



UNIDADE CURRICULAR: Fundamentos de Bases de Dados

CÓDIGO: 21053

DOCENTE: Paulo Pombinho

A preencher pelo estudante

NOME: Sérgio Bruno Alves Neto

N.º DE ESTUDANTE: 2304357

CURSO: Licenciatura em Engenharia Informática

TRABALHO / RESOLUÇÃO:

1.

a) **Quais as tabelas existentes e a função principal de cada uma.**

Após visualizar a imagem, confirmo que existem 5 tabelas, com as seguintes identificações: **person**, **movie**, **genre**, **role** e **movie_gener**. Passando a descreve a função principal de cada uma, temos: as tabelas **person**, **movie**, **genre**, são tabela de entidade onde armazenam informações sobre as pessoas envolvidas nos filmes, isto é, o que define cada filme, temos (título, datas, valores, entre outros), além disso ainda armazena diferentes géneros cinematográficos.

Já as tabelas: **role** e **movie_gener**, são tabelas que estão associadas às outras tabelas, neste exemplo temos que **role** faz ligação entre pessoas e filmes, indicando quais as pessoas que participaram em cada filme e qual o seu papel, se olharmos para o final dessa tabela encontramos o (role type).

Para finalizar as ligações são feitas de muitos para muitos, entre as tabelas **role** e **movie_gener**, uma vez que existe vários filmes, vários géneros, da mesma forma **movie** e **genre** têm ligações de muitos para muitos.

b) Quais as chaves primárias e estrangeiras, o que significam e porque são necessárias.

Respondendo à questão, comecemos pelas chaves primárias, temos na:

→ Tabela **person**, a chave primária é **person_id**. Chave esta que identifica cada pessoa individualmente registada (ator, realização, entre outras).

→ Tabela **movie**, a chave primária é **movie_id**, serve para identificar cada filme individualmente.

→ Tabela **genre**, a chave primária é **genre_id**, serve para identifica de forma única cada género cinematográfico (comédia, terror, suspense, entre outros).

→ Tabela **role**, aqui existem 3 chaves primárias, temos a **movie_id**, **person_id** e **role_type**. Neste caso, temos que em cada linha irá representar o papel que cada pessoa desempenhou em cada filme, daí que este conjunto é que irá distinguir cada participação do papel desempenhado por cada pessoa por cada filme, ou até no mesmo filme.

→ Tabela **movie_gener**, aqui só temos 2 chaves primárias, a **movie_id** e **genre_id**. Nesta tabela iremos ter em cada linha a associação de cada filme com o seu género, não esquecendo que esta associação terá de ser única.

Resumindo, sem chaves primárias, tornar-se-ia quase ou mesmo impossível de garantir que existiria registos em duplicados, existindo, torna-se tudo muito mais simples e eficaz.

Passemos agora às chaves estrangeiras, no referido diagrama temos, na:

→ Tabela **role**, temos 2 chaves primárias, a 1ª é: **movie_id** que se referencia com a tabela **movie(movie_id)**. Esta ligação garante que só irá existir papeis associados aos filmes que existem nesta última tabela. A 2ª chave primária é: **person_id**, que garante que só irá existir papeis associados as pessoas que existem na tabela **person**.

→ Tabela **movie_gener**, temos as 2 chaves primárias acima já referidas, em que **movie_id** faz referência à tabela **movie(movie_id)** e **genre_id** referencia-se com a tabela **genre(genre_id)**, ambas as chaves garantem que só existiram registos associados entre filme-genero para os filmes géneros existentes na tabela.

Resumindo, as chaves estrangeiras são atributos que servem para ligarem entre si ou outras tabelas, ao qual obrigatoriamente fará sempre ligação a algo existente, evitando desta forma que existe dados sem fim ou início.

Para concluir a resposta, passo a explicar o porquê que são necessárias, primeiramente as chaves primárias são importantes para: identificar cada registo, evitar que existam registos em duplicado e por fim, servir de referência a partir de outras tabelas. Já as chaves estrangeiras são necessárias para: representar as relações que possam existir entre entidades (baseando neste diagrama, podemos dar como exemplo, pessoal que participaram em filmes, ou até cada tipo de filme pertence a que género, além disso, garantem coerência e a consistência dos dados registados, de forma que não seja possível existir por exemplo um **movie_id** em **role** se esse filme não existir também em **movie**, por fim ajudam a suportar o tipo de ligações que possam existir entre tabelas, 1 para N e/ou N para N.

c) Que cardinalidades se verificam entre as tabelas (1:N, N:M, etc.) e qual o seu significado.

