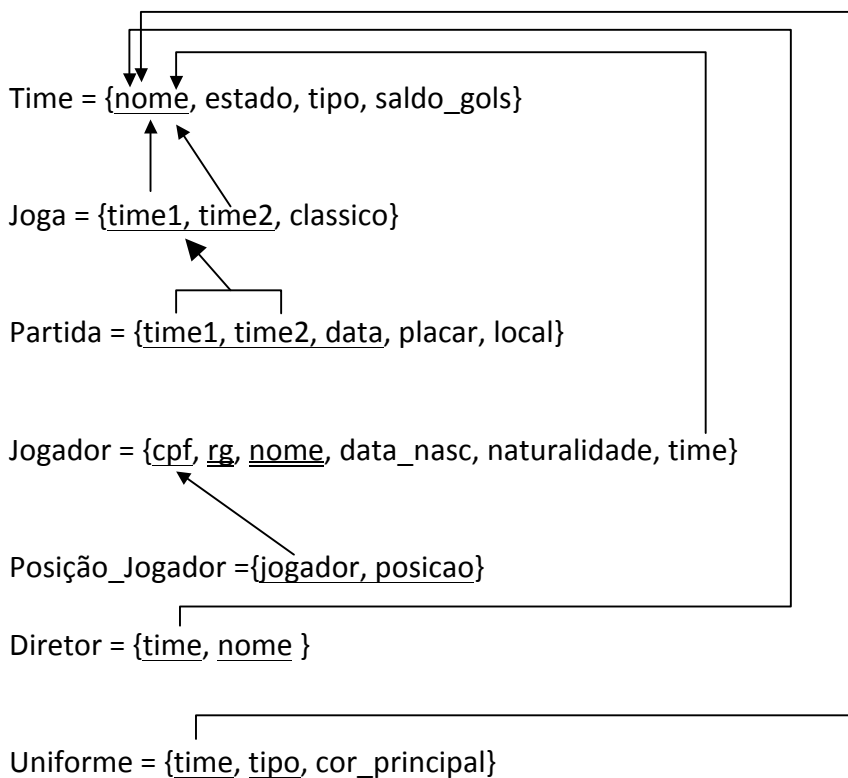


## Prática 2

### Exercícios SQL - DML (SGBD Oracle 19c)

Considere o esquema abaixo (o mesmo da Prática 1):



Nesse esquema, as seguintes restrições semânticas são conhecidas:

- o **tipo** do **Time** só pode assumir os valores 'AMADOR' e 'PROFISSIONAL';
- o atributo **classico** de **Joga** indica se o jogo é um clássico ou não;
- um jogador deve obrigatoriamente atuar em um time;
- o **tipo** do Uniforme pode ser 'TITULAR' ou 'RESERVA';
- o placar inicial (*default*) de uma partida é sempre 0X0;

## LEIA TODAS AS INSTRUÇÕES ANTES DE RESOLVER OS EXERCÍCIOS!

### SET UP INICIAL...

- 1) **Dicionário de Dados:** no Oracle, todos os dados do dicionário são armazenados em tabelas e disponibilizados para os usuários como visões (*views*). A documentação do Oracle traz uma lista das visões (e respectivas estruturas) que compõem o dicionário de dados (<https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/index.html> - *Database Reference (Part II - Static Data Dictionary Views)*).

Teste: `select * from user_tables;`

### 2) Preparando a base de dados....

- remova todas as tabelas criadas na Prática 1
  - **Dica:** usando o comando abaixo, crie um *script* de remoção para todas as suas tabelas (**o script será o resultado da consulta** – rode com o **Executar Script** que fica mais fácil copiar o resultado).


```
select 'drop table '||table_name||' cascade constraints;' from user_tables;
```

( || é um operador de concatenação de string )

- crie novas tabelas executando o *script* **Esquema\_Futebol.sql** (disponível no Tidia)
  - Menu: *File/Open* (ou abra como arquivo texto e faça *copy/paste*)
  - Selecione a conexão apropriada no lado direito da barra de ferramentas (se necessário)
  - Execute todo o *script* com o botão de **Executar Script**
- **alimente a base** com o *script* **Dados\_Futebol.sql** (disponível no Tidia)
- veja com atenção a estrutura e as restrições das tabelas criadas, e compare com o que você implementou na Prática 1 (principalmente FK, UNIQUE e CHECK). Veja também inserção de **NULL** e **DEFAULT** em Partida.

**IMPORTANTE:** quando uma **sessão de conexão** é iniciada, o SGBD inicia, implicitamente, uma **transação**. Os comandos SQL executados são operações dessa transação, e só são efetivados (resultados tornam-se persistentes) quando a transação termina com sucesso. A transação termina com sucesso quando:

- 1) um comando DDL é executado (auto-commit);
- 2) quando o usuário desconecta da base com sucesso (transação termina implicitamente);
- 3) quando o comando **commit** é executado explicitamente.

 Assim, para que as inserções, atualizações e remoções sejam efetivadas no banco, é necessário finalizar a transação com o comando **commit**.

