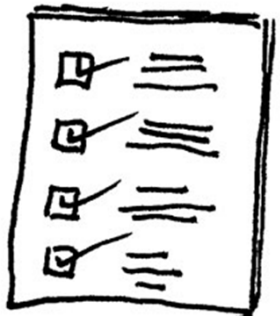
FROM <LISTA DE TABELAS>

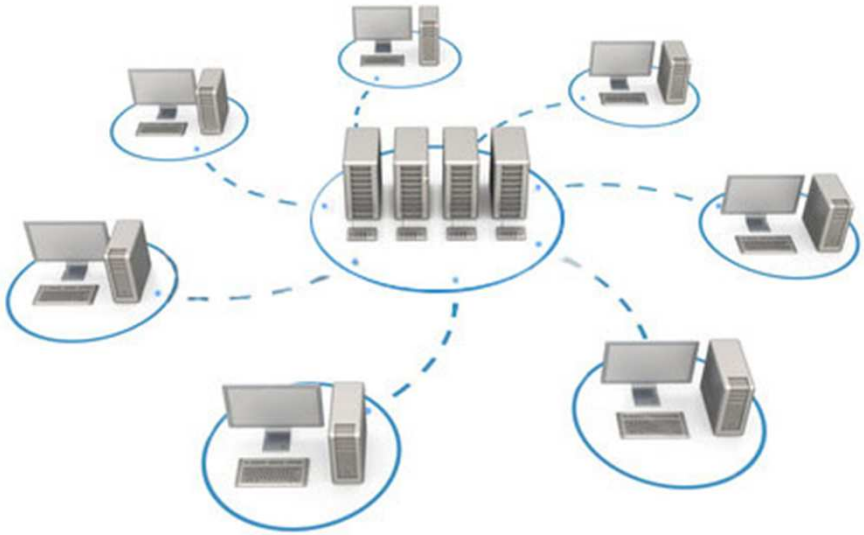
[WHERE <CONDICAO>**]**

[GROUP BY <ATRIBUTO(S) AGRUPADO(S)>**]**

[HAVING <CONDICAO DE AGRUPAMENTO>**]**

[ORDER BY <LISTA DE ATRIBUTOS>**];**





DML (PARTE 3)

COMANDO INSERT

- Adicionar uma nova tupla em conta

```
insert into conta
values ('Perryridge', A-9732, 1200);
```

- ou de forma equivalente

```
insert into conta (nome_agencia, saldo, numero_conta)
        values ('Perryridge', 1200, A-9732);
```

- Adicionar uma nova tupla à conta com saldo igual a nulo

```
insert into account
values ('Perryridge', A-777, null)
```

COMANDO DELETE e TRUNCATE

- DELETE
  - Deleta linha(s) de uma tabela
  - DELETE FROM <target table> [[AS] <correlation name>] [WHERE <search condition>]
  - Exclua todos os registros de contas da agência Ceilandia

```
delete from conta
where nome_agencia = 'Ceilandia‘;
```

- TRUNCATE
  - Deleta todas as linhas de uma tabela sem gerar nenhuma ação de gatilho (triggered action)
- TRUNCATE TABLE <target table> [CONTINUE IDENTITY | RESTART IDENTITY]

Questão 07 - Prova: CESPE - 2013 - CNJ - Técnico Judiciário - Programação de Sistemas

No que se refere ao conceito de banco de dados relacional, julgue os itens seguintes.

[95] Na linguagem de consulta estruturada (SQL), é correto utilizar o comando TRUNCATE TABLE, com a finalidade de excluir todos os dados de uma tabela.

[96] O comando EXTRACT na linguagem SQL é utilizado para extrair dados de uma tabela.

COMANDO UPDATE

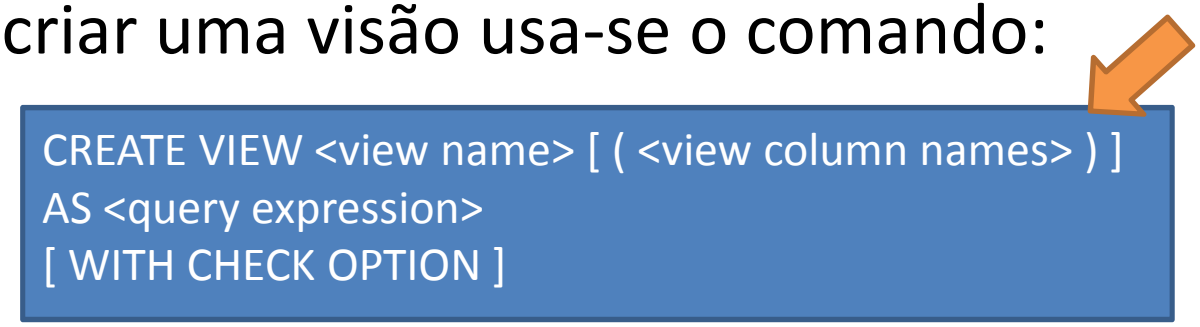
- Acrescentar 6% a todas as contas com saldos acima de $10.000. Todas as outras contas recebem 5%. Escreva dois comandos update:

```
update conta
set saldo = saldo *  1.06
where saldo > 10000
update conta
set saldo = saldo * 1.05
where saldo =< 10000
```

- Aqui, a ordem dos comandos é importante!
- SQL-92 fornece um construtor **CASE** que pode ser usado para realizar ambas as alterações anteriores em um único comando update

Visões

- Fornecem um mecanismo para:
  - Esconder certos dados do alcance de certos usuários.
  - Especificar consultas complexas apenas uma vez
  - Conhecidas como tabelas virtuais cuja definição existe como um objeto do schema
  - Para criar uma visão usa-se o comando:



Exemplos de View

```
CREATE VIEW COMPACT_DISCS_IN_STOCK
( COMPACT_DISC, COPYRIGHT, IN_STOCK ) AS
SELECT CD_TITLE, COPYRIGHT, IN_STOCK
FROM COMPACT_DISC_INVENTORY;
```

```
CREATE VIEW CDS_IN_STOCK_1990S
( COMPACT_DISC, COPYRIGHT, IN_STOCK ) AS
SELECT CD_TITLE, COPYRIGHT, IN_STOCK
FROM COMPACT_DISC_INVENTORY
WHERE COPYRIGHT > 1989 AND COPYRIGHT < 2000;
```

```
CREATE VIEW COMPACT_DISC_PUBLISHERS
( COMPACT_DISC, PUBLISHER ) AS
SELECT CD_INVENTORY.CD_TITLE, LABELS.COMPANY_NAME
FROM CD_INVENTORY, LABELS
WHERE CD_INVENTORY.LABEL_ID = LABELS.LABEL_ID;
```

WITH CHECK OPTION

```
CREATE VIEW EMP_COMM AS
SELECT EMPLOYEE_ID, YEAR_1999, YEAR_2000
FROM EMPLOYEE_COMMISSIONS
WHERE YEAR_1999 > 100;
```

OK

```
CREATE VIEW EMP_COMM AS
SELECT EMPLOYEE_ID, YEAR_1999, YEAR_2000
FROM EMPLOYEE_COMMISSIONS
WHERE YEAR_1999 > 100
WITH CHECK OPTION;
```

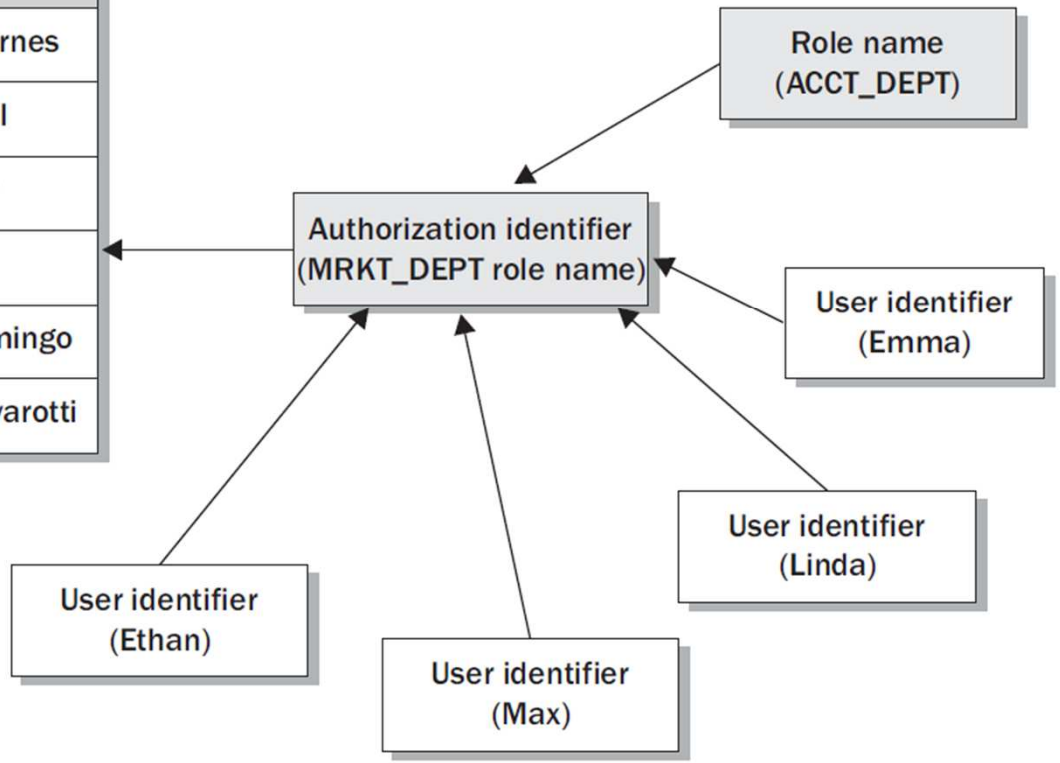
Pergunta relevante!

- **Nos exemplos todas as visões fazem referência a tabelas base. Todas as visões são criadas somente a partir de tabelas base?**
  - Não, **visões podem ser criadas usando expressões de consulta para extrair dados de outras visões**.
  - Além disso, é possível criar uma visão que contém apenas dados calculados e, que portanto, não tem dados armazenados em uma tabela base.

