Questão 1: Dependências funcionais

Para a relação $R(A, B, C, D)$ com as seguintes dependências funcionais:

$$\{A\} \rightarrow \{B\}, \{B, C\} \rightarrow \{A\}, \{D\} \rightarrow \{A\}$$

Encontre todas as chaves candidatas de R.

Listando todas as relações funcionais e seus fechos: A -> AB B -> B C -> C D -> ABD AB -> AB AC -> ABC AD -> ABD	BC -> ABC BD -> ABD CD -> ABCD ABC -> ABC ABD -> ABD ACD -> ABCD BCD -> ABCD ABCD -> ABCD	As superchaves são: CD, ACD, CBD, ABCD Destas, a única superchave irredutível é CD, que é então a única chave candidata deste problema.

Questão 2 (2018-2): Jogo dos vários erros.

Em um projeto de sistema de informação usando banco de dados relacional o banco foi criado com um *script* SQL.

O tempo passou, o sistema está em produção, e uma alteração na estrutura do banco de dados é necessária.

Gerente: “Estagiário, precisa alterar o banco de dados assim, etc”.

Estagiário: “Pode deixar chefe!”

O estagiário alterou o *script* original de criação do banco de dados e rodou, com sucesso, esse novo script de criação. Nem precisou de senha de *root*: usou a conta e senha da própria aplicação, que estavam ali *committed* no repositório mesmo!

Gerente: “Santas máquinas de Turing! Cadê os dados! Sumiu a base!”

Estagiário: “...” ¯_(ツ)_/¯

Ache os erros neste cenário. Para cada erro proponha uma solução. Bonus: como você lidaria com essa situação no papel do gerente?

- O estagiário alterou o script original, o certo é fazer um script de alteração separado.
- O estagiário rodou de novo o script de criação da base de dados, que provavelmente foi o que causou a perda da base toda. Nunca se deve repetir a execução destes scripts em uma base em produção.
- As credenciais de acesso ao banco de dados estavam abertas, nunca se deve adicionar as credenciais ao repositório.
- A aplicação tinha permissões muito elevadas, incluindo a capacidade de apagar toda a base de produção. Devemos conceder às aplicações o mínimo de permissões para a execução do produto.
- Ninguém fez backup antes de alterar uma base de produção, deve-se sempre fazer o backup antes de algo potencialmente catastrófico.
- Não se deve alterar diretamente a base de produção se isso pode causar uma falha extrema no sistema. O correto é clonar a base, testar a alteração, e somente então chavear para a nova base.
- Uma operação tão crítica não deveria ser deixada para um funcionário inexperiente.

A primeira coisa que o gerente deve fazer é registrar o post-mortem do evento, para que esses erros não se repitam. O gerente deveria então investigar esses problemas e descobrir qual a causa de tantos erros.

- Pode ser que não existam procedimentos formais para lidar com a base de dados.
    - Deve-se então construir esse manual de operações para que este erro não se repita, e esse procedimento deve ser testado antes de ser publicado.
- Pode ser que os procedimentos existam mas não foram seguidos corretamente pelos funcionários responsáveis. E.g. falta de backup.
    - Em caso de negligência, este funcionário poderá sofrer sanções administrativas ou encerramento do seu contrato.
    - Caso não tenha havido má-fé, mas sim uma falta de capacidade de trabalho, a empresa deve revisitar seus procedimentos de treinamento e de contratação para garantir que os funcionários são habilitados para a operação.
- O próprio gerente pode ter cometido um erro de julgamento ao deixar uma tarefa crítica na mão de um funcionário pouco capacitado.

Questão 3 (2018-2): MySQL tem dois tipos principais de índice: *hash tables* e *B-trees*.

Hash tables (ou *hashmaps*) são tabelas de *look-up* organizadas pelo valor de *hash* da chave.

B-trees são uma generalização da árvore binária de busca. São árvores de busca com um *branching factor* maior que dois, e são mantidas balanceadas graças a seus algoritmos de inserção e remoção de nós.

Em ambos os casos a chave vem da coluna que está sendo indexada, e o valor armazenado na estrutura de busca é a posição do registro procurado na tabela.

a) Em uma aplicação financeira temos uma tabela com o valor do saldo do cliente e outras informações deste (nome, número da conta, RG, etc). Suponha que desejamos rapidamente buscar clientes por faixa de saldo (e.g. **`SELECT * FROM clientes WHERE saldo >= @saldo_min and saldo <= @saldo_max`**).

Seria uma boa ideia fazer um índice na coluna saldo? Se sim, qual o tipo preferido: *hash table* ou *B-tree*? Explique.

Em princípio esta situação parece ser adequada para o emprego de um índice do tipo B-tree. Contudo, note que a coluna à qual se deseja adicionar um índice é uma coluna onde os dados são alterados com frequência, e isso faz com que o emprego de um índice não seja recomendado.

b) Nesta mesma aplicação financeira e nesta mesma tabela de informações de cliente temos o telefone principal do cliente. Suponha que desejamos frequentemente buscar clientes pelo número de telefone (e.g. **`SELECT * FROM clientes WHERE telefone = @telefone_procurado`**).

Seria uma boa ideia fazer um índice na coluna telefone? Se sim, qual o tipo preferido: *hash table* ou *B-tree*? Explique.

A coluna telefone satisfaz condições para o emprego de um índice:

- É consultada frequentemente
- Os valores não mudam, ou mudam com pouca frequência
- Não temos muitos valores iguais, ou nulos.