## TESTE SOBRE OTIMIZADOR DE CONSULTAS E USO DE ÍNDICES...

- a) Conforme mencionado em aula, um índice é criado automaticamente para cada tabela. Considerando a tabela de **Time**: Qual a coluna usada como chave de indexação? É um índice de chave única ou de chave com repetição? Qual o nome do índice? Essas informações podem ser encontradas via interface do SQL Developer (clique duplo sobre a tabela) ou via consulta ao Dicionário de Dados.
- b) Como o seu esquema acabou de ser criado, ainda **não há estatísticas coletadas para ele**. Portanto, qualquer plano de consulta será gerado sem o uso de estatísticas, considerando apenas regras do otimizador e avaliações de custo. As estatísticas podem ser coletadas por meio do comando abaixo, que executa um procedimento do Oracle para **geração de estatísticas** a respeito das tabelas do esquema. Se o comando não for executado explicitamente, **as estatísticas são geradas automaticamente durante a noite** (quando o Oracle está configurado para isso).

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS(NULL, NULL);
```

- c) Execute, um a um, os comandos abaixo e observe os planos gerados, mais especificamente **o uso ou não do índice em consultas envolvendo o atributo Nome**. **Por que em alguns casos o índice é usado e em outro não?** Explique...

```
EXPLAIN PLAN FOR
```

```
select * from time where nome = 'INTER';
```

```
SELECT plan_table_output  
FROM TABLE(dbms_xplan.display());
```

-----

```
EXPLAIN PLAN FOR
```

```
select * from time where upper(nome) = 'INTER';
```

```
SELECT plan_table_output  
FROM TABLE(dbms_xplan.display());
```

-----

```
EXPLAIN PLAN FOR
```

```
select * from time where nome like 'IN%';
```

```
SELECT plan_table_output  
FROM TABLE(dbms_xplan.display());
```

-----

```
EXPLAIN PLAN FOR
```

```
select * from time where nome like '%TER';
```

```
SELECT plan_table_output  
FROM TABLE(dbms_xplan.display());
```

## EXERCÍCIOS...

- 1) Elabore comandos SQL para as consultas abaixo. Insira mais dados nas tabelas para verificar se os resultados das consultas estão corretos. **Elabore casos de teste para testar se realmente as consultas funcionam para casos gerais e casos especiais. As consultas devem ser eficientes, seguindo as recomendações gerais feitas nas aulas!**

**SOBRE OS DADOS:** assumo que nas tabelas **Joga** e **Partida** um mesmo par de times não será inserido 2 vezes com ordem trocada (por exemplo, assumo não será permitida inserção de <Santos, Palmeiras> e <Palmeiras, Santos>). Esta restrição poderia ser implementada, por exemplo, via *procedure* ou *trigger* (não é necessário implementar nessa prática).

### Dicas:

- em consultas que envolvem junções externas e internas, pode ser necessário colocar condições de seleção (filtros) na cláusula ON do JOIN (interno ou externo) e não na cláusula WHERE para ter o resultado correto. Teste os dois casos e avalie o efeito em consultas desse tipo;
- é possível (dependendo do SGBD) utilizar parênteses para determinar a ordem das junções, o que pode ser útil principalmente em consultas envolvendo várias junções externas e internas.
- é possível fazer junção de uma tabela com ela mesma (ou usar uma mesma tabela mais de uma vez na cláusula **FROM**), repetindo a tabela com "alias" diferentes.

Ex: `select ... from Tabela T1, Tabela T2 ...`

**Q1:** Selecione as partidas realizadas em Santos.

- **OBS:** note que **não são** pedidas todas as informações das partidas e, portanto, a consulta deve retornar **apenas o que for necessário para diferenciar uma partida da outra**.

**Q2:** Selecione data e local de todas as partidas jogadas pelo Palmeiras;

- teste o retorno do atributo de data sem formatação nenhuma (selecionando o atributo direto) e com formatação (Pesquise a função `to_char`).
- **OBS:** verifique no *script* de inserção (`Dados_Futebol.sql`) o formato utilizado para inserção das datas. **Por que a consulta sem formatação retorna os dados em formato diferente e aparentemente incompletos?**

**Q3:** Para cada jogador, selecione cpf, nome, idade, e nome e estado do time para o qual joga. Elabore a consulta “passo-a-passo” e avalie a resposta de cada passo, da seguinte maneira:

- 1) Faça um **select \*** e uma “junção” das tabelas necessárias **sem a condição de junção** (ou seja, um produto cartesiano). **OBS:** não é possível usar operador JOIN nesse exercício.
- 2) Inclua a condição de junção adequada.
- 3) Substitua o **\*** pela lista de atributos pedida na consulta

**Q4:** Para todos os jogadores que jogam em times do estado de SP, selecione cpf, nome, data de nascimento e nome do time. Faça duas versões dessa consulta:

1. Usando somente cláusulas **FROM** e **WHERE** para a operação de junção