Modelo de segurança de SQL

PERFORMERS

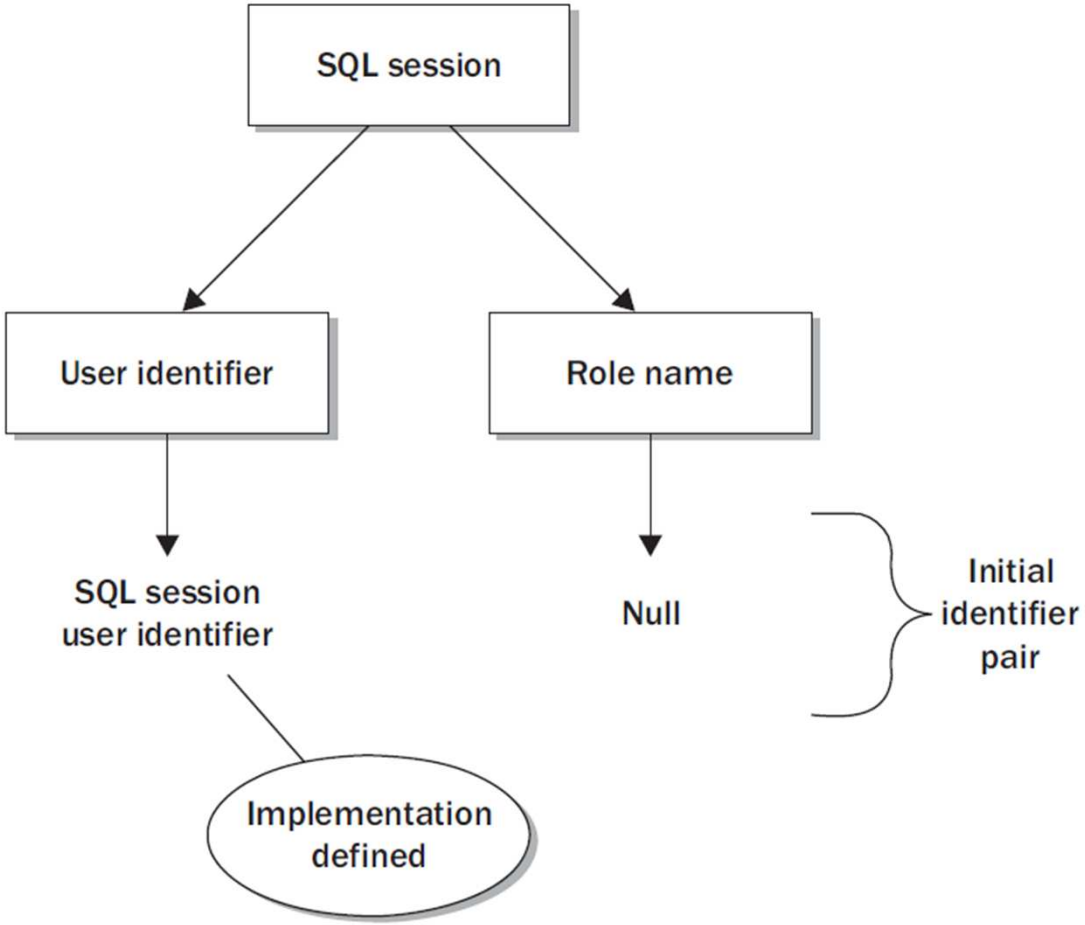
PERFORMER_ID: INT	FULL_NAME: VARCHAR(60)
10001	Jennifer Warnes
10002	Joni Mitchell
10005	Bing Crosby
10006	Patsy Cline
10008	Placido Domingo
10009	Luciano Pavarotti



PUBLIC

ITnerante

Modelo de segurança de SQL



Schema objects

- Base tables
- Views
- Columns
- Domains
- Character sets
- Collations
- Translations
- User-defined types
- Sequences
- Triggers
- SQL-invoked routines

Criando Roles

```
CREATE ROLE <role name>
[ WITH ADMIN { CURRENT_USER | CURRENT_ROLE } ]

Ex:
CREATE ROLE CUSTOMERS;
```

```
DROP ROLE <role name>

Ex:
DROP ROLE CUSTOMERS;
```

Grant e Revoke

- O comando SQL GRANT é usado para prover acesso ou privilégios para os usuários no banco de dados.
- O comando REVOKE serve para remover permições e privilegios de acesso a base de dados dos usuários.

Grant sintaxe

GRANT { ALL PRIVILEGES | <privilege list> }
ON <object type> <object name>
TO { PUBLIC | <authorization identifier list> } [WITH GRANT OPTION]
[GRANTED BY { CURRENT_USER | CURRENT_ROLE }]

GRANT <role name list>
TO { PUBLIC | <authorization identifier list> } [WITH ADMIN OPTION]
[GRANTED BY { CURRENT_USER | CURRENT_ROLE }]

Grant

- Cenário DBA, A1, A2, A3, A4
- **GRANT** CRAETETAB **TO** A1;
- **CREATE SCHEMA** CGU **AUTHORIZATION** A1;
- A1: crias as tabelas Empregado e Departame.
- A1:**GRANT** INSERT, DELETE **ON** EMP, DEP **TO** A2
- A1:**GRANT** SELECT **ON** EMP, DEP **TO** A3 **WITH GRANT OPTION**

Grant

- O privilegio UPDATE ou INSERT pode especificar atributos em particular que podem ser atualizados ou incluídos em uma relação
  - GRANT UPDATE ON EMP (SALARIO) TO A4;
- Outros privilégios (SELECT, DELETE) não especificam atributos
  - Controlados por meio de criação de visões

Revoke

- A sintaxe do comando revoke:

```
REVOKE [ GRANT OPTION FOR ] { ALL PRIVILEGES | <privilege list> }
ON <object type> <object name>
FROM { PUBLIC | <authorization identifier list>
[ GRANTED BY { CURRENT_USER | CURRENT_ROLE } ]
{ RESTRICT | CASCADE }
```

```
REVOKE [ ADMIN OPTION FOR ] <role name list>
FROM { PUBLIC | <authorization identifier list> }
[ GRANTED BY { CURRENT_USER | CURRENT_ROLE } ]
{ RESTRICT | CASCADE }
```

Slide 148

T4 Colocar alguma coisa aqui

Thiago; 08/05/2012

Banco de dados ativo

- Modelo ECA (evento-condição-ação)
  - Evento - ativa a regra: atualização de BD aplicadas explicitamente, eventos temporais, eventos externos.
  - A condição - determina se a ação deve ser executada: opcional >> se nada for declarado executa sempre!
  - A ação executada: sucessão de declarações SQL, mas também poderia ser uma transação de BD ou um programa externo que será executado automaticamente.

Triggers

- Ações que são automaticamente disparadas quando um evento acontece
- INSERT, DELETE e UPDATE
- São permitidos gatilhos em nível de linha (FOR EACH ROW) e em nível de declaração (FOR EACH STATEMENT)
- É possível especificar nomes de variáveis para a tupla nova e antiga dentro da cláusula REFERENCING

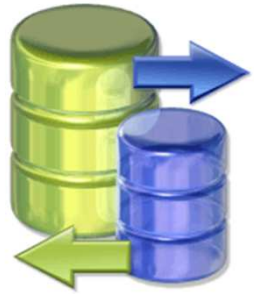
Triggers (Exemplo 01)


```
CREATE TRIGGER TOTALSAL 1
AFTER UPDATE OF SALARIO ON EMPREGADO
REFERENCING OLD ROW AS O, NEW ROW AS N
FOR EACH ROW
WHEN (N.DNO IS NOT NULL)
        UPDATE DEPARTAMENTO
        SET TOTAL_SAL = TOTAL_SAL + N.SALARY –
O.SALARY
        WHERE DNO = N.DNO;
```

Triggers (Exemplo 02)

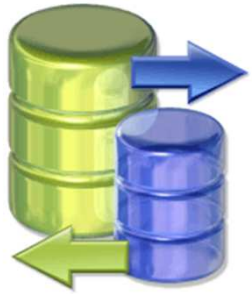

```
CREATE TRIGGER TOTALSAL2
AFTER UPDATE OF SALARIO ON EMPREGADO
REFERENCING OLD ROW AS O, NEW ROW AS N
FOR EACH STATEMENT
WHEN EXISTS (SELECT * FROM N WHERE N.DNO IS NOT
NULL) OR EXISTS (SELECT * FROM O WHERE O.DNO IS NOT
NULL)
        UPDATE DEPARTAMENTO AS D
        SET D.TOTAL_SAL = D.TOTAL_SAL
        + (SELECT SUM(N.SALARIO) FROM N WHERE
D.DNO=N.DNO)
        -  (SELECT SUM(O.SALARIO) FROM O WHERE
D.DNO=O.DNO)
        WHERE DNO IN ((SELECT DNO FROM N) UNION
(SELECT DNO FROM O));
```

SUPORTE A TRANSAÇÃO EM SQL



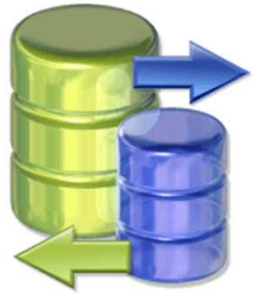
- Transação
  - Uma unidade lógica de trabalho
  - Garantida como atômica
- Em SQL não há BEGIN TRANSACTION explicito
- ROLLBACK e COMMIT são explícitos
- Características específicas:
  - Modo de acesso
  - Tamanho da área de diagnóstico
  - Nível de isolamento

Classificações



- Modo de acesso
  - READ ONLY
  - READ WRITE
- Tamanho da área de diagnostico
  - DIAGNOSTIC SIZE n
- Nível de isolamento (ISOLATION LEVEL <I>)
  - READ UNCOMMITED (leitura suja)
  - READ COMMITED (leitura não-repetível)
  - REPEATABLE READ (fantasmas)
  - SERIALAZABLE

Exemplo



```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSATION
        READ WRITE
        DIAGNOSTIC SIZE 5
        ISOLATION LEVEL SERIALAZABLE
EXEC SQL INSER INTO EMPREGADO (PNOME, UNOME, SSN,
DNO, SALARIO) VALUES (´Thiago´, ´Cavalcanti´, 000457878, 2,
12000);
EXEC SQL UPDATE EMPREGADO
        SET SALARIO = SALARIO *1,1 WHERE DNO =2;
EXEC COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: ...;
```

Comandos SQL Embutidos

- O padrão SQL define que a SQL será embutida em uma variedade de linguagens de programação, como Pascal, PL/I, Fortran, C, e Cobol.
- A linguagem na qual são embutidas consultas SQL é chamada linguagem hospedeira, e as estruturas SQL permitidas na linguagem hospedeira são denominadas SQL embutida.
- EXEC SQL é usado para identificar os pedidos em SQL embutida para o pré-processador EXEC SQL <comando SQL embutido > END EXEC

Exemplo - SQL Embutido

- Suponha que temos na linguagem hospedeira uma variável chamada total e que desejamos encontrar os nomes e cidades dos clientes dos clientes que tenham mais de um total em dólares em qualquer conta.
- Podemos escrever essa consulta como segue:

```
EXEC SQL
  declare c cursor for
  select nome_cliente,cidade_cliente
     from depósito,cliente
     where depósito.nome_cliente = cliente.nome_cliente
                  and depósito.saldo > :total
END-EXEC
```

Exemplo - SQL Embutido

- O comando **open** faz com que a consulta seja avaliada

  EXEC SQL open c **END-EXEC**

- O comando **fetch** determina os valores de uma tupla que serão colocados em variáveis da linguagem host.

  EXEC SQL fetch c **into** :cn :an **END-EXEC**

  Pode-se utilizar um laço para processar cada tupla do resultado; uma variável na área de comunicação do SQL indica que não há mais tupla a ser processada.

- O comando close faz com que o sistema de banco de dados remova a relação temporária mantida para o resultado da consulta.

