Exemplos de Vistas

- Considere-se a vista (com o nome `todosOsConsumidores`) contento os nomes das agências e seus clientes.

`CREATE VIEW` todosOsClientes `As`

$$\Pi_{\text{nomeBalcao,nomeCliente}}(\text{temConta} \bowtie \text{conta})$$
$$\cup \quad \Pi_{\text{nomeBalcao,nomeCliente}}(\text{temEmprestimo} \bowtie \text{emprestimo})$$

- Pode-se assim encontrar todos os clientes da agência de `Penacova` através de:

$$\Pi_{\text{nomeBalcao}}(\sigma_{\text{nomeBalcao}='\text{Penacova}'}(\textit{todosOsClientes}))$$

SQL

- Linguagem de Definição dos Dados
  - Tipos em SQL
  - Manipular Tabelas (relações)
  - Integridade e Segurança
- Linguagem de Manipulação dos Dados
  - Estrutura básica
  - Operações com conjuntos
  - Funções de agregação
  - Valores nulos
  - Junções
  - Sub-consultas embebidas
  - Relações derivadas
  - Vistas
  - Modificação da Base de Dados
- Embedded SQL

Linguagem de Definição de Dados (DDL)

Linguagem para especificar a informação acerca de cada relação, incluindo:

- O esquema de cada relação.
- O domínio de valores associados com cada atributo.
- Restrições de integridade.
- O conjunto de índices a manter para cada relação.
- Informação de segurança e autorização para cada relação.
- As estruturas de armazenamento físico em disco de cada relação.

Tipos em SQL

- `char(n)`: cadeia de caracteres de comprimento fixo *n*.
- `varchar(n)`: cadeia de caracteres de comprimento variável, com o máximo *n* especificado pelo utilizador.
- `int`: inteiro (um subconjunto finito dos inteiros, dependente da máquina).
- `smallint`: inteiro pequeno (um subconjunto do tipo `int`).
- `numeric(p,d)`: número de **vírgula fixa**, com de *p* dígitos no total e com *d* casas decimais.
- `real, double precision`: Números de **vírgula flutuante**, com precisão dependente da máquina.
- `float(n)`: número de vírgula flutuante, com um mínimo de precisão de *n* dígitos.

Os valores “nulos” (`NULL`) são permitidos em todos os tipos de dados.
A declaração de um atributo como `NOT NULL` proíbe os valores nulos para esse atributo.

Tipos Data/Tempo em SQL (cont.)

- `date`: datas, contendo um ano com (4 dígitos), mês e dia, (“2001-7-27”);
- `time`: tempo (diário), em horas, minutos e segundos, (“09:00:30.75”);
- `timestamp`: data mais hora, (“2001-7-27 09:00:30.75”).
- `Interval`: período de tempo.

A subtracção de dois valores de date/time/timestamp devolve um intervalo.
Os valores de intervalos podem ser adicionados a valores de date/time/timestamp.
Pode-se extrair campos do valor date/time/timestamp.

Tipos em MySQL

```
TINYINT[(tamanho)] [UNSIGNED] [ZEROFILL]
SMALLINT[(tamanho)] [UNSIGNED] [ZEROFILL]
MEDIUMINT[(tamanho)] [UNSIGNED] [ZEROFILL]
INT[(tamanho)] [UNSIGNED] [ZEROFILL]
INTEGER[(tamanho)] [UNSIGNED] [ZEROFILL]
BIGINT[(tamanho)] [UNSIGNED] [ZEROFILL]
REAL[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
DOUBLE[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
FLOAT[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
DECIMAL(tamanho,decimais) [UNSIGNED] [ZEROFILL]
NUMERIC(tamanho,decimais) [UNSIGNED] [ZEROFILL]
CHAR(tamanho) [BINARY | ASCII | UNICODE]
VARCHAR(tamanho) [BINARY]
DATE
TIME
TIMESTAMP
DATETIME
TINYBLOB
BLOB
MEDIUMBLOB
LONGBLOB
TINYTEXT
TEXT
MEDIUMTEXT
LONGTEXT
ENUM(value1,value2,value3,...)
SET(value1,value2,value3,...)
```


Instrução Create Table

- Uma tabela SQL é definida recorrendo ao comando `CREATE TABLE`:

  `CREATE TABLE` r `(`A_1 D_1`,` A_2 D_2`, ...,` A_n D_n`,`
  `restrição_de_ integridade 1,`
  `...,`
  `restrição_de_integridade k)`

  - r é o nome da relação;
  - A_i é o nome de um atributo no esquema de relação r;
  - D_i é o tipo de dados dos valores no domínio do atributo A_i.