Baseando-me no diagrama fornecido, a cardinalidade lógica existente entre **person** e **movie**, temos uma ligação de N para N, isto porquê? Uma pessoa pode estar em vários filmes, e em cada filme ter várias pessoas associadas. Se olharmos para a tabela **role**, iremos ter uma ligação de 1 para N, porque uma pessoa pode ter vários registos nesta tabela, mas em contrapartida cada registo desta tabela irá estar ligado a uma só pessoa. Um filme pode ter muitos registos, mas cada esse registo refere-se unicamente a um filme. Resumidamente, 1 para N, significa que do lado 1, só temos um registo, enquanto, que no lado N esse registo pode estar ligados a vários registos.

2.

a) Liste os 5 filmes mais recentes, com estado “Released”. Mostre as colunas title, release_date e status.

Aqui fica o resultado obtido (o ficheiro com o código será enviado separadamente).

	title	release_date	status
►	Sin City: A Dame to Kill For	2014-08-20	Released
	Harry Potter and the Half-Blood Prince	2009-07-07	Released
	Terminator Salvation	2009-05-20	Released
	The Dark Knight	2008-07-16	Released
	Indiana Jones and the Kingdom of the Crystal S...	2008-05-21	Released

b) Liste, para cada original_language, o número total de filmes.

Ordene, de forma descendente, por número total de filmes e de forma ascendente por original_language.

Aqui fica o resultado obtido (o ficheiro com o código será enviado separadamente).

	original_language	total_filmes
▶	en	463
	de	8
	es	6
	ja	5
	zh	4
	it	3
	fr	2
	ko	2
	pt	2
	af	1
	da	1
	hi	1
	ru	1
	sv	1

c) Liste person.name e movie.title das pessoas que aparecem em ambos os papéis (role_type='cast' e role_type='crew') no mesmo filme. Adicionalmente filtre de forma que os filmes em questão tenham, obrigatoriamente, a palavra “wars” no título. Ordene, de forma ascendente, por person.name e movie.title.

Aqui fica o resultado obtido (o ficheiro com o código será enviado separadamente).

	person_name	movie_title
▶	Phil Tippett	Star Wars
	Rick McCallum	Star Wars

d) Liste todos os géneros (genre_name) que têm mais de 100 filmes com status = 'Released'. Apresente também o número total de filmes (total_filmes) e ordene por esse número DESC. Nota: Deve obrigatoriamente usar HAVING.

Aqui fica o resultado obtido (o ficheiro com o código será enviado separadamente).

	genre_name	total_filmes
▶	Drama	251
	Thriller	149
	Action	139
	Adventure	128
	Comedy	113
	Crime	103

3.

a) A query não executa corretamente no MySQL e não devolve o resultado pretendido. Explique qual o erro principal e porquê.

Olhando para o código apresentado, automaticamente verificamos que a query está incorreta, porque usa **GROUP BY original_language** mas, se olharmos para a lista de seleção, inclui a coluna **title**, que não está no group by, nem numa função de agregação.

Como sabemos em SQL (no MySQL com o modo **only_full_groups_by** ativo, todas as colunas do **Select** que não estejam agregadas terão de permanecer no **group by**.

Neste caso específico, existe muitos títulos diferentes para cada **original_language**, logo p SGBD não sabe qual o valor de **title** que deve mostrar para cada língua, torando o resultado duvidoso.

Pelo indicando acima, esta query nunca iria executar corretamente, mas, vamos supor que corria, não iria devolver o resultado pretendido (a duração média por língua), mas sim um título arbitrário por cada língua.

b) Explique de que forma deveria corrigir a query de forma que funcione corretamente e devolva o resultado pretendido.

Para que funcione corretamente a query, e devolva o que realmente é pretendido (duração média dos filmes e por língua), seria necessário eliminar a coluna **title** da lista de seleção, mantendo apenas a língua e a duração média. Desta forma o resultado pretendido seria uma linha por **original_language** e não por filme, desta forma, todas as colunas do **Select** ficariam compatíveis com o **Group by**. A forma mais correta de escrever a query seria:

```
SELECT original_language, AVG(runtime) AS avg_runtime  
FROM movie  
WHERE vote_count >= 50  
GROUP BY original_language  
ORDER BY avg_runtime DESC;
```

Desta forma o SGBD iria calcular a média (**runtime**) apenas para os filmes com **vote_count >=50** e ficariam agrupados por língua, e iria ordenar o resultado pela duração média de forma decrescente, como era inicialmente pretendido.

Concluído. Sérgio Neto, estudante n.º 2304357.