UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

Projeto - Assessoria de Eventos - Parte 3

Sistema de organização de festas de aniversário infantil e formaturas

SCC0240 - Bases de Dados - Turma 2

Profª Elaine Parros Machado de Sousa

Data de entrega: 25/06/2018

Código entregue no escaninho de César Augusto Lima

Grupo:

César Augusto Lima	Nº 9771525
Flavio Vinicius Vieira Santana	Nº 9866552
Mateus Castilho Leite	Nº 9771550
Vinicius Henrique Borges	Nº 9771546

1. Documentação de alterações

1.1. Correção dos erros apontados na Parte 1

Os erros presentes nas Parte 1 do projeto já foram corrigidos na parte 2.

1.2. Correção dos erros apontados na Parte 2

Na parte 2 do projeto os erros estavam na justificativa de alguns mapeamentos, portanto, conforme instruído pela professora, suas correções não estão presentes nessa parte do projeto.

1.3. Alterações no mapeamento

Por necessidade na implementação da aplicação, houveram algumas alterações no mapeamento. As alterações foram:

- Adição do atributo “Data” na tabela “Contrato”
- Adição de notação de “Not Null” a chaves estrangeiras em diversas tabelas

2. Sobre a implementação

Para a implementação da aplicação que utiliza o banco de dados apresentado nas partes anteriores do projeto foi utilizado o SGBD Oracle e a linguagem de programação escolhida foi Python (versão 3.6.5).

Para o funcionamento correto da aplicação é necessária a extensão do Python chamada “cx_Oracle” e a biblioteca de Oracle Client chamada “Oracle Instant Client”, de preferência na versão 12.2.

Quanto ao sub-esquema da base de dados escolhido para a implementação, escolhemos a festa de formatura (por conter o serviço específico da compra de convites) e decidimos deixar de fora toda a parte que envolvia os fornecedores das festas.

3. Comandos SQL presentes no código

1 - *server.cur.execute('select * from CLIENTE order by CODIGO')*

Exibe a lista de clientes cadastrados ordenados pelo código.

2 - *server.cur.prepare("insert into CLIENTE(CODIGO, NATUREZA, EMAIL, TELEFONE) VALUES(:1, :2, :3, :4)")*

e

server.cur.execute(None, {'1':cod, '2':nat, '3':em, '4':tel})

server.con.commit()

Insere um novo cliente no sistema e efetua um commit.

3 - *server.cur.prepare("delete from CLIENTE where CODIGO = :1")*

e

server.cur.execute(None, {'1':cod})

server.con.commit()

Deleta um cliente pelo seu código.

4 - *server.cur.prepare("update CLIENTE set NATUREZA = :2, EMAIL = :3, TELEFONE = :4 where CODIGO = :1")*

e

server.cur.execute(None, {'1':cod, '2':nat, '3':em, '4':tel})

server.con.commit()

Edita as informações de um cliente.

5 - *server.cur.execute('select * from LOCAL order by CEP, NUMERO')*

Exibe os locais de festa.

6 - *server.cur.prepare('insert into LOCAL(CEP, NUMERO, CAPACIDADE, EMAIL, TELEFONE, NOME) VALUES(:1, :2, :3, :4, :5, :6)')*

e

server.cur.execute(None, {'1':cep1, '2':num, '3':cap, '4':em, '5':tel, '6':nom})
server.con.commit()

Adiciona um novo local de festa.

7 - *server.cur.prepare("delete from LOCAL where CEP = :1 and NUMERO = :2")*

e

server.cur.execute(None, {'1':cep, '2':num})
server.con.commit()

Deleta um local de festa.

8 - *server.cur.prepare("update LOCAL set NOME = :3, CAPACIDADE = :4, EMAIL = :5, TELEFONE = :6 where CEP = :1 and NUMERO = :2")*

e

server.cur.execute(None, {'1':cep1, '2':num, '3':nom, '4':cap, '5':em, '6':tel})
server.con.commit()

Edita as informações de um local de festa.

9 - *server.cur.execute('select * from FUNCIONARIO order by CPF')*

Exibe a lista de funcionários cadastrados ordenados pelo CPF

10 - *server.cur.prepare('insert into FUNCIONARIO(CPF, NOME, EMAIL, TELEFONE, SALARIO, BANCO, AGENCIA, NUMERO, ENDERECO, CARGO) VALUES(:1, :2, :3, :4, :5, :6, :7, :8, :9, :10)')*

e

server.cur.execute(None, {'1':cpf1, '2':nom, '3':em, '4':tel, '5':sal, '6':ban, '7':ag, '8':num, '9':end, '10':car})

Adiciona uma nova tupla na tabela de Funcionário.

11 - *server.cur.prepare("delete from FUNCIONARIO where CPF = :1")*

e

server.cur.execute(None, {'1':cpf})

Deleta um Funcionário do banco de dados através da chave primária CPF.

12 - *server.cur.prepare("update FUNCIONARIO set NOME = :2, EMAIL = :3, TELEFONE = :4, SALARIO = :5, BANCO = :6, AGENCIA = :7, NUMERO = :8, ENDERECO = :9, CARGO = :10 where CPF = :1")*

e

server.cur.execute(None, {'1':cpf, '2':nom, '3':em, '4':tel, '5':sal, '6':ban, '7':ag, '8':num, '9':end, '10':car})

Atualiza a tupla de Funcionário cujo CPF é igual ao informado pelo usuário.