  EXEC SQL close c **END-EXEC**

BONUS TRACK

PL/SQL – Uma rápida introdução

Prof. Thiago Cavalcanti

PL/SQL

- Linguagem procedural
- Blocos de instrução com condicionais e loops
- Extensão da linguagem SQL para o banco de dados Oracle
- Combina o poder e a flexibilidade de SQL com as estruturas de código de procedimentos encontradas nas linguagens de programação de 3a. Geração
  - Estruturas de procedimento como
    - Variáveis e tipos (pré-definidos ou não)
    - Estruturas de controle (IF-THEN-ELSE e laços)
    - Procedimentos e funções
    - Tipos de objeto e métodos (Versão 8 em diante)

Elementos básicos do PL/SQL

- Variáveis e tipos
  - Utilizadas para transmitir informação entre programa PL/SQL e a base de dados
  - Localização de memória que pode ser lida ou ter valor armazenado a partir do programa PL/SQL
  - Não inicializadas recebem por default o valor NULL

Elementos básicos do PL/SQL

- Operadores
  - Atribuição :=
  - Diferente de < > ou ~=
  - Referência à base de dados @
- Comentários

```
-- indica comentário de uma linha
```

```
/* indica comentário de mais de
uma linha */
```

- Data do sistema
  - data_provaTI := SYSDATE;

Elementos básicos do PL/SQL

- Declaração de constante
  - desconto_padrao **CONSTANT** NUMBER(3,2) := 8.25;
- Declaração de valor default
  - participante BOOLEAN **DEFAULT** TRUE;
- Declaração de variável com tipo de um atributo de tabela
  - <variavel > carro.modelo**%TYPE**;

Elementos básicos do PL/SQL

- Registro
  - Declaração e uso
    - TYPE **nome** IS RECORD (<campos>);
    - <variável> **nome**;
  - Para declarar registro com mesmos tipos que uma tabela da base de dados
    - <variável> cliente%ROWTYPE;

Estruturas de controle

- IF-THEN-ELSE

```
IF <expressão booleana 1> THEN <instruções-1>
[ELSEIF <expressão booleana 2> THEN
<instruções-2>]
[ELSE <instruções-3>]
END IF;
```

- CASE

```
CASE <seletor>
WHEN <valor1> THEN <instruções 1>;
...
WHEN <valorn> THEN <instruções n>;
[ELSE <instruções m>;]
END CASE;
```

- Laços: Simples, WHILE, FOR, GOTO (Rótulos)

```
LOOP
<instruções>
END LOOP;
```

```
WHILE <condição> LOOP
<instruções>
END LOOP;
```

```
FOR <contador>IN [REVERSE]
<inferior>.. <superior>
LOOP
<instruções>
END LOOP;
```

Recuperação de Dados para Variável

- Utilizar comando → SELECT ... INTO
  - SELECT <atributo(s)> INTO <variável(is)> FROM <tabela(s)> WHERE... ;
- Considere
  - Número de <variável(is)> deve ser igual ao número de <atributo(s)>
  - Os tipos de cada <atributo> e da <variável> correspondente devem ser compatíveis
  - Deve ser recuperada uma única tupla

Blocos PL/SQL

- Várias instruções de SQL podem estar contidas em um único **bloco** de PL/SQL e enviadas como uma só unidade para o servidor
- Estrutura de blocos
  - Unidade básica
  - Todos os programas são construídos por blocos que podem ser encadeados entre si
  - Cada bloco executa uma unidade lógica de trabalho

Estrutura de blocos

DECLARE

/* Seção para declarar variáveis, tipos, cursores e subprogramas locais */

BEGIN

/*Seção executável - comandos procedurais e SQL.

É a única obrigatória */

EXCEPTION

/*Comandos Manipulação de erros*/

END;

```
DECLARE
  v_chassi VARCHAR(20) := '235-456-YWR';
-- Variável alfanumérica    inicializada
--com 235-456-YWR
  v_modelo VARCHAR2(20);
-- Tamanho de variável string com no
-- máximo  20 caracteres
```

```
BEGIN
  /* Início da Execução */
  -- recupera modelo do carro com chassi
  -- 235-456-YWR
  SELECT modelo
    INTO v_modelo
    FROM carro
    WHERE chassi = v_chassi;
```

```
EXCEPTION
  /* Seção de Exceção */
  WHEN NO_DATA_FOUND THEN
    -- Manipula a condição de erro
    INSERT INTO log_table (info)
      VALUES ('Carro com Chassi 235-456-
YWR não existe!');
END;
```

Sobre blocos em PL/SQL

- Blocos anônimos
  - Construídos de forma dinâmica e executados só uma vez
- Blocos nomeados
  - Blocos anônimos com um rótulo que fornece o nome do bloco
- Tratamento de Exceções
  - Para cada tipo de erro pode-se colocar um WHEN na seção EXCEPTION
  - A opção WHEN OTHERS pode ser usada para tratar qualquer erro diferente dos listados

Subprogramas: Procedures, Functions e Package

- Procedimentos, funções e pacotes que são armazenados na base de dados
- Em geral não são alterados depois de construídos e são executados muitas vezes
- São executados explicitamente, através de uma chamada a eles

Sintaxe de Procedure

```
CREATE [OR REPLACE] PROCEDURE <nome>
 [(parâmetro [{IN | OUT | IN OUT}] tipo, ....)]
{IS | AS}
<definições  de variáveis>
BEGIN <corpo-do-procedimento>
END <nome>;|
```

```
CREATE OR REPLACE PROCEDURE InsereCarro (
p_chassi   carro.chassi%TYPE,
  p_modelo    carro.modelo%TYPE,
  p_km_carro carro.km_carro%TYPE,
  p_data_carro carro.data_carro %TYPE) AS
BEGIN
  -- Inserir nova tupla na tabela carro
INSERT INTO carro (chassi, modelo,
km_carro, data_carro)
    VALUES (p_chassi, p_modelo, p_km_carro,
p_data_carro);
COMMIT;
END InsereCarro;
/
```

Sintaxe da function

- São chamadas como parte de uma expressão, retornando um valor à expressão
- No corpo da função deve aparecer um RETURN para retornar um valor ao ambiente

RETURN <expressão>;

```
CREATE [OR REPLACE ]FUNCTION <nome>
[(parâmetro [{IN | OUT | IN OUT}] tipo, ....)]
RETURN <tipo-retorno> {IS | AS}
BEGIN <corpo-do-procedimento>
END <nome>;
```

Sintaxe do Package

- Fornecem mecanismo para ampliar o poder da linguagem
- Os elementos do pacote podem aparecer em qualquer ordem, mas um objeto tem que ser declarado antes que seja referenciado
- Na definição do pacote só é apresentada a especificação do mesmo
- A implementação é apresentada à parte através do corpo do programa

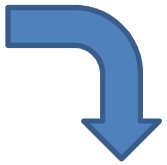
Parâmetros em PL/SQL

Especificação de parâmetros:
parameter_name [IN | OUT | IN OUT]
datatype [{:= | DEFAULT} expression]

Modos/Comportamentos
IN – internamente são como constantes, não podem ser alterados no sub-programa, passagem por referência
OUT – para retornar valores
IN OUT – permite a passagem de valores para o sub-programa e o retorno, passagem por valor

Exemplos

```
CREATE OR REPLACE PROCEDURE reajuste
(v_codigo_emp IN emp.empno%type,
v_porcentagem IN number)
IS BEGIN
    UPDATE emp SET sal = sal + (sal*(v_porcentagem/100))
    WHERE empno = v_codigo_emp;
COMMIT;
END reajuste;
```



```
select sal from emp where empno=7369;
execute reajuste (7369,10);
select sal from emp where empno=7369
```

```
CREATE OR REPLACE PROCEDURE consulta_emp
(p_id IN emp.empno%type,
p_nome OUT emp.ename%type,
p_salario OUT emp.sal%type)
IS BEGIN
    SELECT ename, sal INTO p_nome, p_salario
    FROM emp WHERE empno = p_id;
END consulta_emp;
```

```
CREATE OR REPLACE PROCEDURE formata_fone
(p_fone IN OUT varchar2)
IS BEGIN
    p_fone := ´(´ || substr(p_fone, 1, 3) ||`)`
                  || substr(p_fone, 4, 3)  ||
              `-` || substr(p_fone, 7);
END formata_fone;
```

Oracle/PLSQL: Grant/Revoke

Grant Privileges on Tables:

É possível garantir privilégios sobre tabelas usando o comando:

grant privileges on object to user;

Você pode garantir privilégios sobre select, insert, update, delete, references, alter, e index.

Vejamos um exemplo:

grant select, insert, update, delete on empregados to thiago;

Obs: all (grant all …) e public (grant … to public)

Revoke Privileges on Tables:

É possivel revogar privilegios de usuários por meio do comando:

revoke privileges on object from user;

É possível retirar qualquer um dos privilégios descritos acima.

Vejamos um exemplo:

revoke delete on empregado from flavinha;

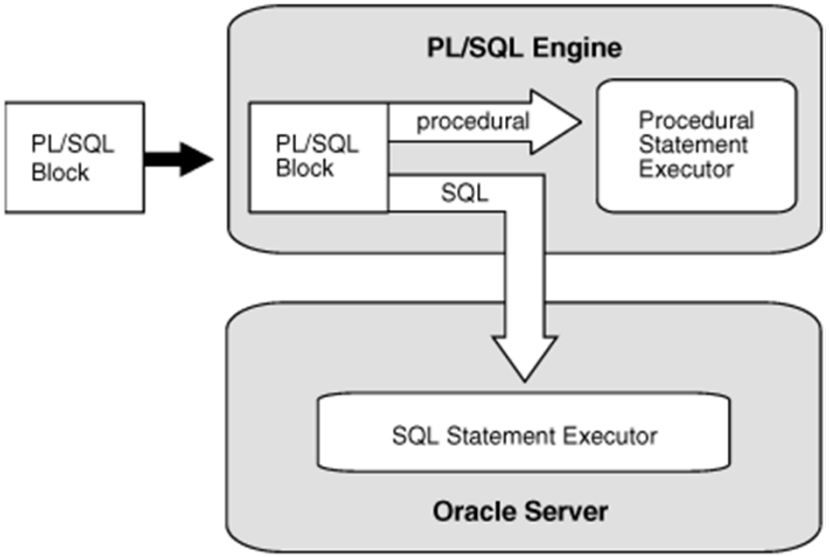
Obs: all (revoke all …) e public (revoke … from public)

Engine PL/SQL

- Os processos de banco de dados da Oracle PL/SQL Server podem ser divididos em: blocos e subprogramas.
- **Blocos anônimos (Anonymous Blocks)**
  - Blocos anônimos de PL / SQL podem ser submetidos a ferramentas interativas como o SQL * Plus e Enterprise Manager, ou incorporado em um pré-compilador Oracle ou programa OCI. Em tempo de execução, o programa envia estes blocos para o banco de dados Oracle, onde são compilados e executados.

- **Subprogramas Armazenados (Stored Subprograms)**
  - Subprogramas podem ser compilados e armazenados numa base de dados Oracle, pronto para ser executado. Uma vez compilados, é um objecto de esquema conhecido como uma função armazenada(stored function) ou procedimento armazenado (stored procedure), que podem ser referenciadas por diversas aplicações ligadas ao banco de dados.

- **Gatilho (Database Triggers)**
  - Um gatilho de banco de dados é um subprograma armazenado associado a uma tabela de banco de dados, visão ou evento. O gatilho pode ser chamado uma vez, quando algum evento ocorre, ou muitas vezes, uma para cada linha afetada por uma instrução INSERT, UPDATE ou DELETE.