Portanto é recomendável criar um índice nesta coluna (desde que já não existam muitos outros índices nesta mesma tabela).
Como a busca frequênte é por valor exato, recomenda-se um índice do tipo hash table.

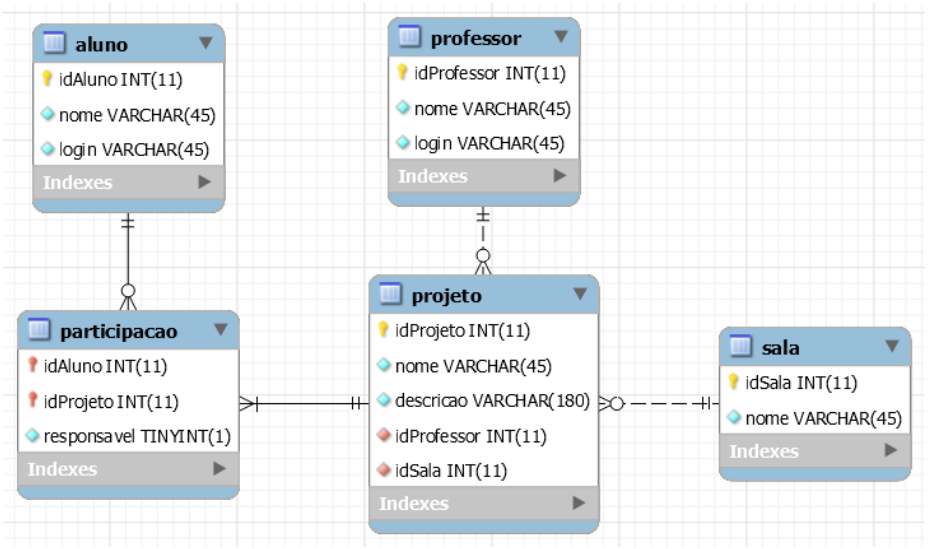
Questão 4 (2018-2):

A Expo Engenharia 2018 vem ai! Será a quarta edição da nossa Expo Engenharia, que está tão grande esse ano que teremos que usar dois andares: o 4º andar e parte do 2º andar (salas 204/205). Vai ser dia 6 de dezembro, uma quinta-feira, das 19h as 22h.

Esse ano vamos precisar de um CRUD para inscrição dos projetos, pois o Google Forms já não estava servindo. Temos os seguintes requisitos:

- O aluno entra no sistema escolhendo seu nome/login em uma lista de alunos. (Isto é uma versão simplificada, na versão real vou também adicionar senha, etc. Ou não – mau professor, mau professor...)
- O aluno cadastra novos projetos. O projeto tem um nome, uma descrição, um professor associado (que conhece o projeto – e.g. o professor da disciplina) e uma sala preferida para exibição (especialmente crítico para os labs).
- O aluno adiciona seus colegas ao projeto. Cada projeto está associado a vários alunos, sendo que pelo menos um deles deve ser declarado responsável pelo projeto.

Eis a primeira versão da modelagem de banco de dados para o projeto:



Dicionário de dados:

Tabela	Campo	Descrição
aluno	nome	Nome do aluno
	login	Login Insper do aluno (para mandar e-mails para <login>@al.insper.edu.br)
professor	nome	Nome do professor
	login	Login Insper do professor (para mandar e-mails para <login>@insper.edu.br)
projeto	nome	Nome do projeto
	descrição	Descrição do projeto
	idProfessor	Professor que conhece o projeto (para levantar questões de segurança)
	idSala	Sala ou laboratório preferencial para exibição do projeto
sala	nome	Nome da sala ou laboratório
participação	idAluno/idProjeto	Conecta aluno com projeto
	responsavel	Booleana que indica que o aluno idAluno é o responsável pelo projeto idProjeto

a) Verdadeiro (V) ou falso (F):

V	Podemos ter alunos sem projeto
F	Podemos ter projeto sem sala
V	O login do professor poderia ser a chave primária da tabela professor
V	O projeto satisfaz a segunda forma normal se consideramos que o nome é indivisível
F	A relação entre participação e aluno é uma relação não-identificadora

Nas questões a seguir, escreva código SQL para as tarefas pedidas.

b) Crie a tabela participação

```
CREATE TABLE participacao (
    idAluno INT NOT NULL,
    idProjeto INT NOT NULL,
    responsável BOOLEAN DEFAULT FALSE,
    PRIMARY KEY (idAluno, idProjeto),
    FOREIGN KEY idAluno REFERENCES aluno(idAluno),
    FOREIGN KEY idProjeto REFERENCES projeto(idProjeto)
);
```

c) Adicione uma coluna para guardar o ano da Expo (pois esse CRUD servirá para os anos seguintes também) e outra coluna para guardar o semestre do projeto (e.g. 1 para Design de Software, 4 para Camada Física da Computação, etc).

```
ALTER TABLE projeto ADD ano INT, ADD semestre INT;
```

d) Liste os alunos que são responsáveis por mais de um projeto.

```
SELECT
    nome
FROM
    aluno
    INNER JOIN participacao USING (idAluno)
WHERE
    responsavel = TRUE
GROUP BY
    idAluno
HAVING
    COUNT(idProjeto) > 1;
```

e) Liste os professores que não estão ligados a nenhum projeto.

```
SELECT DISTINCT
    professor.nome
FROM
    professor
    LEFT OUTER JOIN projeto USING (idProfessor)
WHERE
    idProjeto IS NULL;
```