Opções de consulta

Consulta 1 - Arquivo: "Python/query1.py"
"Para cada cliente físico, mostrar o número de festas e o total gasto nos últimos cinco anos, ordenado decrescentemente pelo montante gasto."
Código
"SELECT F.NOME, F.CPF, COUNT(C.FESTA) AS NUM_FESTAS, SUM(C.PRECO) AS TOTAL_FESTAS
FROM CLIENTE_FISICO F JOIN CONTRATO C
ON F.CODIGO=C.CLIENTE
WHERE EXTRACT(YEAR FROM C.DATA) >= EXTRACT(YEAR FROM SYSDATE) - 5
GROUP BY F.CPF, F.NOME
ORDER BY SUM(C.PRECO) DESC"

Consulta 2 - Arquivo: "Python/query2.py"
"Mostrar o número de clientes físicos, o número de clientes jurídicos e a razão entre o número de clientes físicos e o número de clientes jurídicos."
Código
"SELECT NATUREZA, COUNT(NATUREZA)
FROM CLIENTE
GROUP BY NATUREZA;"

Consulta 3 - Arquivo: "Python/query3.py"
"Para cada tipo de funcionário, calcular a média salarial e a quantidade de empregados. Ordenar decrescentemente pela média salarial"
Código
"SELECT CARGO, AVG(SALARIO) AS SALARIO_MEDIO, COUNT(CPF) AS NUM_EMPREGADOS
FROM FUNCIONARIO
GROUP BY CARGO
ORDER BY AVG(SALARIO) DESC;"

Consulta 4 - Arquivo: "Python/query4.py"

“Para cada vendedor, listar a quantidade de vendas executadas nos últimos 30 dias e o preço total das festas vendidas. Ordenar decrescentemente por preço.”

Código

“SELECT F.NOME, F.CPF, COUNT(C.FESTA) AS NUM_FESTAS, SUM(C.PRECO) AS TOTAL_VENDIDO
FROM VENDEDOR V JOIN FUNCIONARIO F
ON V.CPF=F.CPF
LEFT JOIN CONTRATO C
ON C.VENDEDOR=F.CPF
WHERE C.DATA >= SYSDATE - 30
GROUP BY F.CPF, F.NOME
ORDER BY SUM(C.PRECO) DESC;”

Consulta 5 - Arquivo: “Python/query5.py”

“Para cada supervisor, listar a quantidade de festas supervisionadas nos últimos 30 dias e a duração total das festas. Ordenar decrescentemente por duração total.”

Código

“SELECT FU.NOME, FU.CPF, COUNT(S.FESTA) AS NUM_FESTAS, SUM(FE.DURACAO) AS TEMPO_TOTAL
FROM SUPERVISOR JOIN FUNCIONARIO FU
ON SUPERVISOR.CPF=FU.CPF
JOIN SUPERVISIONA S
ON FU.CPF=S.SUPERVISOR
JOIN FESTA FE
ON FE.CODIGO=S.FESTA
WHERE FE.DATA_HORA >= SYSDATE - 30 AND FE.DATA_HORA < SYSDATE
GROUP BY FU.CPF, FU.NOME
ORDER BY SUM(FE.DURACAO) DESC;”

Consulta 6 - Arquivo: “Python/query6.py”

“Mostrar o total gasto com salários e o total arrecadado com festas nos últimos 30 dias.”

Código

“SELECT SUM(F.SALARIO) AS GASTO_SALARIOS, SUM(C.PRECO) AS ARRECADACAO_FESTAS
FROM FUNCIONARIO F, CONTRATO C
WHERE (C.DATA >= SYSDATE - 30) AND (C.DATA < SYSDATE);”

Consulta 7 - Arquivo: “Python/query7.py”

“Para cada local, listar a quantidade e o preço total das festas que lá ocorreram nos últimos 30 dias. Ordenar decrescentemente pelo preço total.”

Código

“SELECT L.NOME, COUNT(F.CODIGO) AS NUM_FESTAS, SUM(C.PRECO) AS PRECO_TOTAL, L.CEP, L.NUMERO
FROM LOCAL L JOIN FESTA F
ON (L.CEP=F.CEP) AND (L.NUMERO=F.NUMERO)
JOIN CONTRATO C

```
ON C.FESTA=F.CODIGO
WHERE F.DATA_HORA >= SYSDATE - 30 AND F.DATA_HORA < SYSDATE
GROUP BY L.CEP, L.NUMERO, L.NOME
ORDER BY SUM(C.PRECO) DESC;”
```

4. Conclusão

Nesse projeto conseguimos ter uma boa noção sobre como elaborar um banco de dados desde os primeiros estágios de modelagem até a implementação do mesmo. Assim, conseguimos ver a importância de cada uma das etapas para uma boa estruturação de um banco de dados.

Uma das dificuldades foi em elaborar a descrição do problema e os requisitos de dados na Parte 1, pois não tínhamos muita noção do que poderíamos fazer com um tema amplo como Assessoria de Eventos. Também achamos estranho nós mesmos elaborarmos o problema, pois normalmente o problema é exposto e tentamos achar uma maneira de resolvê-lo.

Outra dificuldade surgiu com a utilização de uma interface para realizar as operações, pois tivemos que aprender várias coisas que não estavam relacionadas com a disciplina, o que acabou atrasando o desenvolvimento de certas coisas.

Assim, para a aplicação nas outras turmas, talvez fosse interessante um maior direcionamento na primeira parte do projeto para ajudar na elaboração do problema. Além disso, ao invés de deixar toda a implementação do banco de dados em um programa na última parte do projeto, poderiam ser criados alguns programas pequenos durante o curso para adquirir mais experiência. Uma outra alternativa seria disponibilizar algum bom programa de turmas anteriores para poder ser estudado pelos grupos.