E-fólio B | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Fundamentos de Bases de Dados

CÓDIGO: 21053

DOCENTE: Paulo Pombinho

A preencher pelo estudante

NOME: Vítor Manuel Metrogos Frango

N.º DE ESTUDANTE: 1802925

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 07 de janeiro de 2024

TRABALHO / RESOLUÇÃO:

Q1.

a) A expressão em álgebra relacional abaixo descrita

$\Pi_{id,\ nome}$

É a projeção que seleciona exclusivamente as colunas 'id' e 'nome' da tabela, neste caso em particular 'funcionario'

$\sigma_{cargo='analista' \vee (salario \geq 2500 \wedge departamento='financeiro')}(funcionario)$

È a seleção que filtra as linhas da tabela 'funcionario' com base em duas condições onde 'cargo' é 'analista' ou ('salario' é igual ou superior a '2500' e 'departamento' é 'financeiro'.

Consulta em SQL

```
SELECT id, nome
FROM funcionario
WHERE cargo = 'analista' OR (salario >= 2500 AND departamento =
'financeiro');
```

Devolve ao user os funcionários que são analistas ou que trabalham no departamento financeiro com um salario de pelo menos 2500, em que:

`FROM funcionario`: Indica que a consulta será realizada na tabela 'funcionario'

`WHERE cargo = 'analista'`: filtra os funcionários cujo 'cargo' é igual a 'analista'

`OR (salario >= 2500 AND departamento = 'financeiro'`: inclui também os funcionários que ganham um salario maior ou igual a 2500 e trabalham no departamento 'financeiro'

b) Para a consulta em SQL

```
SELECT nome, departamento
FROM funcionario
WHERE salario > (SELECT AVG(salario) FROM funcionario);
```

Iremos obter a equivalente expressão em álgebra relacional,

$\Pi_{nome,departamento}(\sigma_{salario>\tau AVG(salario)}(funcionário)$ que se traduz em:

Projeção(π): Operador que seleciona colunas especificas de uma tabela

Seleção(σ): Operador que filtra as linhas de uma tabela baseado em determinada premissa

Por sua vez a subconsulta calcula a média dos salários de todos os funcionários.

Detalhadamente teremos:

$\Pi_{nome,\ departamento}$**:** Seleciona as colunas 'nome' e 'departamento'

$\sigma_{salario>\tau AVG(salario)}(funcionário)$**:** Filtra as linhas da tabela 'funcionario' onde 'salario' é maior que a média dos salários na tabela 'funcionario'

Em conclusão a expressão em álgebra relacional, $\Pi_{nome,departamento}(\sigma_{salario>\tau AVG(salario)}(funcionário)$ devolve ao utilizador todos os nomes e departamentos em cujo o salario é superior à media salarial de todos os funcionários da tabela 'funcionario'

Q2.

a) Para a implementação de uma estrutura de segurança numa base de dados deveremos, e para o solicitado no enunciado, seguir os passos:

- Definição de papeis e permissões:
  - ‘GerenteAgencia’ que pode incluir permissões para aceder e modificar dados de clientes da agência a qual reporta, realizar transações, aprovar ou reprovar empréstimos de valores residuais ou mais baixos e gerar ou aceder aos relatórios financeiros da sua agência
  - ‘DiretorRegional’ neste caso as permissões são mais amplas visto ter abaixo da sua hierarquia diversas agências. Desta forma terá acesso a informação de várias agências na sua região, aprovação de empréstimos de maior valor (acima do valor que é aprovado para o GerenteAgencia), acesso a relatórios financeiros regionais (escalonados pelos diversos GerenteAgencia) e capacidade e realizar auditorias as agencias que lhe reportam.
- Implementação na Base de Dados:

Criar os papeis:

```
CREATE ROLE 'DiretorAgencia'
CREATE Role 'DiretorRegional'
```

Cria os diferentes papeis (ROLE)

Atribuição de permissões:

```
GRANT SELECT INSERT ON clientes TO DiretorAgencia;
GRANT SELECT INSERT ON relatorios TO DiretorRegional;

GRANT SELECT UPDATE ON clientes TO DiretorAgencia;
GRANT SELECT UPDATE ON relatorios TO DiretorRegional;
```