ITnerante

ITnerante

TIMASTERS

CESPE UnB
UNIVERSIDADE DE BRASÍLIA

FUNDAÇÃO CESGRANRIO

Fundação Carlos Chagas

ESAF
Escola de Administração Fazendária

FGV

ITnerante

TIMASTERS

FUNDAÇÃO CESGRANRIO

Prof. Thiago Cavalcanti

Questão 08 - Prova: Cesgranrio - BNDES 2013 - Desenvolvimento

Considere as tabelas a seguir para responder às questões de nºs 38 e 39. Essas tabelas pertencem ao esquema de um banco de dados de uma locadora de veículos.

```
CREATE TABLE VEICULO (
   PLACA      CHAR(7)     NOT NULL,
   MODELO     VARCHAR2(50) NOT NULL,
   COD_CAT    CHAR(2)     NOT NULL,
   CONSTRAINT  VEICULO_PK PRIMARY KEY (PLACA),
   CONSTRAINT  VEICULO_FK FOREIGN KEY (COD_CAT) REFERENCES
     CATEGORIA (COD_CAT))

CREATE TABLE CATEGORIA (
   COD_CAT    CHAR(2)     NOT NULL,
   DESCR      VARCHAR2(80) NOT NULL,
   VAL_DIARIA  NUMBER(7,2),
   CONSTRAINT  CATEGORIA_PK PRIMARY KEY (COD_CAT))

CREATE TABLE ALUGUEL (
   PLACA      CHAR(7)  NOT NULL,
   DATA_DEV    NUMBER(6),
   DATA_ALG    NUMBER(6)  NOT NULL,
   CONSTRAINT  ALUGUEL_PK PRIMARY KEY (PLACA, DATA_ALG),
   CONSTRAINT  ALUGUEL_FK FOREIGN KEY (PLACA) REFERENCES
     VEICULO (PLACA))
```

Observações:

- A tabela VEICULO contém as informações sobre os veículos que a locadora dispõe para aluguel. Ela possui uma coluna chamada COD_CAT, que contém a categoria à qual um veículo pertence.
- A tabela CATEGORIA representa a tabela de preços da locadora. Ela registra o valor que será cobrado por um dia de aluguel de um veículo de uma determinada categoria.
- A tabela ALUGUEL é usada para registrar todas as operações de aluguel. A coluna DATA_ALG guarda a data na qual um veículo foi alugado, enquanto a coluna DATA_DEV guarda a data na qual o veículo foi devolvido. Ela é informada ao sistema quando o cliente devolve o veículo à locadora. Ambas as datas estão no formato AAMMDD. Dessa forma, a data 05/02/2011 será armazenada como 110205.

Questão 08 - Prova: Cesgranrio - BNDES 2013 – Desenvolvimento – Q.38

Qual comando SQL será executado com sucesso, independente do estado das tabelas que compõem a base de dados da locadora de veículos?

(A) INSERT INTO CATEGORIA
(DESCR,VAL_DIARIA,COD_CAT)
VALUES ('sedan compacto',90.00,'uc')

(B) DELETE FROM CATEGORIA X WHERE NOT EXISTS
(SELECT COUNT(*) FROM VEICULO V,ALUGUEL A
WHERE V.COD_CAT=X.COD_CAT AND V.PLACA=A.PLACA
AND A.DATA_DEV IS NOT NULL
GROUP BY V.COD_CAT)

(C) INSERT INTO ALUGUEL VALUES ('LJJ2222',120618)

Questão 08 - Prova: Cesgranrio - BNDES 2013 – Desenvolvimento – Q.38

(D) DELETE FROM VEICULO X WHERE
NOT EXISTS (SELECT COUNT(*)
FROM ALUGUEL A
WHERE X.PLACA=A.PLACA AND
A.DATA_DEV IS NOT NULL
GROUP BY A.PLACA)
(E) UPDATE VEICULO X SET COD_CAT='xs' WHERE
EXISTS (SELECT COUNT(*)
FROM VEICULO V,ALUGUEL A
WHERE V.COD_CAT='xs' AND V.PLACA=A.PLACA AND
A.DATA_DEV IS NOT NULL
GROUP BY V.COD_CAT)

Questão 09 - Prova: Cesgranrio - BNDES 2013 – Desenvolvimento – Q.39

Qual consulta permite exibir a placa e o modelo dos veículos que **NÃO** foram alugados no mês de junho de 2012?

(A) SELECT PLACA,MODELO
FROM VEICULO V
WHERE (SELECT COUNT(*) FROM ALUGUEL
WHERE PLACA=V.PLACA AND DATA_ALG>=120601 AND
DATA_ALG<=120630)>0

(B) SELECT V.PLACA,V.MODELO
FROM VEICULO V MINUS
SELECT V.PLACA,V.MODELO
FROM VEICULO V,ALUGUEL A
WHERE V.PLACA=A.PLACA AND A.DATA_ALG>=120601 AND
A.DATA_ALG<=120630

(C) SELECT PLACA,MODELO
FROM VEICULO WHERE PLACA IN
(SELECT PLACA FROM ALUGUEL
WHERE DATA_ALG>=120601 AND DATA_ALG<=120630

Questão 09 - Prova: Cesgranrio - BNDES 2013 – Desenvolvimento – Q.39

Qual consulta permite exibir a placa e o modelo dos veículos que **NÃO** foram alugados no mês de junho de 2012?

(D) SELECT V.PLACA,V.MODELO
FROM VEICULO V,ALUGUEL A
WHERE V.PLACA=A.PLACA AND (A.DATA_ALG<120601 OR A.DATA_ALG>120630)

(E) SELECT V.PLACA,V.MODELO
FROM VEICULO V,ALUGUEL A
WHERE V.PLACA=A.PLACA AND A.DATA_ALG>=120601 AND A.DATA_ALG<=120630
MINUS
SELECT V.PLACA,V.MODELO
FROM VEICULO V

Questão 10 - Discursiva – BNDES 2013 - Desenvolvimento

Questão no 3 Seja o esquema relacional da base de dados de uma empresa. Para cada tabela, os atributos que compõem a chave primária estão sublinhados e os atributos que são chaves estrangeiras são indicados explicitamente, conforme abaixo.

departamento (id_depto, nome_depto)

projeto (id_proj, nome_proj, local_proj, id_depto)

id_depto ref departamento (id_depto)

empregado (matricula, nome, endereco, sexo, data_nasc, salario, id_depto)

id_depto ref departamento (id_depto)

especialização (id_esp, nome_esp, tipo_esp)

gerencia (id_depto, matricula, data)

id_depto ref departamento (id_depto)

matricula ref empregado (matricula)

trabalha (matricula, id_proj, mes, horas)

matricula ref empregado (matricula)

id_proj ref Projeto (id_proj)

possui (matricula, id_esp)

matricula ref empregado (matricula)

id_esp ref especialização (id_esp)

Elabore em SQL as seguintes consultas à base:

Resposta da letra A

a) Consulta 1 – Listar os nomes dos empregados e os nomes dos departamentos onde trabalham, que ganham mais do que o maior salário pago a um empregado do departamento de nome igual a 'informatica' **(valor: 3,0 pontos)**

- select e.nome, d.nome_depto from empregado e join departamento d on e.id_depto = d.id_depto where e.salario > (
- select max(e2.salario) from empregado e2 join departamento d2 on e2.id_depto = d2.id_depto where d2.nome_depto = 'informatica')

Resposta letra B

b) Consulta 2 – Listar, para cada projeto localizado no 'Rio de Janeiro', o identificador do projeto, o identificador do departamento que o controla e a soma das horas trabalhadas pelos empregados no projeto, no mês de janeiro.

- select p.id_proj, p.id_depto, sum(t.horas) as horas_trabalhadas from projeto p join trabalha t on t.id_proj = p.id_proj where p.local_proj = 'Rio de Janeiro' and t.mes = 1
- group by p.id_proj, p.id_depto

Resposta letra C

c) Consulta 3 – Listar os identificadores de todos os projetos que envolvam um empregado cujo nome é 'jose da silva'. O empregado pode envolver-se no projeto como trabalhador ou como gerente do departamento que controla o projeto. Não deve haver repetição na resposta.

- select t.id_proj from empregado e join trabalha t on e.matricula = t.matricula where e.nome = 'jose da silva'
- union
- select p.id_proj from empregado e join gerencia g on g.matricula = e.matricula join projeto p on p.id_depto = g.id_depto where e.nome = 'jose da silva'

Questão 11 - Prova: Cesgranrio - BNDES 2013 – Suporte – Q.67

O modelo relacional a seguir representa um banco de dados simplificado de uma empresa de comércio. As chaves estão sublinhadas.

CLIENTE(NomeC, EnderecoC)

PRODUTO(NomeP)

FORNECEDOR(NomeF)

PRODUZ(NomeF,NomeP,Preco)

PEDIDO(NomeC,NomeF,NomeP,Quantidade)

Se o dono da empresa deseja saber quais clientes nunca pediram um produto do fornecedor cujo nome é "Barateira", que consulta SQL deve fazer?

Questão 11 - Prova: Cesgranrio - BNDES 2013 – Suporte – Q.67

(A) **SELECT * FROM CLIENTE WHERE CLIENTE.NOMEC IN (SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")**

(B) **SELECT * FROM CLIENTE WHERE CLIENTE. NOMEC NOT IN (SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")**

(C) **SELECT * FROM CLIENTE WHERE CLIENTE.NOMEC= PEDIDO.NOMEC AND CLIENTE.NOMEC NOT IN (SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")**

(D) **SELECT * FROM CLIENTE,PEDIDO WHERE CLIENTE.NOMEC=PEDIDO.NOMEC AND CLIENTE.NOMEC IN (SELECT NOMEC FROM PEDIDO WHERENOMEF="Barateira")**

(E) **SELECT * FROM CLIENTE,PEDIDO WHERE CLIENTE.NOMEC=PEDIDO.NOMEC AND NOMEF<>"Barateira"**

Questão 12 -Prova: Petrobras 2012 – Engenharia de Software

Considere as informações a seguir para responder às questões de nos 31 a 33. As tabelas são utilizadas para descrever um banco de dados que armazena dados sobre linhas de ônibus, motoristas e viagens por eles realizadas.

```
CREATE TABLE MOTORISTA (
 MATRICULA NUMBER(7,0) NOT NULL,
 NOME VARCHAR2(50) NOT NULL,
 CPF NUMBER(11,0) NOT NULL,
 CNH VARCHAR2(15) NOT NULL,
 CONSTRAINT MOTORISTA_PK PRIMARY KEY (MATRICULA),
 CONSTRAINT MOTORISTA_UK1 UNIQUE (CPF),
 CONSTRAINT MOTORISTA_UK2 UNIQUE (CNH))
CREATE TABLE LINHA (
 NUMERO CHAR(5) NOT NULL,
 ORIGEM VARCHAR2(50) NOT NULL,
 DESTINO VARCHAR2(50) NOT NULL,
 CONSTRAINT LINHA_PK PRIMARY KEY (NUMERO)
)
CREATE TABLE VIAGEM (
 MAT_MOT NUMBER (7,0) NOT NULL,
 NUM_LINHA CHAR(5) NOT NULL,
 INICIO DATE NOT NULL,
 FINAL DATE,
 CONSTRAINT VIAGEM_PK PRIMARY KEY (MAT_MOT,NUM_LINHA,INICIO),
 CONSTRAINT VIAGEM_FK1 FOREIGN KEY (MAT_MOT) REFERENCES MOTORISTA (MATRICULA),
 CONSTRAINT VIAGEM_FK2 FOREIGN KEY (NUM_LINHA) REFERENCES LINHA (NUMERO))
```

Questão 12 - Prova: Petrobras 2012 – Engenharia de Software

31Considerando a possibilidade de que dois ou mais pares (ORIGEM,DESTINO) tenham o mesmo número de viagens, qual consulta permite exibir o par que possui o maior número de viagens (concluídas ou não) registradas no banco de dados?