2. Usando operador **JOIN** para a operação de junção

**Q6:** Selecionar, para todas as partidas realizadas em Santos, os nomes dos times, data da partida, placar e se é um clássico ou não.

**Q7:** Selecionar, para todos os confrontos clássicos (em Joga), nome e estado de cada um dos times.

**Q8:** Selecionar cor do uniforme titular de todos os times profissionais do estado de MG.

**OBS:** apensar do identificador do time não ser pedido explicitamente, a resposta só será interpretável se retornar o identificador do time associado à cor de uniforme

**Q9:** Selecionar as partidas em que pelo menos um time do estado de SP jogou.

**Q10:** Selecionar times que nunca jogaram em São Carlos e nem em Belo Horizonte.

**OBS:** essa consulta é similar à **consulta 11 da Aula 15**.

- Elabore com cuidado os casos de teste para avaliar o resultado da consulta.

**Q11:** Selecionar nome e estado dos times que possuem uniforme titular mas a cor principal não foi cadastrada.

**Q12:** Para os jogadores que já jogaram alguma partida, selecione nome, data de nascimento, time em que joga, data e local da partida, e se foi um clássico ou não.

**Q13:** Para cada time, selecionar nome, estado e, se houver, nomes dos diretores.

**Q14:** Selecionar a quantidade de partidas de confrontos clássicos e de confrontos não clássicos ocorridos em períodos de férias de verão (meses de janeiro e fevereiro).

**Dicas:**

- pesquise e use a função **EXTRACT**
- lembre-se de tratar: confrontos clássicos e não clássicos sem nenhuma partida cadastrada (contagem “zero”)
- lembre-se que a informação sobre ser clássico ou não clássico pode ser omitida na tabela

**Q15:** Selecionar a quantidade de jogos clássicos que ocorreram por mês no ano de 2018, considerando apenas meses com pelo menos 1 partida. Ordene o resultado do mês com maior quantidade para o de menor quantidade de jogos.

**Dica:**

- lembre-se que, para fazer sentido, a saída deve conter **{mes, contagem}**

**Q16:** Para cada time, selecionar: nome, estado, saldo de gols, e a quantidade de jogos clássicos que jogou por ano, considerando apenas anos em que o time jogou alguma partida (clássica ou não).

**Dica:**

- lembre-se de tratar contagem “zero” para os times que jogaram alguma partida no ano, mas nenhuma clássica.

**Q17:** Selecionar nomes dos times profissionais que jogaram clássicos e não marcaram gols em pelo menos 2 partidas.

**Dica:** veja com atenção o formato do placar no esquema da base de dados.

**Q18:** Selecionar a quantidade de times e a média do saldo de gols dos times por estado, por tipo de time. Ordenar o resultado por estado e depois por tipo de time.

**Dica:**

- lembre-se que para fazer sentido, a saída deverá ter: **{estado, tipo, contagem, media}**

**Q19:** Para os confrontos “clássicos” (**Joga**), selecione a quantidade de vezes que os 2 times já se enfrentaram (**PARTIDA**).

**Dica:**

- lembre-se de tratar contagem “zero” para os pares de times de confronto clássico que não têm partidas cadastradas.

**Q20:** Selecionar os times do estado de SP que jogaram em todos os locais em que o “Santos” jogou.

**Dica:**

- lembre-se que times que jogaram em apenas alguns (**mas não em todos**) os lugares que “Santos” jogou **não devem aparecer na resposta**.

**Q21:** Selecionar os times com o menor saldo de gols em cada estado. Apresentar nome do time, estado e saldo de gols.

**Dica:**

- se num estado houver mais de um time com o menor saldo de gols encontrado para o estado em questão, esses times aparecem em linhas separadas na resposta (e com mesmo estado e valor de saldo de gols. Ex: {<palmeiras, SP, 10>  
<santos, SP, 10>}

**DESAFIO:** Considere a consulta abaixo:

**Query:** Selecionar, para cada jogador: nome, data de nascimento, time em que joga, estado do time e, se o time jogou alguma partida clássica, listar também data e local da partida.

**Código:**

```
SELECT J.NOME, J.DATA_NASC, J.TIME, T.ESTADO, P.LOCAL, P.DATA
FROM JOGA JA JOIN PARTIDA P ON (JA.TIME1 = P.TIME1 AND JA.TIME2 = P.TIME2) AND JA.CLASSICO = 'S'
      RIGHT JOIN JOGADOR J ON JA.TIME1 = J.TIME OR JA.TIME2 = J.TIME
      JOIN TIME T ON J.TIME = T.NOME
ORDER BY J.NOME;
```

- a) Analise o código resposta **sem executá-lo**. Avalie a sequência de junções. Você acha que funciona? Por que?
- b) Elabore casos de teste considerando jogadores cujos times jogaram partidas clássicas e jogadores cujos times não jogaram partidas clássicas (nesse caso os dados de partida deveriam vir com **null** na resposta).
- c) Execute o código. Funciona? O resultado foi o esperado considerando sua resposta no item a)? Explique.....
- d) Elabore uma outra versão dessa consulta, também com apenas 1 junção externa.