- Exemplo:

```
CREATE TABLE balcao
  (balcaoNome char(15) not null,
  balcaoCidade char(30),
  depositos integer)
```


Restrições de integridade

- `not null`
- `primary key (`A_1`, ...,` A_n`)`
- `unique (`A_1`, ...,` A_n`)`
- `check (`P`)`, em que P é um predicado.

Exemplo:

```
CREATE TABLE balcao
  (balcaoNome char(15),
  balcaoCidade char(30),
  depositos integer,
  PRIMARY KEY (balcaoNome),   CHECK (depositos >= 0))
```

Instrução Drop Table

O comando `DROP TABLE` remove da base de dados toda a informação sobre a relação (e não apenas os dados).

```
DROP TABLE Emprestimo
```

Caso hajam chaves externas deve-se utilizar a declaração `cascade constraints.`

```
DROP TABLE Emprestimo CASCADE CONSTRAINTS
```


Instrução Alter Table

O comando `ALTER TABLE` é utilizado para modificar o esquema, ou as restrições sobre relações já existente.

- Para adicionar novos atributos:

  `ALTER TABLE` *r* `ADD` *A D*

  em que *A* é o nome do atributo a adicionar à relação *r* e *D* o domínio de *A*.
  Todos os tuplos existentes ficam com `NULL` no novo atributo.
- Para eliminar atributos de uma relação

  `ALTER TABLE` *r* `DROP` *A*

  em que *A* é o nome de um atributo na relação *r*.

Instrução Alter Table (cont.)

- Para adicionar novas restrições:

  `ALTER TABLE` *r* `ADD CONSTRAINT` *N R*

  em que *N* é um nome dado à nova restrição e *R* define a restrição. Por exemplo:

  ```
  ALTER TABLE Conta ADD CONSTRAINT saldo_pos CHECK (balanco
  > 0)
  ```
- Para remover restrições (definidas previamente com um nome):

  `ALTER TABLE` *r* `DROP CONSTRAINT` *N*

  por exemplo:

  ```
  ALTER TABLE Conta DROP CONSTRAINT saldo_pos
  ```


Integridade e Segurança

- Restrições ao Domínio
- Integridade Referencial
- Asserções
- Segurança e Autorizações

Restrições ao Domínio

- As restrições de integridade impõem-se para garantir que os dados fiquem protegidos contra "estragos" acidentais. Devem asseguram que da actualização dos dados não resulta a perda da consistência.
- Restrições ao domínio são a forma mais elementar de restrição de integridade.
- Testam condições sobre valores a introduzir em atributos. Fazem-no, restringindo o domínio do atributo em causa.
- No momento em que se define a tabela.
- A forma mais comum de restrição ao domínio é a proibição do valor `NULL` (`NOT NULL` depois do atributo).
- Podem-se impor outras restrições usando, depois dum nome de atributo A, `CHECK(condição(`A_1`,...,` A_n`))` onde `condição(...)` denota uma condição imposta sobre os atributos A_1,...,A_n.