(A) SELECT L.ORIGEM,L.DESTINO
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO
HAVING COUNT(*)=(SELECT MAX(COUNT(*))
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO)

(B) SELECT L.ORIGEM,L.DESTINO
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY V.NUM_LINHA
HAVING COUNT(*)=(SELECT MAX(COUNT(NUM_LINHA))
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO)

Questão 12 - Prova: Petrobras 2012 – Engenharia de Software

(C) SELECT L.ORIGEM,L.DESTINO
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO
HAVING COUNT(*)=(SELECT MAX(COUNT(DISTINCT NUM_LINHA))
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO)
(D) SELECT L.ORIGEM,L.DESTINO
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO
ORDER BY COUNT(*) DESC
(E) SELECT L.ORIGEM,L.DESTINO
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO AND COUNT(*)=(SELECT MAX(COUNT(*))
FROM VIAGEM V,LINHA L
WHERE V.NUM_LINHA=L.NUMERO
GROUP BY L.ORIGEM,L.DESTINO)
GROUP BY L.ORIGEM,L.DESTINO

Questão 13 -Prova: Petrobras 2012 – Engenharia de Software

Considere as informações a seguir para responder às questões de nos 31 a 33. As tabelas são utilizadas para descrever um banco de dados que armazena dados sobre linhas de ônibus, motoristas e viagens por eles realizadas.

```
CREATE TABLE MOTORISTA (
 MATRICULA NUMBER(7,0) NOT NULL,
 NOME VARCHAR2(50) NOT NULL,
 CPF NUMBER(11,0) NOT NULL,
 CNH VARCHAR2(15) NOT NULL,
 CONSTRAINT MOTORISTA_PK PRIMARY KEY (MATRICULA),
 CONSTRAINT MOTORISTA_UK1 UNIQUE (CPF),
 CONSTRAINT MOTORISTA_UK2 UNIQUE (CNH))
CREATE TABLE LINHA (
 NUMERO CHAR(5) NOT NULL,
 ORIGEM VARCHAR2(50) NOT NULL,
 DESTINO VARCHAR2(50) NOT NULL,
 CONSTRAINT LINHA_PK PRIMARY KEY (NUMERO)
)
CREATE TABLE VIAGEM (
 MAT_MOT NUMBER (7,0) NOT NULL,
 NUM_LINHA CHAR(5) NOT NULL,
 INICIO DATE NOT NULL,
 FINAL DATE,
 CONSTRAINT VIAGEM_PK PRIMARY KEY (MAT_MOT,NUM_LINHA,INICIO),
 CONSTRAINT VIAGEM_FK1 FOREIGN KEY (MAT_MOT) REFERENCES MOTORISTA (MATRICULA),
 CONSTRAINT VIAGEM_FK2 FOREIGN KEY (NUM_LINHA) REFERENCES LINHA (NUMERO))
```

Questão 13 - Prova: Petrobras 2012 – Engenharia de Software – Q32

Considere os parâmetros a seguir:

- Para o cálculo da média de viagens, devem ser levados em conta apenas os dias em que o motorista realizou pelo menos uma viagem, ao invés dos 31 dias do mês de março.
- As viagens não finalizadas não devem ser levadas em conta.
- Apenas o início da viagem precisa ocorrer no mês de março de 2012.
- A função TO_CHAR(INICIO,'DD') retorna o dia do mês (ex: 15).

Qual consulta permite exibir o CPF do motorista e o número médio de viagens diárias que cada um deles realizou no mês de março de 2012?

(A) SELECT M.CPF,COUNT(*)/COUNT(DISTINCT TO_CHAR(INICIO,'DD'))
FROM VIAGEM V,MOTORISTA M
WHERE M.MATRICULA=V.MAT_MOT AND INICIO >= '01-03-2012' AND
INICIO < '01-04-2012' AND FINAL IS NOT NULL
GROUP BY M.CPF,TO_CHAR(INICIO,'DD')

(B) SELECT M.CPF,AVG(TO_CHAR(INICIO,'DD'))
FROM VIAGEM V,MOTORISTA M
WHERE M.MATRICULA=V.MAT_MOT AND INICIO >= '01-03-2012' AND
INICIO < '01-04-2012' AND FINAL IS NOT NULL
GROUP BY M.CPF,TO_CHAR(INICIO,'DD')

Questão 13 - Prova: Petrobras 2012 – Engenharia de Software

(C) SELECT M.CPF,COUNT(*)/COUNT(DISTINCT TO_CHAR(INICIO,'DD'))
FROM VIAGEM V,MOTORISTA M
WHERE M.MATRICULA=V.MAT_MOT AND INICIO >= '01-03-2012' AND INICIO < '01-04-2012'
GROUP BY M.MATRICULA

(D) SELECT M.CPF,AVG(TO_CHAR(INICIO,'DD'))
FROM VIAGEM V,MOTORISTA M
WHERE M.MATRICULA=V.MAT_MOT AND INICIO >= '01-03-2012' AND INICIO < '01-04-2012'
GROUP BY M.CPF

(E) SELECT M.CPF,COUNT(*)/COUNT(DISTINCT TO_CHAR(INICIO,'DD'))
FROM VIAGEM V,MOTORISTA M
WHERE M.MATRICULA=V.MAT_MOT AND INICIO >= '01-03-2012' AND INICIO < '01-04-2012' AND FINAL IS NOT NULL
GROUP BY M.CPF

Questão 14 - Prova: Petrobras 2012 – Engenharia de Software Q.33

As figuras a seguir exibem os estados das tabelas MOTORISTA, LINHA e VIAGEM.

MOTORISTA

MATRICULA	NOME	CPF	CNH
22222	JOANA FONTES	22222222222	22222SP
44444	AUGUSTO MEDEIROS	44444444444	44444RJ
11111	CARLOS PACHECO	11111111111	11111RJ

LINHA

NUMERO	ORIGEM	DESTINO
233L	CENTRO	VILA NOVA
410	PINHEIROS	SANTA LUZIA

VIAGEM

MAT_MOT	NUM_LINHA	TO_CHAR(INICIO,'DD-MM-YYYYHH24:MI')	TO_CHAR(FINAL,'DD-MM-YYYYHH24:MI')
11111	233L	11-03-2012 08:00	11-03-2012 09:00
11111	233L	11-03-2012 09:30	11-03-2012 10:30
11111	410	01-03-2012 00:00	01-03-2012 01:00
11111	410	14-03-2012 13:00	14-03-2012 14:00
11111	410	14-03-2012 21:00	-
11111	410	15-03-2012 13:00	15-03-2012 14:00
11111	410	25-03-2012 13:00	25-03-2012 14:00
11111	410	31-03-2012 23:59	01-04-2012 01:00
11111	410	25-04-2012 13:00	25-04-2012 14:00
22222	233L	12-03-2012 09:30	12-03-2012 10:30
22222	410	11-03-2012 16:00	11-03-2012 17:00
22222	410	13-03-2012 17:00	-

Questão 14 - Prova: Petrobras 2012 – Engenharia de Software

Qual comando é capaz de alterar o estado do banco de dados com sucesso?

(A) DELETE FROM LINHA WHERE ORIGEM=’CENTRO’

(B) INSERT INTO VIAGEM (MAT_MOT,NUM_LINHA,INICIO) VALUES (22222,’410’,TO_DATE(‘11-03-2012 16:00’,’dd-mm-yyyy hh24:mi’))

(C) INSERT INTO VIAGEM VALUES (11111,’233L’, TO_DATE(‘19-03-2012 21:30’,’dd-mm-yyyy hh24:mi’))

(D) UPDATE MOTORISTA SET MATRICULA = 11112 WHERE MATRICULA = 11111

(E) UPDATE MOTORISTA V SET V.CPF=0 WHERE NOT EXISTS (SELECT * FROM VIAGEM WHERE MAT_MOT=V.MATRICULA)

Questão 15 - Prova: Petrobras 2012 – Infra-estrutura

Q 63. Gatilhos são procedimentos armazenados e especializados, ativados por eventos no banco de dados. Em SQL, um gatilho típico é composto de três componentes, que são:

(A) Evento, Condição e Ação

(B) Evento, Ação e Resultado

(C) Condição, Tratamento e Resultado

(D) Condição, Restrição e Ação

(E) Asserção, Evento e Resultado

Questão 16 - Prova: BNDES 2011 – Analista de Sistemas

Q. 53 As tabelas Projeto, Funcionario e Participacao_Projeto participam da base de dados de um sistema de controle de projetos.

A estrutura dessas tabelas é a seguinte:

Projeto (IDProjeto, Nome, DataInicio, DataFim)

Funcionario (Matricula, Nome, DataNascimento)

Participacao_Projeto (**IDProjeto, Matricula**)

Colunas sublinhadas participam da chave primária

Colunas em negrito participam de chaves estrangeiras

O comando SQL que retorna o nome somente dos funcionários que participaram de TODOS os projetos é:

Questão 16 - Prova: BNDES 2011 – Analista de Sistemas

(A) SELECT DISTINCT F.NOME FROM FUNCIONARIO F
INNER JOIN PARTICIPACAO_PROJETO PP
ON F.MATRICULA = PP.MATRICULA
INNER JOIN PROJETO P
ON P.IDPROJETO = PP.IDPROJETO

(B) SELECT DISTINCT F.NOME FROM FUNCIONARIO F
INNER JOIN PARTICIPACAO_PROJETO PP
ON F.MATRICULA = PP.MATRICULA
WHERE EXISTS (SELECT 1 FROM PROJETO P
WHERE P.IDPROJETO = PP.IDPROJETO)

(C) SELECT F.NOME FROM FUNCIONARIO F
WHERE MATRICULA IN
(SELECT MATRICULA FROM PARTICIPACAO_PROJETO)

Questão 16 - Prova: BNDES 2011 – Analista de Sistemas

(D) SELECT F.NOME FROM FUNCIONARIO F
WHERE NOT EXISTS
(SELECT 1 FROM PROJETO P
WHERE NOT EXISTS (SELECT 1 FROM
PARTICIPACAO_PROJETO PP
WHERE PP.IDPROJETO = P.IDPROJETO
AND PP.MATRICULA = F.MATRICULA))

(E) SELECT F.NOME FROM FUNCIONARIO F
WHERE NOT EXISTS
(SELECT 1 FROM PARTICIPACAO_PROJETO PP
WHERE PP.MATRICULA = F.MATRICULA
AND NOT EXISTS (SELECT 1 FROM PROJETO P
WHERE PP.IDPROJETO = P.IDPROJETO))

Questão 17 - Prova: BNDES 2011 – Analista de Sistemas

Q. 54 Na base de dados de um sistema de controle de clientes, foi criada a tabela CLIENTES, que conta com as colunas:

ID, NOME, ENDERECO, CIDADE e UF. Os valores da coluna ID não se repetem.

Sobre essa tabela CLIENTES foi criada a visão VCLIENTES_RJ, que busca apresentar os clientes do estado do Rio de Janeiro. O comando de criação da visão VCLIENTES_RJ é:

```
CREATE VIEW VCLIENTES_RJ
AS SELECT ID, NOME, ENDERECO, CIDADE, UF
FROM CLIENTES WHERE UF = 'RJ'
```

Um usuário submeteu o seguinte comando para execução pelo gerenciador do banco de dados:

```
UPDATE VCLIENTES_RJ SET NOME ='JOAO'  WHERE ID IN (1,2,3) AND UF = 'SP'
```

O comando UPDATE acima, quando submetido para execução, resulta na atualização de:

(A) nenhuma linha, pois, como a visão VCLIENTES_RJ somente apresenta clientes do Rio de Janeiro, não é possível

atualizar o nome de um cliente de São Paulo.

(B) nenhuma linha, pois não é possível realizar atualização sobre visões.

(C) até três das linhas da visão, cujo novo valor para a coluna Nome pode ser verificado através de consulta à própria visão

VCLIENTES_RJ.

(D) até três linhas da visão VCLIENTES_RJ, não sendo atualizadas linhas da tabela CLIENTES.

(E) até três linhas da tabela CLIENTES.

Quatão 18 - Prova: BNDES 2011 – Analista de Sistemas - Disc

Considere um sistema de pré-inscrição em disciplinas de uma universidade. A cada semestre letivo, diversas turmas são disponibilizadas. Pode haver uma ou mais turmas para cada disciplina. Os alunos escolhem as turmas que desejam frequentar. Dentre as tabelas da base de dados desse sistema, estão as tabelas Turmas e PreInscricoes. Há uma linha na tabela Turmas para cada turma disponibilizada e uma linha em PreInscricoes para cada inscrição de aluno em uma turma.

