Ficha 6 – SQL (alterações e triggers)

Bases de Dados, FCT-NOVA

Ano letivo 2015/16

Grupo 1. Na base de dados com que temos vindo a trabalhar não são guardadas informações sobre os funcionários de cada um dos departamentos. Vamos alterar a base de dados por forma a suprir essa falta.

De cada um dos funcionários não docentes dum departamento queremos, para já, guardar apenas informação do seu nome, sexo (que só pode ser `M` ou `F`), e data em que entrou ao serviço do departamento. Para além disso, cada funcionário deverá ter um número interno. Mas não estamos muito preocupados com qual o número atribuído a cada funcionário. Se o sistema tratasse disso, por nós tanto melhor. Até podia atribuir como código um número (único para cada funcionário) sequencial.

1. Crie na base de dados a tabela de funcionários, com os atributos como descrito acima.
2. Defina sobre essa tabelas as restrições que achar apropriado (não esquecendo chave primária e chaves estrangeiras).
3. Crie um sequência SQL para atribuir os códigos de funcionário automaticamente.
   Para criar uma sequência, utilize o comando:

   ```
   CREATE SEQUENCE nome_seq
   START WITH valor
   INCREMENT BY passo;
   ```

   O valor corrente de uma expressão pode ser obtido com a pseudo-coluna `CURRVAL`. O valor seguinte com a pseudo-coluna `NEXTVAL`.
4. Introduza alguns funcionários na base de dados. Para testar as restrições tente inserir um funcionário com sexo `'N'` e um funcionário do departamento com código `26`.

Grupo 2. A base de dados também não permite armazenar associações entre docentes e cadeiras. E, como sabemos, há vários tipos de associações que se podem estabelecer. Nomeadamente, a base de dados não permite guardar a informação sobre quem é o docente responsável por cada uma das cadeiras, nem sobre quem são os docentes que lecionam cada uma das cadeiras.

Neste grupo vamos alterar a base de dados para passar a permitir guardar esta informação.

1. Crie uma tabela `lecionacao` que irá armazenar a informação sobre que docentes lecionam que cadeiras num dado momento. Não se esqueça de, sobre essa tabela, definir restrições de integridade apropriadas. Estas restrições devem ser de tal forma que não seja permitido dizer que um docente leciona uma cadeira que não exista na base de dados, nem dizer que uma cadeira é lecionada por um docente que não exista na base de dados. Teste, introduzindo alguns dados.
2. Modifique a base de dados por forma a que passe a ser possível armazenar a informação sobre quem é o docente responsável por cada uma das cadeiras (onde cada cadeira só pode ter um responsável, mas um docente pode ser responsável por mais que uma cadeira, ou mesmo nenhuma). Note-se que o responsável por uma cadeira não é obrigatoriamente um dos docentes que a lecciona.

3. Introduza, para todas as cadeiras existentes na base de dados, informação sobre quem é o seu responsável. Faça as coisas de forma a que o docente responsável por uma cadeira seja sempre do mesmo departamento que a cadeira.

4. Imponha agora restrições de integridade sobre a base de dados por forma a que não seja possível associar como responsável duma cadeira um docente de um departamento diferente daquele de que faz parte a cadeira. Teste as restrições tentando dizer que o docente cujo código é `4` (que é do Dep. de Matemática) é responsável pela cadeira de `Bases de Dados`.

Grupo 3. Olhemos agora para a forma como estão guardadas a categoria atual de cada docente, e as várias categorias pelas quais os docentes passaram. Não faz lá muito sentido, pois não?!?

Note que não há na base de dados nada que relacione a categoria atual com o histórico! Por exemplo, no histórico relativo ao docente 4, há entradas para as categorias 2, 3 e 4. Mas a categorias atual de 4, de acordo com o que está na relação de docentes é a 2!!

Além disso, guardando-se só uma data em cada tuplo do histórico é impossível saber em que período um docente esteve numa dada categoria, sem assumir que o docente sempre esteve em alguma categoria, e que sempre subiu de uma categoria para a imediatamente a seguir.

Vamos então mudar isto. E o que vos propomos é o seguinte modelo para armazenar a categoria atual e o histórico de categorias dos vários docentes:

- Na relação `docentes` vamos guardar a categoria atual do docente (tal como está agora).
- Na relação de `historico_categorias` vamos guardar, em cada tuplo, uma categoria passada de um docente com referência de qual a categoria, qual o docente, qual a data em que o docente iniciou nessa categoria e qual a data em que deixou de a ter.
- Para a categoria atual do docente existirá sempre um tuplo na relação categorias com valor `null` no atributo da data em que o docente deixou de ter essa categoria.
- A relação `historico_categorias` só vai poder ser modificada indirectamente via alterações da categoria de um docente na relação `docentes`.
  - sempre que se insere um novo docente com uma dada categoria, introduz-se um tuplo no histórico guardando a informação de que o docente tem, desde o dia em que é inserido, essa categoria;
  - sempre que se altera o atributo da categoria atual do docente, há que, no histórico, guardar a informação de que o docente passa a ter a nova categoria a partir da dada de alteração, e de que desde a véspera deixou de ter a categoria que tinha;
  - sempre que se apaga um docente (só da base de dados, claro!), apagam-se os elementos a ele referentes no histórico.

Se quisermos fazer isto na base de dados que temos, há ainda outro problema: o que fazer aos dados que já lá estão. O que se propõe é que se aproveitem os dados que já lá estão e se convertam estes dados para o novo formato. Para isso há que assumir algumas coisas sobre os dados que lá estão. Assuma então que:

- Os dados mais corretos são os que estão no histórico de categorias (i.e. as categorias que estão atualmente na relação `docentes` devem ser ignoradas).
- A data atualmente no histórico refere-se à data em que o docente iniciou funções nessa categoria. Havendo categoria subsequente para esse docente, assume-se que a data de termo de funções na categoria anterior é a véspera.
- A progressão de todos os docentes deu-se sem interrupções de funções do docente na casa, e sempre de uma categoria para a outra imediatamente superior

Vamos então a isso. Mas uma coisa de cada vez.

1. Comecemos por adicionar no histórico o atributo, `data_fim`, para a data de cessação de funções. Introduza logo uma restrição que garante que esta data é sempre maior que a data de início de funções.

2. Para ajudar no preenchimento do novo atributo, comecemos por criar uma view que nos dê as mudanças de categorias de docentes. Mais concretamente, crie uma view que, para cada docente que alguma vez mudou de categoria dê: o código do docente, o código da categoria (que deixou de ter), a data em que iniciou funções nessa categoria e a data em que cessou funções nessa categoria (véspera da data em que assumiu funções na categoria seguinte).

3. Preencha agora correctamente o novo atributo do histórico (usar a view da alínea anterior deve ajudar).

4. Atualize os valores do atributo da categoria dos docentes, com a última categoria que têm no histórico.

5. Crie um trigger que, sempre que se insere um novo docente, introduza no histórico a informação correspondente. Teste esse trigger introduzindo um novo docente (e.g. João Nunes, Prof. Catedrático do Dep. de Informática) e vendo depois o que ficou no histórico.

6. Crie um trigger que garanta que sempre que se apaga um docente, se apagam do histórico os tuplos correspondentes. Verifique, apagando o docente que acabou de introduzir.

7. Finalmente, crie um trigger que, sempre que se altera o atributo com a categoria de um docente, altere a relação do histórico por forma a manter a consistência dos dados, i.e. que coloque a data de cessação de funções na categoria anterior e crie um novo tuplo para a nova categoria. Teste o trigger promovendo os docentes 1 e 2 à categoria seguinte.