Exemplos de restrições ao domínio

```
create table alunos(
     num_aluno number(6) not null,
     nome varchar(30) not null,
     local varchar(25),
     data_nsc date not null,
     sexo char(1) not null check (sexo in ('F','M')),
     cod_curso number(3) not null);
create table produtos(
     id_produto number(6) not null,
     nome varchar2(30) not null,
     iva number(2) not null check (iva in (5,12,21)));
create table balcao(balcaoNome char(15),
     balcaoCidade char(30),
     depositos  number check (depositos >= 0));
```


Integridade Referencial

Chaves Externas (também designadas por "estrangeiras")

- Garante que um valor que ocorre numa relação para um certo conjunto de atributos também ocorre num outro conjunto de atributos de outra relação.
  Exemplo: Se "Coimbra-central" é o nome de uma agência que ocorre num dos tuplos da relação `Conta`, então existe um tuplo na relação `Balcao` para o balcão "Coimbra-central".
- Definição Formal:
  Sejam $r_1(R_1)$ e $r_2(R_2)$ duas relações com chaves primárias K_1 e K_2 respectivamente.
  O subconjunto α de R_2 é uma chave externa referindo K_1 na relação r_1, se para todo t_2 em r_2 existe um tuplo t_1 em r_1 tal que $t_1[K_1] = t_2[\alpha]$.

Aquando da escrita do modelo relacional as chaves externas são distinguidas colocando-as a sublinhado a tracejado.

Modificação da Base de Dados

Os testes abaixo devem ser efectuados de modo a preservar-se a seguinte restrição de integridade referencial:

$$\Pi_{\alpha}(r2) \subseteq \Pi_{K}(r1)$$

Inserção: se um tuplo t_2 é inserido em r_2, o sistema tem de garantir que existe um tuplo t_1 em r_1 tal que $t_1[K] = t_2[\alpha]$.
Ou seja

$$t_2[\alpha] \in \Pi_{K}(r1)$$

Remoção: se um tuplo t_1 é removido de r_1, o sistema deve calcular o conjunto de tuplos em r_2 que referenciam t_1:

$$\sigma_{\alpha=t_1[K]}(r_2)$$

Se este conjunto não é vazio, ou o comando de remoção é rejeitado, ou os tuplos que referenciam t_1 devem ser eles próprios removidos (remoções em cascata são possíveis).

Modificação da Base de Dados (Cont.)

Actualização. Existem duas situações:
Se um tuplo t_2 é actualizado na relação r_2 em que é modificado o valor da chave externa α, então é efectuado um teste similar ao da inserção. Seja $t_{2'}$ o novo valor do tuplo t_2. O sistema deve garantir que

$$t_{2'}[\alpha] \in \Pi_K(r1)$$

Se o tuplo t_1 é actualizado em r_1, e a operação altera o valor da chave primária (K), então é efectuado um teste semelhante ao da remoção. O sistema deve calcular

$$\sigma_{\alpha=t_1[K]}(r_2)$$

usando o valor anterior de t_1 (o valor antes da actualização). Se o conjunto é não vazio, a actualização pode ser rejeitado, ou a actualização pode ser propagada em cascata, ou os tuplos podem ser removidos.

Integridade Referencial em SQL

As chaves **primárias**, **candidatas** e **externas** podem ser especificadas na instrução SQL `create table`:

- A cláusula `primary key` inclui a lista dos atributos que formam a chave primária.
- A cláusula `unique key` inclui a lista dos atributos que formam a chave candidata.
- A cláusula `foreign key` inclui a lista de atributos que constituem a chave externa assim como o nome da relação referenciada pela chave externa.

Integridade Referencial em SQL — Exemplo

```
CREATE TABLE cursos(cod_curso number(3) NOT NULL,
          nome varchar(35) NOT NULL,
          PRIMARY KEY (cod_curso));

CREATE TABLE cadeiras(cod_cadeira number(3) NOT NULL,...,
          PRIMARY KEY (cod_cadeira));

CREATE TABLE curso_cadeira(cod_curso number(3) NOT NULL,
          cod_cadeira number(3) NOT NULL,
          semestre number(2) NOT NULL,
          PRIMARY KEY (cod_curso, cod_cadeira),
          FOREIGN KEY (cod_curso) REFERENCES cursos,
          FOREIGN KEY (cod_cadeira) REFERENCES cadeiras);
```


Acções em Cascata em SQL

```
CREATE TABLE Conta
  ...
  FOREIGN KEY (balcaoNome) REFERENCES Balcao
                           ON DELETE CASCADE
                           ON UPDATE CASCADE
  ...)
```

- Com as cláusulas `ON DELETE CASCADE`, se a remoção de um tuplo na relação `Balcao` resulta na violação da restrição da integridade referencial, a remoção propaga-se em "cascata" para a relação `Conta`, removendo o tuplo que referia a agência que tinha sido eliminada.
- Actualizações em cascata são semelhantes.