O quadro a seguir apresenta as principais colunas das tabelas Turmas e PreInscricoes, bem como o domínio de dados dessas colunas.

Tabela	Coluna	Domínio de dados	Participa em chave primária ou estrangeira?
Turmas	IDTurma	Inteiro	Participa na chave primária da tabela Turmas
	Sala	Inteiro	Não
	IDDisciplina	Inteiro	Participa em chave estrangeira que referencia a tabela disciplina
	HoraInicio	Data/Hora	Não
	HoraFim	Data/Hora	Não
	NumeroInscritos	Inteiro	Não
PreInscri-coes	IDTurma	Inteiro	Participa na chave primária da tabela PreInscricoes e em chave estrangeira que referencia a tabela Turmas
	MatriculaAluno	Inteiro	Participa na chave primária da tabela PreInscricoes e em chave estrangeira que referencia a tabela Alunos

Questão 18 - Prova: BNDES 2011 – Analista de Sistemas - Disc

a) Apresente o comando SQL (ANSI) que remove as linhas da tabela Turmas referentes às turmas que não possuem nenhum aluno inscrito.

- Delete from turmas
  Where idturma not in (
  select idturma from preinscricoes)

Questão 18 - Prova: BNDES 2011 – Analista de Sistemas - Disc

b) Apresente o comando SQL (ANSI) que atualiza a coluna NumeroInscritos da tabela Turmas com o total de alunos inscritos na turma.

- Update turmas
  Set numeroinscritos = (
    select count(*) from preinscricoes p
    where p.idturma = turmas.idturma)

Questão 18 - Prova: BNDES 2011 – Analista de Sistemas - Disc

c) Alguns alunos se inscrevem em mais de uma turma de uma mesma disciplina. Apresente o comando SQL (ANSI) que retorna o valor de MatriculaAluno para os alunos que se inscreveram em duas ou mais turmas da mesma disciplina.

- Select matriculaaluno from preinscricoes p1
  Inner join turmas t1 on t1.idturma = p1.idturma
  Where exists (
  select 1 from preinscricoes p2
  inner join turmas t2
  On t2.idturma = p2.idturma
  Where t2.iddisciplina = t1.iddisciplina
  And t2.idturma <> t1.idturma
  And p2.matriculaaluno = p1.matriculaaluno)

Questão 19 - PROVA: Transpetro 2012 – Analista de Sistemas Junior

Q. 45 Um projetista de banco de dados novato na profissão foi incumbido de criar um banco de dados para armazenar dados sobre clientes de uma empresa (identificador e nome), vendedores (identificador e nome) dessa empresa que realizam as vendas e sobre a data das realizações das vendas. Esse projetista decidiu criar uma única tabela, cujo esquema é apresentado abaixo.

```
CREATE TABLE VENDAS(
        ID_CLIENTE INTEGER,
        NOME_CLIENTE VARCHAR(60),
        ID_VENDEDOR CHAR(2),
        NOME_ VENDEDOR VARCHAR(60),
        DATA_VENDA DATE
);
```

Após criar a tabela, o projetista incluiu alguns registros nela de tal forma que seu estado atual é o que segue.

ID_CLIENTE	NOME_CLIENTE	ID_VENDEDOR	NOME_VENDEDOR	DATA_VENDA
118	João	V4	Izaias	12/04/2009
209	José	V7	Nogueira	10/08/2009
209	José	V4	Izaias	07/08/2010
360	Ana	V8	Martins	21/08/2010

Questão 19 - PROVA: Transpetro 2012 – Analista de Sistemas Junior

Ao analisar a solução proposta pelo novato, outro projetista mais experiente informou ao primeiro que, em virtude de a tabela não estar adequadamente normalizada, algumas operações em SQL realizadas sobre ela podem levar o banco de dados a um estado inconsistente ou à perda de informações. Quais são essas operações dentre as listadas abaixo?

(A) UPDATE, DELETE ou INSERT.

(B) UPDATE ou DELETE, apenas.

(C) DELETE ou INSERT, apenas.

(D) DELETE ou SELECT, apenas.

(E) INSERT ou SELECT, apenas.

ID_CLIENTE	NOME_CLIENTE	ID_VENDEDOR	NOME_VENDEDOR	DATA_VENDA
118	João	V4	Izaias	12/04/2009
209	José	V7	Nogueira	10/08/2009
209	José	V4	Izaias	07/08/2010
360	Ana	V8	Martins	21/08/2010

Questão 20 - CESGRANRIO - 2012 - CMB - Assistente Técnico - Administrativo

Na linguagem SQL, para filtrar registros a serem atualizados em uma tabela pelo comando UPDATE, usa-se a cláusula:

(a) SUM

(b) FROM

(c) WHERE

(d) QUERY

(e) FILTER

Questão 21 - CESGRANRIO - 2012 - CMB - Assistente Técnico - Administrativo

Considere a tabela de nome Filial contendo 10 registros em um determinado banco de dados. Os atributos dessa tabela são:

- Codigo
- Nome
- Estado
- Cidade

Para excluir todos os registros dessa tabela, deve-se usar o comando

(a) DELETE FROM Filial
(b) DELETE Codigo, Nome, Estado, Cidade FROM Filial
(c) REMOVE Codigo, Nome, Estado, Cidade FROM Filial
(d) SELECT * FROM Filial REMOVE ALL
(e) UPDATE Filial SET ALL = <null>

Questão 22 - PROVA: Transpetro 2012 – Analista de Sistemas Junior

Q. 47 Considere que um banco de dados relacional foi criado através da execução dos comandos em SQL a seguir.

```
CREATE TABLE Empregado (
    empregadoMatricula char(5) PRIMARY KEY,
    empregadoNome varchar(25),
    departamentoID int
);

CREATE TABLE Departamento (
    departamentoID int UNIQUE,
    departamentoNome varchar(25)
);

ALTER TABLE Empregado ADD CONSTRAINT fk_empregado_depto
FOREIGN KEY (departamentoID)
REFERENCES Departamento(departamentoID);

INSERT INTO Departamento VALUES (1, 'Vendas');
INSERT INTO Departamento VALUES (3, 'Engenharia');
INSERT INTO Departamento VALUES (4, 'Contabilidade');
INSERT INTO Departamento VALUES (5, 'Propaganda');
INSERT INTO Empregado VALUES ('10001', 'Edméia', 1);
INSERT INTO Empregado VALUES ('10003', 'Suely', 3);
INSERT INTO Empregado VALUES ('10005', 'Marcelo', 3);
INSERT INTO Empregado VALUES ('10007', 'Vagner', 4);
INSERT INTO Empregado VALUES ('10009', 'Adriana', 4);
INSERT INTO Empregado VALUES ('10011', 'João', NULL);
```

Questão 22 - PROVA: Transpetro 2012 – Analista de Sistemas Junior

Agora considere que a consulta de seleção fornecida a seguir foi executada sobre o banco de dados com esquema e estado resultantes da sequência de comandos acima.

```
SELECT *
FROM Empregado LEFT OUTER JOIN Departamento
          ON Empregado.departamentoID = Departamento.departamentoID;
```

Qual a quantidade de registros retornados pela consulta acima?

(A) 3 (B) 4 (C) 5 (D) 6 (E) 7

ITnerante

TIMASTERS

Prof. Thiago Cavalcanti

Questão 23 - Prova: ESAF - STN 2013 – Infra – Q.13

13- As três cláusulas de uma consulta SQL são:

a) *start, from, to*.

b) *select, from, where*.

c) *select, up, what*.

d) *start, from, who*.

e) *select, initial, final*.

Questão 24 - Prova: ESAF - STN 2013 – Infra – Q.14

14- Assinale a opção correta relativa a operações em SQL.

a) *Strings* são especificados utilizando-se *slashes*.

b) O operador de comparação **diffrent** permite pesquisar disparidades.

c) A cláusula **as** é particularmente útil para definir a noção de variáveis de tupla.

d) A cláusula **until** é particularmente útil para definir a noção de variáveis de tupla.

e) O operador de localização **not loop** permite pesquisar disparidades.

Questão 25 - Prova: ESAF - STN 2013 – Gestão – Q.24

24- Assinale a opção correta.

a) A DML procedural requer a especificação de que dados são necessários e a forma de obtê-los.

b) A DML associativa requer apenas a especificação de que dados são necessários.

c) A DML funcional requer a especificação de que dados são necessários e a forma de obtê-los.

d) A DML declarativa requer a especificação de que dados são necessários e a forma de obtê-los.

e) A DML procedural requer apenas a especificação de que dados são necessários.

Questão 26 - ESAF - 2012 - CGU - Analista de Finanças e Controle - prova 3 - Desenvolvimento de Sistemas

A Linguagem de Definição de Dados de uma SQL permite especificar:

a) O esquema para cada estrutura. As restrições de normalização. O conjunto dos índices a serem mantidos para cada relação. As informações de acesso e flexibilidade para cada relação.

b) O domínio dos valores gerados por cada atributo. As restrições de integridade. O conjunto dos coeficientes a serem mantidos nas relações de objetos. A estrutura de armazenamento físico de cada relação na estrutura de relacionamentos.

c) O esquema para cada relação. O domínio dos valores das relações. As permissões de integridade. O conjunto dos índices a serem extraídos de cada relação.

d) O esquema para cada relação. As informações de segurança e autorização para cada relação. A estrutura de armazenamento físico de cada relação no disco.

e) As restrições de interatividade. As variáveis e índices a serem mantidos para cada relação. As informações de segurança e automação para cada relação. A estrutura de armazenamento físico de cada relação no disco.

Questão 27 - ESAF - 2008 - CGU - Tecnologia da Informação - Desenvolvimento de Sistemas

Em relação às cláusulas, funções e operadores SQL, é correto afirmar que

a) a cláusula *GROUP BY* é utilizada para especificar as condições que devem reunir os registros que serão selecionados.

b) a função de soma *SUM* é utilizada para devolver o número de registros da seleção.

c) a cláusula *HAVING* somente pode ser especificada em conjunto com a cláusula *GROUP BY*.

d) o operador *UNION ALL* combina os resultados de duas consultas SQL em uma única tabela, desde que as consultas tenham o mesmo número de colunas e dados compatíveis. Nesse caso, os registros duplicados são automaticamente removidos.

e) a ordem de duas instruções *SELECT* que fazem uso do operador *EXCEPT* não altera o resultado da consulta.

Questão 28 - ESAF - 2006 - CGU - Analista de Finanças e Controle - Tecnologia da Informação - Prova 3

Analise as seguintes afirmações relacionadas a conceitos básicos de banco de dados e linguagem SQL.

I. Na linguagem SQL um INNER JOIN retorna todas as tuplas comuns às duas tabelas.

II. Em uma Junção entre duas tabelas a cláusula USING só poderá ser usada quando o nome do atributo for igual nas duas tabelas.

III. Na linguagem SQL um RIGHT OUTER JOIN retorna todas as tuplas que não são comuns às duas tabelas.

IV. Uma Junção é usada para compor informações complexas a partir de tabelas sem nenhum tipo de relacionamento.

Indique a opção que contenha todas as afirmações verdadeiras.

(a) I e III

(b) II e III

(c) III e IV

(d) I e II

(e) II e IV

Questão 29 - ESAF - 2010 - CVM - Analista de TIC - Infraestrutura

Um *trigger*

a) é uma instrução que o sistema executa, sob comando do usuário, para restauração colateral de um banco de dados com *trigs.*

b) pertence à tríade usuário-entidade-ação.