Insere ou atualiza os privilégios nos diferentes papeis, para os casos de ainda não existirem ou de já terem sido criado previamente

Associação de utilizadores aos diferentes papeis

```
ALTER ROLE GerenteAgencia ADD MEMBER ZeDosDados;
ALTER ROLE GerenteRegional ADD MEMBER PresidenteDasAgencias;
```

Adiciona aos utilizadores ZeDosDados os privilégios de GerenteAgencia e a PresidenteDasAgencias os privilégios de GerenteRegional

Após da criação de papeis, atribuição de permissões e associação de utilizadores ao diferentes papeis, há aspetos de enorme importância na segurança de uma base de dados, como por exemplo; segurança/auditoria, testes/validação e manutenção/atualização

b) Brevemente respondida na alínea anterior aquando da implementação, mas que irei aprofundar um pouco mais, visto ter de ser efetuado através de dois comando diferente

Dá ao ‘GerenteAgencia’acesso à tabela ‘clientes’

```
GRANT SELECT ON clientes TO GerenteAgencia;
```

Restringe o acesso à tabela ‘informações financieras’

```
REVOKE ALL ON InformacoesFinanceiras FROM GerenteAgencia;
```

Remove todas as permissões que o GerenteAgencia possa ter sobre a tabela ‘InformacoesFinanceiras’

c)

```
GRANT SELECT, INSERT ON Agencias TO DiretorRegional;
GRANT SELECT, INSERT ON Funcionarios TO DiretorRegional;

GRANT SELECT, UPDATE ON Agencias TO DiretorRegional;
GRANT SELECT, UPDATE ON Funcionarios TO DiretorRegional;
```

Atraves destas linhas podemos conceder as permissões solicitadas pelo enunciado sendo que INSERT insere inicialmente a permissão e o UPDATE altera a permissão caso já exista uma outra que não a solicitada

Para restingir ao DiretorRegional a edição na tabela ‘salarios’ , deveremos utilizar

```
GRANT SELECT ON Salarios TO DiretorRegional;
```

d)

Por forma a conceder ao DiretorRegional a permissão de atribuir o papel de DiretorAgencia a outros utilizadores podemos utilizar

```
GRANT GerenteAgencia TO DiretorRegional WITH ADMIN OPTION
```

O GRANT já utilizado nas alíneas anteriores permite conceder privilégios a um determinado papel ou utilizador, enquanto o parâmetro WITH ADMIN OPTION é utilizado para conceder a um papel ou utilizador o mesmo papel a outros utilizadores e gerir os seus privilégios

Q3.

Para chegar ao solicitado no enunciado e efetuar a normalização até à terceira forma normal (3FN) da tabela apresentada, irei por exclusão de parte identificar em que forma normal se encontra ou se pelo contrário não esta em nenhuma e após essa verificação desenhar a tabela desde 1FN ate 3FN passando pela 2FN

<u>Para que possa estar na 1FN a tabela deve:</u>

1. Possuir atributos atómicos, ou seja, não ter grupos repetidos ou atributos multivalorados
2. Possuir valores simples, cada coluna contém valores únicos não listas ou conjuntos

Facto: A coluna ‘Produtos’ contem uma lista de produtos com atributos que não são atómicos, logo, é uma violação dos critérios para esta na 1FN

Solução para normalização para 1FN

1. Dividir a coluna ‘Produtos’ em várias linhas mantendo uma linha para cada produto do Id_Encomenda
2. Aplicar as alterações em que cada linha terá um único produto por encomenda cumprindo dessa forma a 1FN

ID_Encomenda (PK)	Data_Encomenda
E001	2023-03-10
E002	2023-03-11
E003	2023-03-12

Tabela Encomenda

ID_Encomenda (PK)	ID_Porduto (PK)	Nome_Produto	Quantidade	Preco_Unitario
E001	P01	T-Shirt	2	15.00
E001	P02	Calças	1	40.00
E002	P03	Casaco	1	60.00
E003	P01	T-Shirt	3	15.00
E003	P04	Sapatos	1	80.00

Tabela Encomendas Produtos

Para que possa estar na 2FN a tabela deve:

1 – Estar na 1FN e todos os atributos não chave devem ser dependentes de toda a chave primaria e não somente de parte dela

Facto: Na tabela (alterada para 1FN) e assumindo que ID_Encomenda seja a chave primaria (PK) e que cada ID de produto é único por encomenda, a tabela estará portanto na 2FN visto não existirem chaves compostas. Ou, como ID_Encomenda determina Nome_Produto e Preco_Unitario e assumindo que ID_Encomenda e ID_Produto juntos formam a chave primaria (PK) da tabela Encomendas_Produto (representada mais acima) reforça que esta na 2FN

Para que possa estar na 3FN a tabela deve:

1 - Estar na 2FN e nenhuma coluna não chave depender de outra chave, ou seja, devem depender apenas da chave primaria

Facto: ID_Produto (Nome_Produto, Preco_Unitario) determina o nome e o preço de um produto é determinada pela encomenda especifica e pelo produto. ID_Encomenda + ID_Produto (Quantidade), a quantidade do produto é determinada pela encomenda especifica e pelo produto.
Desta forma e como o nome do produto e o preço unitário são determinados pelo ID do produto e não pela chave primaria da encomenda, temos uma violação da normalização 3FN

Solução para normalização para 3FN

1 – Criação de uma nova tabela ‘Produtos’ com os campos ‘ID_Produto’, ‘Nome_Prod§uto’ e ‘Preco_Unitario’

2 – A tabela Encomenda terá os campos ‘ID_Encomenda’, ‘ID_Produto’ e ‘Quantidade’

3 – Remover ‘Nome_Produto’ e ‘Preco_Unitario’ da tabela ‘Encomenda’ e utilizar um JOIN com a tabela ‘Produtos’ quando necessitar dessas informações.

ID_Encomenda (PK)	Data_Encomenda
E001	2023-03-10
E002	2023-03-11
E003	2023-03-12

Tabela Encomendas – Possui informação sobre as encomendas

ID_Porduto (PK)	Nome_Produto	Preco_Unitario
P01	T-Shirt	15.00
P02	Calças	40.00
P03	Casaco	60.00
P04	Sapatos	80.00

Tabela Produtos – Possui informação sobre produtos

ID_Encomenda (PK)	ID_Porduto (PK)	Quantidade
E001	P01	2
E001	P02	1
E002	P03	1
E003	P01	3
E003	P04	1

Tabela Encomenda Produtos – Relaciona as encomendas com produtos e a quantidade de cada produto encomendado

Desta forma obtemos as 3 tabelas normalizadas 3FN em que as informações estão armazenadas de forma eficiente sem a existência de redundâncias e dependências que podem gerar anomalias de introdução, atualização e eliminação

Q4.

De forma a ter a resposta mais organizada irei detalha la em três partes.

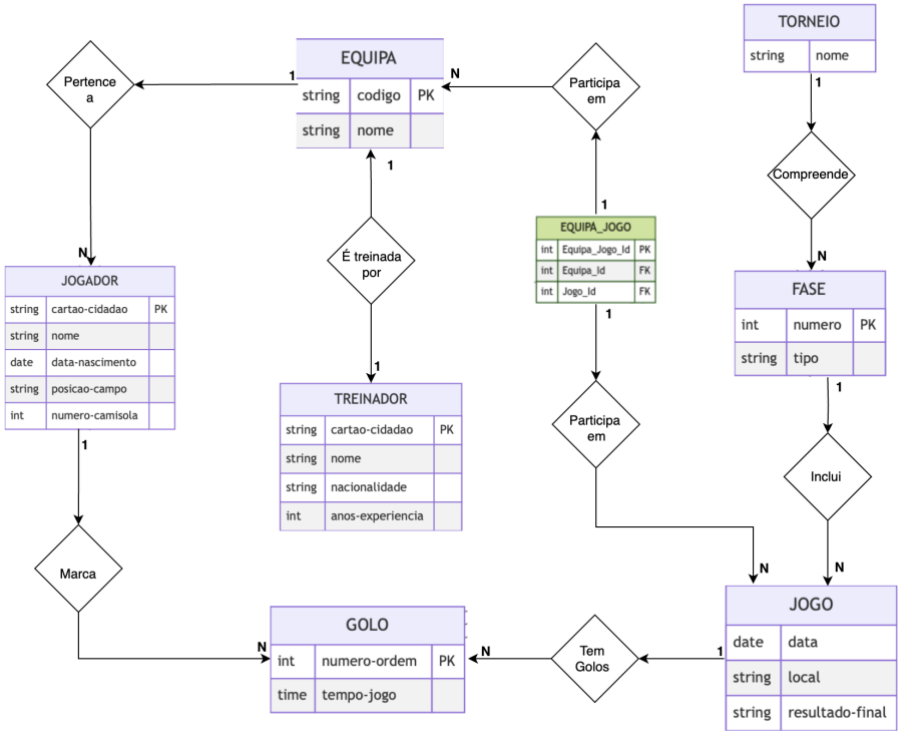
Em primeiro ligar irei identificar as principais entidades e os seus atributos:

1. Equipa
    a. Equipa_Id (chave primaria)
    b. Nome
    c. Jogadores (relação com entidade Jogador)
    d. Treinador (relação com entidade Treinador)
2. Jogador
    a. Cartão de cidadão (chave primaria)
    b. Nome
    c. Data de Nascimento
    d. Posição em campo
    e. Numero da Camisola
    f. Equipa (Relação com entidade Equipa)
3. Treinador
    a. Cartão de cidadão (chave primaria)
    b. Nome
    c. Nacionalidade
    d. Anos de experiência
    e. Equipa (Relação com a entidade Equipa)
4. Torneio
    a. Fases (Relação com a entidade Fase)
    b. Jogos (Relação com a entidade Jogo)
5. Fase
    a. Número (chave primaria)
    b. Tipo
    c. Jogos (Relação com a entidade Jogo)
6. Jogo
    a. Jogo_Id (chave primaria)
    b. Data
    c. Local
    d. Resultado Final
    e. Equipas Participantes (Relação com a entidade Equipa)
    f. Golos (Relação com a entidade Golo)
    g. Fase (Relação com entidade Fase)
7. Golo
    a. Numero de Ordem (chave primaria em cada Jogo)
    b. Jogador (Relação com entidade Jogador)
    c. Tempo de Jogo
    d. Jogo (Relação com a entidade Jogo)
8. Equipa_Jogo
    a. Equipa_Jogo_Id (chave primaria)
    b. Equipa_Id (chave primaria)
    c. Jogo_Id (chave primaria)

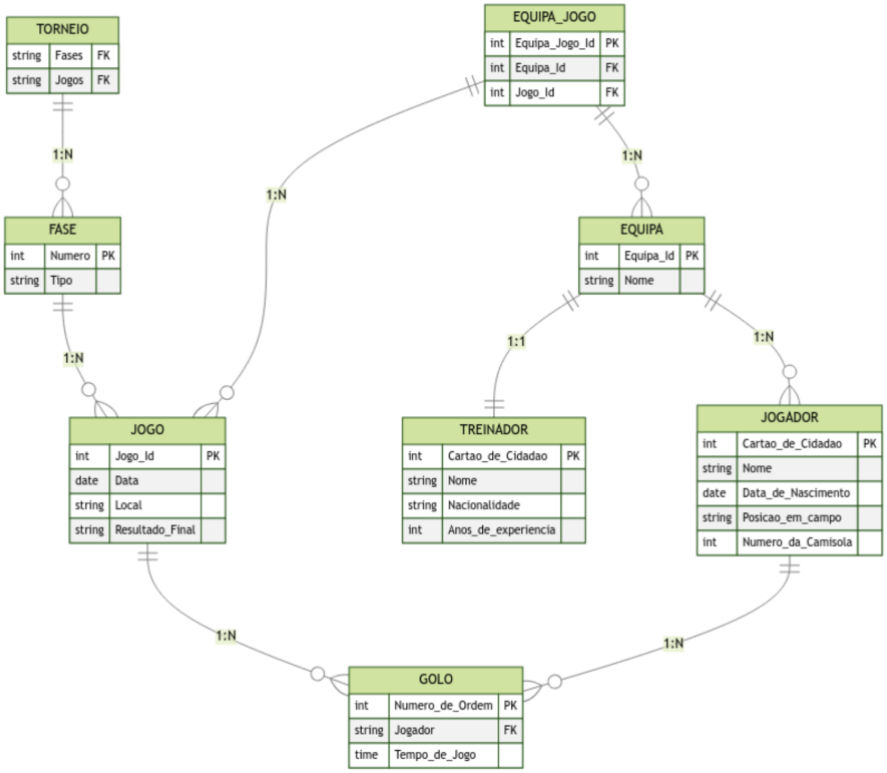
Para os relacionamentos entre entidades do Modelo Entidade-Relação e cardinalidade entre eles, vamos ter:

1. Equipa-Jogador (Pertence a...)
    a. Cada jogador pertence a uma e apenas uma equipa, enquanto uma equipa é composta por vários jogadores
    b. Cardinalidade: Um-para-muitos (1:N) em que um jogador está associado a uma equipa e uma equipa tem vários jogadores
2. Equipa-Treinador (É Treinada por...)
    a. Cada equipa tem um treinador responsável por gerir o treino dos jogadores e cada treinador é responsável por apenas uma equipa
    b. Cardinalidade: Um-para-um (1:1) um treinador esta associado a uma equipa e uma equipa associada a um treinador
3. Equipa-Jogo (Participa em...)
    a. Uma equipa participa em vários jogos ao longo do torneio e em cada jogo participam 2 equipas
    b. Cardinalidade: Muitos-para-muitos (N:N) uma equipa pode participar em vários jogos e um jogo envolve sempre duas equipas
4. Jogo-Golo (Tem golos...)
    a. Cada jogo pode ter vários golos marcados mas cada golo pertence a um único jogo
    b. Cardinalidade: Um-para-muitos (1:N) um jogo pode ter vários golos mas cada golo esta associado a apenas um jogo
5. Jogador-Golo (Marca...)
    a. Um jogador pode marcar vários golos mas cada golo é marcado por um jogador
    b. Cardinalidade: Um-para-muitos (1:N) um jogador pode marcar vários golos mas cada golo é marcado apenas por um jogador
6. Torneio-Fase (Compreende...)
    a. O torneio é composto por várias fases no entanto cada fase pertence a um único torneio
    b. Cardinalidade: Um-para-muitos (1:N) uma fase pode incluir vários jogos mas cada jogo esta so e so associado a um única fase
7. Fase-Jogo (Inclui...)
    a. Cada fase do torneio inclui vários jogos mas cada jogo é associado a uma única fase
    b. Cardinalidade: Um-para-muitos (1:N) uma fase pode incluir vários jogos mas cada jogo esta associado a uma única fase
8. Equipa_Jogo (tabela associativa...)
    a. Utilizo esta tabela para gerir a relação de N:N entre Equipa e Jogo e dessa forma garantir a normalização 3FN

Desta forma vou agora gerar o seguinte diagrama do Modelo Entidade-Relação



c) Para a base de dados relacional do modelo anterior temos o seguinte modelo



NOTAS:

Utilizei para teste de comandos o DataGrip da JetBrains em base de dados SQLite no entanto os comandos podem variar (ser ou não reconhecidos, por exemplo o GRANT em SQLite) dependendo do sistema que estaremos a utilizar (MySQL, PostgreSQL, SQL Server ou outros)

No diagrama Modelo-Entidade da alínea a) da pergunta 4 a tabela associatica (Equipa_Jogo) está representada de cor diferente para melhor leitura

Bibliografia:

(1) Database System Concepts, Henry F. Korth, Abraham Silberschatz, and S. Sudarshan

(2) "What are ACID Transactions?" - Databricks. https://www.databricks.com/glossary/acid-transactions.

(3) "ACID Model vs BASE Model For Database" - GeeksforGeeks. https://www.geeksforgeeks.org/acid-model-vs-base-model-for-database/.

(4) http://db-book.com/

(5) The Theory of Relational Databases, 1983 David Maier, http://web.cecs.pdx.edu/~maier/TheoryBook/TRD.html

(6) copilot for DataGrip (JetBtains)

(7) Para produção dos diagramas - https://diagrams.helpful.dev/s/s:1B8W7GdX