Acções em cascata em SQL (cont.)

- Se existe uma cadeia de dependências de chaves externas através de várias relações, com um `ON DELETE CASCADE` especificado em cada dependência, uma remoção ou actualização num dos extremos pode-se propagar através de toda a cadeia.
- Se uma remoção ou actualização em cascata origina uma violação de uma restrição que não pode ser tratada por uma outra operação em cascata, o sistema aborta a transacção. Como resultado, todas as alterações provocadas pela transacção e respectivas acções em cascata serão anuladas.
- A integridade referencial é verificada apenas no final da transacção
  - Passos intermédios podem violar a integridade referencial desde que passos posteriores a reponham.
  - Caso contrário seria impossível criar alguns estados da base de dados, por exemplos, inserir dois tuplos cujas chaves externas apontam um para o outro.

Integridade Referencial em SQL (cont.)

- Alternativas às operações em cascata:
  - ON DELETE SET NULL
  - ON DELETE SET DEFAULT
- Valores nulos em atributos de chaves externas complicam a semântica de integridade referencial da SQL, devendo-se evitar recorrendo a `NOT NULL`.
- Se algum atributo de uma chave externa é nulo, o tuplo satisfaz automaticamente a restrição de integridade referencial!

Segurança

Segurança — ao contrário das restrições de integridade, que pretendiam proteger a base de dados contra estragos acidentais, a segurança preocupa-se com proteger a base de dados de estragos propositados.

- A nível do sistema operativo
- A nível da rede
- A nível físico
- A nível humano
- A nível da base de dados

Mecanismos de autenticação e autorização para permitir acessos selectivos de (certos) utilizadores a (certas) partes dos dados

Autorizações

Diferentes formas de autorização, em dados da bases de dados:

- Autorização de leitura - permite ler, mas não modificar dados.
- Autorização de inserção - permite inserir novos tuplos, mas não modificar tuplos existentes.
- Autorização de modificação - permite modificar tuplos, mas não apagá-los.
- Autorização de apagar (“delete”) - permite apagar tuplos

Autorizações (Cont.)

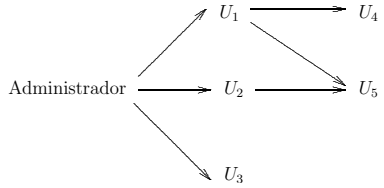
Diferentes formas de autorização, para alterar esquemas:

- Autorização de índex - permite criar e apagar ficheiros de indexação.
- Autorização de recursos - permite criar novas relações.
- Autorização de alteração - permite criar e apagar atributos duma relação.
- Autorização de remoção (“drop”) - permite apagar relações.

Atribuição de Privilégios

- A passagem de privilégios de um utilizador para outro pode ser representado por um grafo de autorizações.
- Os nós do grafo são utilizadores.
- A raiz é o administrador da base de dados.

Por exemplo: no grafo abaixo um arco $U_i \rightarrow U_j$ indica que o utilizador U_i atribuiu ao utilizador U_j um determinado privilégio.



- Todos os arcos têm que fazer parte de algum caminho com origem no administrador.
- Um dado utilizador, que não o administrador, só pode atribuir privilégios a outro utilizador se para isso estiver mandatado, isto é, se tiver o privilégio de atribuir privilégios.
- O comando `GRANT` é o comando usado para atribuir privilégios.

DML — SQL

Linguagem a que o utilizador recorre para obter informação a partir da base de dados.

- Estrutura básica
- Operações com conjuntos
- Funções de agregação
- Valores nulos
- Junções
- Sub-consultas embebidas
- Relações derivadas
- Modificação da Base de Dados

Esquema Utilizado em Exemplos