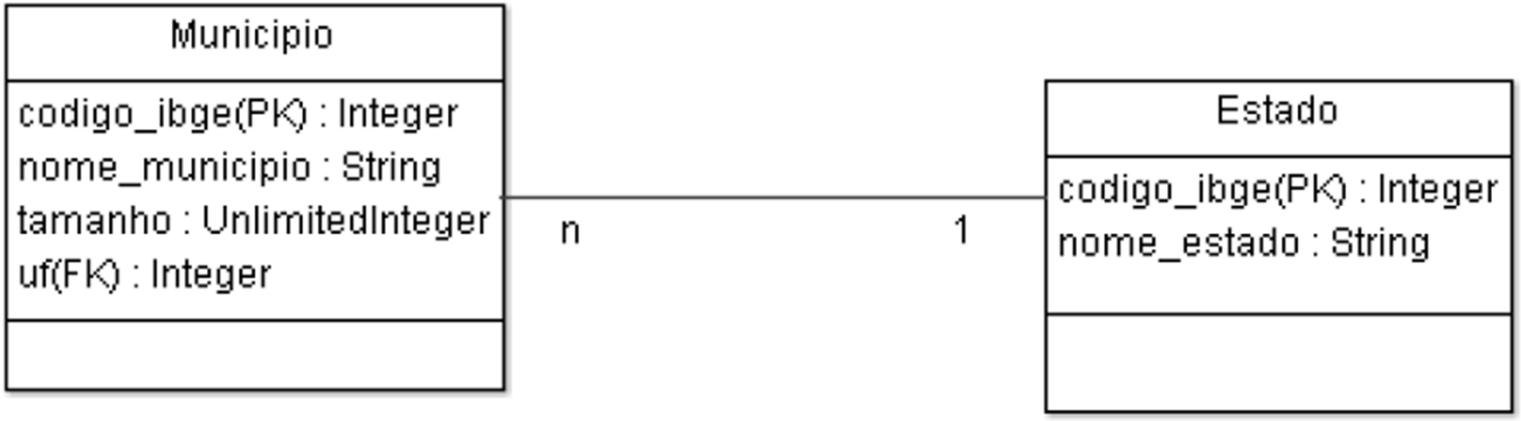
c) é uma instrução que o sistema executa automaticamente como um efeito colateral de uma modificação no banco de dados.

d) é criado pelo modelo premissa-condição- ação.

e) é um instrumento do *actiondriver.*

Prof. Thiago Cavalcanti

Questão 30 - Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas – Q. 31



O diagrama acima apresenta uma classe denominada Municipio, para armazenar informações sobre determinado município, seu código, nome, tamanho e o estado a que pertence. Também há uma classe denominada Estado, para armazenar dados da unidade da Federação, como seu código e sua denominação. Com base nesse diagrama, e considerando a utilização da linguagem SQL, assinale a opção que apresenta a forma correta de criação dessas classes em tabelas, considerando a criação das chaves primárias (PK) e estrangeiras (FK).

Questão 30 - Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas – Q. 31

(a) create table estado(id integer primary key, nome varchar);

create table municipio (ibge integer primary key, nome varchar, tamanho bigint, uf char(2), constraint fk_estado foreign key (ibge) references estado);

(b) create table estado(codigo_ibge char(7), nome_estado integer);

create table municipio(codigo_ibge char(7), nome_municipio integer, tamanho bigint, uf integer, constraint fk_estado foreign key (codigo_ibge) references estado);

(c) create table estado(codigo_ibge integer primary key, nome_estado varchar);

create table municipio(codigo_ibge integer primary key,nome_municipio varchar, tamanho bigint, uf integer, constraint fk_estado foreign key (uf) references estado);

Questão 30 - Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas – Q. 31

(d) create table estado(codigo_ibge integer primary key, nome_estado varchar);

create table municipio(codigo_ibge integer primary key,
nome_municipio varchar, tamanho bigint, uf integer,
constraint fk_estado foreign key (codigo_ibge) references estado);

(e) create table municipio(codigo_ibge integer primary key,
nome_estado varchar);

create table estado(codigo_ibge integer primary key, nome_municipio varchar, tamanho bigint, uf integer,
constraint fk_estado foreign key (codigo_ibge) references estado);

Questão 31 - Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas – Q. 37

A Linguagem de Manipulação de Dados (DML) é uma linguagem de consulta que se baseia tanto na álgebra relacional quanto no cálculo relacional de tuplas. Os comandos que fazem parte da DML incluem

(a) SELECT, CREATE TABLE e CREATE INDEX.

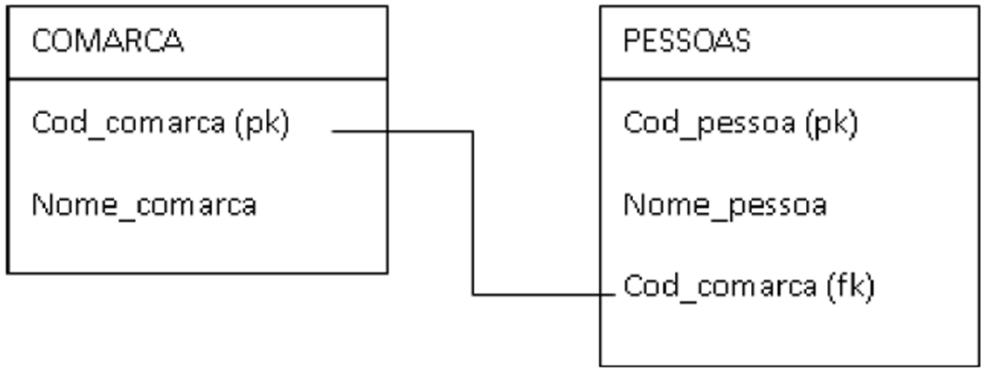
(b) INSERT, SELECT, UPDATE e DELETE.

(c) CREATE TABLE, CREATE INDEX e DROP VIEW.

(d) CREATE TABLE, CREATE INDEX e CREATE VIEW.

(e) CREATE TABLE, INSERT, UPDATE e DELETE.

Questão 32 - Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas – Q. 37



Em conformidade com as tabelas acima identificadas, assinale a opção correspondente a consulta escrita na linguagem SQL cuja execução retornará o nome de todas as COMARCAS que não tenham PESSOAS relacionada a esses nomes.

Questão 32 - Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas – Q. 37

(a) (SELECT nome_comarca
FROM comarca C)
EXCEPT
(SELECT nome_comarca
FROM comarca C, pessoas P
WHERE C.Cod_comarca=P.Cod_comarca)

(b) SELECT nome_comarca
FROM comarca C, pessoas P
WHERE C.Cod_comarca<>P.Cod_comarca

(c) SELECT nome_comarca
FROM comarca C
WHERE Cod_comarca IN
(SELECT Cod_comarca
FROM pessoas P)

(d) SELECT nome_comarca
FROM comarca C, pessoas P
WHERE C.Cod_comarca=P.Cod_comarca
AND P.Cod_pessoa IS NULL

(e) SELECT nome_comarca
FROM comarca C
WHERE EXISTS
(SELECT *
FROM pessoas P
WHERE Cod_comarca=C.Cod_comarca)

Questão 33 - Prova: CESPE - 2012 - TRE-RJ - Técnico Judiciário - Programação de Sistemas

Julgue os itens que se seguem, a respeito de comandos SQL.

[54] INSERT INTO é o comando utilizado para inserir dados em uma tabela.

[55] SORT BY é o comando indicado para ordenar o resultado de uma consulta.

[56] O comando EXTRACT é utilizado para extrair dados de uma tabela em SQL.

Questão 34 - Prova: CESPE - 2008 - STJ - Analista Judiciário - Tecnologia da Informação

Acerca de arquiteturas de aplicações cliente-servidor e para a Internet, julgue os próximos itens.

[85] Gatilhos (triggers) podem ser usados para impor restrições de integridade semântica em um banco de dados relacional. Um gatilho pode especificar evento, condição e ação. A ação é executada se a condição for satisfeita quando ocorrer o evento. Se uma condição não for especificada, a ação será disparada pelo evento. Uma ação pode ser uma sucessão de declarações SQL.

Questão 35 - Prova: CESPE - 2009 - ANAC - Técnico Administrativo - Informática

A respeito de SQL e PL/SQL, julgue os itens a seguir.

[101] Quando se executa com sucesso o comando SQL “TRUNCATE TABLE cidades”, todos os registros da tabela cidades são removidos.

[102] O comando SQL DROP TABLE permite copiar os registros de uma tabela para outra e a seleção dos registros a serem copiados deve ser especificada pelas cláusulas FROM e WHERE.

[103] O comando SQL ALTER TABLE possibilita alterar a estrutura de uma tabela, como por exemplo, adicionando ou removendo uma coluna de uma tabela.

[105] A estrutura básica de um bloco PL/SQL é composta por quatro seções: DECLARE, BEGIN, EXCEPTION e END. As seções DECLARE, BEGIN e END são obrigatórias

Questão 36 - Prova: CESPE - 2008 - STF - Analista Judiciário - Tecnologia da Informação

O armazenamento e a recuperação de grandes quantidades de dados é um trabalho importante e muito explorado em um sistema gerenciador de banco de dados (SGBD). Com relação aos conceitos que envolvem esse sistema, julgue os itens que se seguem.

[83] A operação de junção externa (outer join) é uma extensão da operação de junção para tratar informações omitidas.

[85] Em uma consulta SQL, o operador DISTINCT irá remover todas as colunas duplicadas do conjunto que forma o resultado.

[86] Apenas as operações union e intersect são disponibilizadas pela linguagem SQL para manipulação de conjuntos.

[91] Duas exigências devem ser satisfeitas para a definição de um mecanismo de gatilho: especificar as condições nas quais o gatilho deve ser executado; e especificar as ações que devem ser tomadas quando um gatilho for disparado.

ITnerante

TIMASTERS

Questão 37 - Prova: FCC - 2012 - MPE-AP - Analista Ministerial - Tecnologia da Informação

Atenção: Utilize os comandos SQL abaixo para responder às questões de números 52 e 53.

```
CREATE TABLE Time (Codigo INT, Nome VARCHAR(40));
CREATE TABLE Jogo (Time1 INT, Time2 INT,Placar1 INT,Placar2 INT);
INSERT INTO Time VALUES (1,"Brasil");
INSERT INTO Time VALUES (2,"Argentina");
INSERT INTO Time VALUES (3,"Bolivia");
INSERT INTO Jogo VALUES (1,2,10,0);
INSERT INTO Jogo VALUES (3,2,4,2);
INSERT INTO Jogo VALUES (1,3,2,0);
```

Note que os exemplos abaixo consideram que as linhas apresentadas acima já foram executadas. Para representar a separação de colunas em uma consulta, será utilizado o símbolo | (barra vertical).

Questão 37 - Prova: FCC - 2012 - MPE-AP - Analista Ministerial - Tecnologia da Informação

Para que o resultado de uma consulta consiga trazer apenas as seguintes linhas e colunas:

Brasil|10|0|Argentina
Bolivia|4|2|Argentina
Brasil|2|0|Bolívia

É necessária a execução do comando
(a) SELECT a.Nome,b.Placar1,b.Placar2,a.Nome FROM Time a, Jogo b;
(b) SELECT (SELECT Nome FROM Time WHERE Time.Codigo = Jogo.Time1) as Time1,Placar1,Placar2,(SELECT Nome FROM Time WHERE Time.Codigo = Jogo.Time2) as Time2 FROM Jogo;
(c) CREATE VIEW champ AS SELECT a.Nome,b.Placar1,c.Nome,b.Placar2 WHERE (b.Placar1 > b.Placar2 AND b.Time1 = a.Codigo) OR (b.Placar2 > b.Placar1 AND b.Time2 = a.Codigo) FROM Time as c,Jogo as b,Time as c;
(d) SELECT a.*,b.* FROM Time a, Jogo b INNER JOIN Time;
(e) SELECT a.Nome,b.Placar1,b.Placar2,a.Nome FROM Time a, Jogo b LEFT JOIN Jogo;

Atenção: Para responder às questões de números 31 e 32, utilize o código SQL abaixo.

```
CREATE TABLE livros(Codigo INT,Titulo TEXT,Autor INT,ISBN TEXT NOT NULL);
CREATE TABLE autores(Nome TEXT,Codigo INT);
CREATE VIEW view1 AS
        SELECT A.Codigo,A.Titulo,A.ISBN,A.Autor,B.Nome
        FROM livros A, autores B ON A.Autor = B.Codigo ORDER BY A.Codigo;
CREATE TRIGGER tr_livros
BEFORE INSERT ON livros
FOR EACH ROW BEGIN
        SELECT CASE
        WHEN
                (NEW.Codigo IS NULL) OR
                (NEW.Titulo IS NULL) OR
                (NEW.Autor IS NULL) OR
                ((SELECT
                        Codigo FROM     livros
                        WHERE Codigo = NEW.Codigo) IS NOT NULL) OR
                ((SELECT
                        Codigo FROM autores
                        WHERE Codigo = NEW.Autor) IS NULL)
        THEN
                RAISE(ABORT,"Dados invalidos")
        END;
END;

– insert 1
INSERT INTO autores VALUES("Boris Pasternak",1);
– insert 2
INSERT INTO livros VALUES(1,"Doutor Jivago",1,"9788577990375");
– insert 3
INSERT INTO autores VALUES("Margaret Mitchell",2);
– insert 4
INSERT INTO livros VALUES(1,"E o Tempo Levou",2,"");
– query view
SELECT * FROM view1;
```

Verifica se o livro já existe!

Verifica se o autor não existe!

Questão 38 - TRIBUNAL REGIONAL FEDERAL DA 2 REGIÃO - Analista Judiciário - Informática - 2012

31. Com base no código apresentado, ao ser executado o comando:

```
CREATE TABLE livros(Codigo INT,Titulo TEXT,Autor INT,ISBN TEXT NOT NULL);
CREATE TABLE autores(Nome TEXT,Codigo INT);
```

-- insert 4

INSERT INTO livros VALUES(1,"E o Tempo Levou",2,"");

Será:

(a) executado o comando RAISE(ABORT,"Dados invalidos").

(b) inserida uma nova linha na tabela livros.

(c) criada uma nova linha na VIEW view1.

(d) ignorado, pois o ISBN não pode ser nulo (NULL).

(e) inserida uma nova coluna na tabela livros.

Questão 38 - TRIBUNAL REGIONAL FEDERAL DA 2 REGIÃO - Analista Judiciário - Informática - 2012

31. Com base no código apresentado, ao ser executado o comando:

```
CREATE TABLE livros(Codigo INT,Titulo TEXT,Autor INT,ISBN TEXT NOT NULL);
CREATE TABLE autores(Nome TEXT,Codigo INT);
```

-- insert 4

INSERT INTO livros VALUES(1,"E o Tempo Levou",2,"");

Será:

(a) executado o comando RAISE(ABORT,"Dados invalidos").

(b) inserida uma nova linha na tabela livros.

(c) criada uma nova linha na VIEW view1.

(d) ignorado, pois o ISBN não pode ser nulo (NULL).

(e) inserida uma nova coluna na tabela livros.

Questão 39 - TRIBUNAL REGIONAL FEDERAL DA 2 REGIÃO - Analista Judiciário - Informática - 2012

Q.32. Sobre o código apresentado, considere:

I . O comando SELECT * FROM view1; exibirá informações sobre os dois livros inseridos durante a execução deste código com seus respectivos autores.

II. O trecho de SQL (SELECT Codigo FROM livros WHERE Codigo = NEW.Codigo) IS NOT NULL garante que o código do livro sendo inserido é único.

III . A cláusula BEFORE INSERT ON livros configura a trigger para ser executada antes de um insert ou update na tabela livros.

É correto o que consta em

(a) II e III , apenas.

(b) I , II e III .

(c) II, apenas.

(d) I e III , apenas.

(e) III , apenas.

Questão 39 - TRIBUNAL REGIONAL FEDERAL DA 2 REGIÃO - Analista Judiciário - Informática - 2012

Q.32. Sobre o código apresentado, considere:

I . O comando SELECT * FROM view1; exibirá informações sobre os dois livros inseridos durante a execução deste código com seus respectivos autores.

II. O trecho de SQL (SELECT Codigo FROM livros WHERE Codigo = NEW.Codigo) IS NOT NULL garante que o código do livro sendo inserido é único.

III . A cláusula BEFORE INSERT ON livros configura a trigger para ser executada antes de um insert ou update na tabela livros.

É correto o que consta em

(a) II e III , apenas.

(b) I , II e III .

(c) II, apenas.

(d) I e III , apenas.

(e) III , apenas.

Questão 40 - TRIBUNAL DE JUSTIÇA DO ESTADO DE PE – APJ - Analista de Suporte - 2012

44. Em SQL, a função utilizada para extrair caracteres de um campo texto é

(a) MAX().

(b) LEN().

(c) AVG().

(d) MID().

(e) FORMAT().

Questão 40 - TRIBUNAL DE JUSTIÇA DO ESTADO DE PE – APJ - Analista de Suporte - 2012

44. Em SQL, a função utilizada para extrair caracteres de um campo texto é

(a) MAX().

(b) LEN().

(c) AVG().

(d) MID().

(e) FORMAT().

Questão 41 - TRIBUNAL REGIONAL FEDERAL DA 2 REGIÃO -Analista Judiciário Informática - 2012

Q.33. Sobre transações em SQL, considere:

I . Uma transação é uma série de manipulação de dados em comandos SQL que executa uma unidade de trabalho lógica.

II. Os comandos COMMIT, ROLLBACK e INTERSECT fazem parte do controle de transações do SQL.

III . O comando COMMIT garante que as mudanças efetuadas durante a transação sejam armazenadas de forma permanente no banco de dados, terminando a transação. O comando ROLLBACK garante que as mudanças efetuadas dentro da transação sejam ignoradas, porém não termina a transação até que o comando END TRANSACTION seja executado.

É correto o que consta em

(a) I , apenas.

(b) I , II e III .

(c) III , apenas.

(d) II e III , apenas.

(e) I e II, apenas.

Questão 41 - TRIBUNAL REGIONAL FEDERAL DA 2 REGIÃO -Analista Judiciário Informática - 2012

Q.33. Sobre transações em SQL, considere:

I . Uma transação é uma série de manipulação de dados em comandos SQL que executa uma unidade de trabalho lógica.

II. Os comandos COMMIT, ROLLBACK e ~~INTERSECT~~ fazem parte do controle de transações do SQL.

III . O comando COMMIT garante que as mudanças efetuadas durante a transação sejam armazenadas de forma permanente no banco de dados, terminando a transação. O comando ROLLBACK garante que as mudanças efetuadas dentro da transação sejam ignoradas~~, porém não termina a transação até que o comando END TRANSACTION seja executado.~~

É correto o que consta em

(a) I , apenas.

(b) I , II e III .

(c) III , apenas.

(d) II e III , apenas.

(e) I e II, apenas.

Prof. Thiago Cavalcanti

Questão 42 - FGV - 2010 - CODESP-SP - Analista de Sistemas

- Analise o comando SQL abaixo:

```
SELECT P.P#,
       'Peso em gramas =' AS TEXTO1,
       P.PESO * 454 AS PESOGM,
       P.COR,
       'Quantidade máxima =' AS TEXTO2,
       MAX ( FP.QDE ) AS QDEMAX
FROM   P, FP
WHERE  P.P# = FP.P#
AND   ( P.COR = COR ('Vermelho') OR P.COR = COR ('Azul') )
AND    FP.QDE > QDE ( 200 )
GROUP  BY P.P#, P.PESO, P.COR
HAVING SUM ( FP.QDE ) > QDE ( 350 ) ;
```

Questão 42 - FGV - 2010 - CODESP-SP - Analista de Sistemas

Da análise feita, verifica-se que é correto afirmar que

a) os resultados que satisfazem à condição SUM (FP.QDE) > QDE (350) são eliminados.

b) os resultados que satisfazem à condição indicada na cláusula WHERE são agrupados pelos valores das colunas indicadas na cláusula GROUP BY.

c) a cláusula FROM é avaliada para produzir uma tabela temporária, com eliminação das tuplas em duplicata na tabela FP e classificada por PESOGM.

d) a cláusula FROM é avaliada para produzir uma nova tabela P, a partir da execução do SELECT sobre a tabela FP.

e) as linhas que satisfazem às condições indicadas na cláusula WHERE são eliminadas.

Questão 43 - FGV - 2010 - DETRAN-RN - Assessor Técnico - Administração de Banco de DadosDisciplina

Sobre o comando "**drop table pedido;**" assinale a alternativa correta:

a) Cria a tabela pedido.

b) Elimina a tabela pedido.

c) Duplica a tabela pedido.

d) Cria uma chave primária na tabela pedido.

e) Extrai dados da tabela pedido.

Questão 44 - FGV - 2010 - DETRAN-RN - Assessor Técnico - Administração de Banco de Dados

Assinale a alternativa que corresponde à funcionalidade da seguinte sintaxe: **SELECT * FROM tabela ;**

a) Selecionar apenas a primeira coluna da tabela.

b) Inserir o símbolo asterisco na tabela.

c) Selecionar todas as colunas da tabela.

d) Inserir a coluna asterisco na tabela.

e) Apagar todas as colunas da tabela.

Questão 45 - FGV - 2010 - DETRAN-RN - Assessor Técnico - Administração de Banco de Dados

Assinale a alternativa correta sobre fragmento de comando a seguir:

(**select distinct** *nome_cliente*
from *contas*)
intersect
(**select distinct** *nome_cliente*
from *emprestimos*)

a) Se um cliente possui conta mas não possui empréstimos no banco, aparecerá no resultado.

b) Se um cliente tem diversas contas e empréstimos no banco, aparecerá todas as repetições no resultado.

c) Se um cliente tem diversas contas e empréstimos no banco, não aparecerá no resultado.

d) Se um cliente tem diversas contas e empréstimos no banco, aparecerá somente uma vez no resultado.

e) Se um cliente não possui conta mas possui empréstimos no banco, aparecerá no resultado.


“Os planos do diligente levam à vantagem certa.”
Provérbios 21:5
Diligência
É uma habilidade adquirida que combina persistência criativa, esforço inteligente, planejado e executado de forma honesta , com competência e eficácia, de modo a alcançar um resultado dentro do mais alto nível de excelência.
Steven K. Scott (adaptado)

Valeu Galera!! :P

Banco de dados

Modulo III – Linguagem SQL

Curso Preparatório - ITnerante

Prof. Thiago Cavalcanti

Gabarito

1. [75] C
2. [112] C
3. A
4. A
5. A
6. C
7. 95 C 95 E
8. E
9. B
10. Disc
11. B
12. A
13. E
14. E
15. A
16. D
17. A
18. Disc
19. A
20. C
21. A
22. D
23. B
24. C
25. A
26. D
27. C
28. D
29. C
30. C
31. B
32. A
33. [54] C [55] E [56] E
34. [85] C
35. [101] C [102] E [103] C [104] Anulada [105] E
36. [83] C [85] E [86] E [91] C
37. B
38. A
39. C
40. D
41. A
42. B
43. B
44. C
45. D