**Checklist 1 - SQL e PL - GDI EC 2018.1**

**Data de Lançamento: 05/04/2018**

**Data de Entrega Parcial: 18/04/2018**

**Data de Entrega Final: 25/04/2018**

**Orientações:**

1. **Os alunos devem indicar que itens são abordados em cada trecho de código. É possível apresentar um só trecho de código que contenha mais de um item.**
2. **É necessário que os alunos saibam explicar exatamente o que cada trecho de código faz; caso contrário, não será considerada a pontuação do item.**
3. **As consultas SQL devem sempre retornar pelo menos uma linha.**
4. **Funções e procedimentos PL devem estar acompanhados de trechos de código que os executem, para demonstração.**
5. **Gatilhos devem estar acompanhados de instruções que os ativem, para demonstração.**

**SQL - básicas**

1. **Uso de BETWEEN com ​valores numéricos**SELECT \* FROM Curso WHERE nota\_mec BETWEEN 7 AND 9 ORDER BY nota\_mec DESC;
2. **Uso de BETWEEN com ​datas**SELECT \* FROM Professor WHERE data\_nasc BETWEEN to\_date('01/01/1970', 'dd/mm/yyyy') AND to\_date('12/12/1979', 'dd/mm/yyyy');
3. **Uso de LIKE/NOT LIKE com tokens (% ou \_)**SELECT nome FROM Disciplina WHERE nome LIKE 'F%';
4. **Uso de IN com subconsulta**SELECT nome FROM Aluno WHERE cpf\_aluno IN (SELECT cpf\_aluno FROM Ic);
5. **Uso de IS NULL/IS NOT NULL**SELECT nome FROM Aluno WHERE cpf\_padrinho IS NULL;
6. **Uso de ORDER BY**1.
7. **Criação de VIEW**CREATE VIEW P\_NomeEmail AS SELECT P.nome, E.email FROM Professor P, Email E WHERE P.cpf\_professor = E.cpf\_professor;
8. **Consulta sobre VIEW**SELECT nome FROM P\_NomeEmail GROUP BY nome HAVING COUNT(nome) > 1;
9. **Deleção de VIEW**DROP VIEW P\_NomeEmail;
10. **Criar CHECKs**
11. **Criar PK ​Composta**
12. **Criar FK ​Composta**
13. **Usar Valor DEFAULT ​(Ex: Data do Sistema)**
14. **Usar ALTER TABLE para Modificação de Coluna**ALTER TABLE Aluno MODIFY cr VARCHAR2(10);
15. **Usar ALTER TABLE para Adicionar Coluna**ALTER TABLE Aluno ADD cr NUMBER(4,2);
16. **Usar ALTER TABLE para Remover de Coluna**ALTER TABLE Aluno DROP COLUMN cr;
17. **Operadores aritméticos no SELECT**SELECT A.nome, AVG(C.media) AS Media\_Geral FROM Aluno A, Cursa C WHERE A.cpf\_aluno = C.cpf\_aluno GROUP BY A.nome;
18. **Função de agregação sem ​GROUP BY**SELECT P.titulo, (P.investimento - (SELECT AVG(investimento) FROM Projeto)) AS DIFF\_FROM\_AVG FROM Projeto P;
19. **Função de agregação com ​GROUP BY**
20. **Uso de DISTINCT**SELECT DISTINCT A.nome FROM Aluno A, Cursa C WHERE C.cpf\_aluno = A.cpf\_aluno;
21. **Uso de HAVING**8.
22. **Uso de HAVING com subconsulta**
23. **Uso de WHERE + HAVING**
24. **Junção entre duas tabelas**7.
25. **Junção entre três tabelas + condição de seleção (M:N)**
26. **Junção usando INNER JOIN**
27. **Junção usando LEFT OUTER JOIN**
28. **Junção usando RIGHT OUTER JOIN**
29. **Junção usando FULL OUTER JOIN**
30. **Uma subconsultacom ​uso de ANY ou SOME**
31. **Uma subconsultacom ​uso de ALL**
32. **Uma subconsultacom ​uso de EXISTS/NOT EXISTS**
33. **Uma subconsultacom ​uso de ALIAS ​com consultas aninhadas (ALIAS​ externo sendo referenciado na subconsulta)**
34. **Uso de UNION**
35. **Uso de INTERSECT**
36. **Uso de MINUS**
37. **INSERT com subconsulta**
38. **UPDATE com subconsulta**
39. **DELETE com subconsulta**DELETE FROM Cursa C1 WHERE C1.media < (SELECT AVG(C2.media) FROM Cursa C2 WHERE C1.codigo\_disc = C2.codigo\_disc);
40. **Uso de GRANT (VERDE)**

GRANT SELECT ON Disciplina TO ANONYMOUS;

1. **Uso de REVOKE (VERDE)**

REVOKE SELECT ON Disciplina FROM ANONYMOUS;

**SQL - avançadas**

1. **Subconsulta dentro da cláusula FROM (VIEW implícita)**SELECT MAX(Media\_Geral) AS Maior\_CR FROM (SELECT A.nome, AVG(C.media) AS Media\_Geral FROM Aluno A, Cursa C WHERE A.cpf\_aluno = C.cpf\_aluno GROUP BY A.nome);
2. **Operação aritmética com função de agregação como operador**18.
3. **Uso de BETWEEN com valores numéricos retornados por funções de agregação**
4. **Junção entre três tabelas usando INNER JOIN ou OUTER JOIN**
5. **ORDER BY com mais de dois campos**
6. **EXISTS com mais de uma tabela, sem fazer junção**

**PL - básicas**

1. **Bloco anônimo com declaração de variável e instrução**
2. **Bloco anônimo com exceção**
3. **Uso de IF-THEN-ELSE**
4. **Uso de ELSIF**
5. **Uso de CASE**
6. **LOOP com instrução de saída**
7. **WHILE LOOP**
8. **FOR LOOP**
9. **Recuperação de dados para variável**
10. **Recuperação de dados para registro**
11. **Output de string com variável**
12. **Uso de cursor explícito com ​variável**
13. **Uso de cursor explícito com ​registro**
14. **Uso de cursor explícito com ​parâmetro**
15. **Cursor dentro de FOR (sem DECLARE)**
16. **Procedimento sem ​parâmetro**
17. **Procedimento com ​parâmetro IN**
18. **Procedimento com ​parâmetro OUT**
19. **Procedimento com ​parâmetro INOUT**
20. **Uso de procedimento dentro de outro bloco PL ​(pode-se usar um dos procedimentos criados anteriormente)**
21. **Função sem ​parâmetro**
22. **Função com ​parâmetro IN**
23. **Função com ​parâmetro OUT**
24. **Função com ​parâmetro INOUT**
25. **Criação de pacote (declaração e corpo) com pelo menos dois componentes**
26. **BEFORE TRIGGER**

CREATE OR REPLACE TRIGGER red\_investimento

BEFORE UPDATE OF Investimento ON Projeto

FOR EACH ROW BEGIN

IF(:OLD.Investimento > :NEW.Investimento) THEN

RAISE\_APPLICATION\_ERROR (-200111,’Reducao do investimento nao permitida’);

END IF;

END;

1. **AFTER TRIGGER**

CREATE OR REPLACE TRIGGER red\_investimento

AFTER Investimento ON Projeto

FOR EACH ROW BEGIN

IF(:OLD.Investimento > :NEW.Investimento) THEN

RAISE\_APPLICATION\_ERROR (-200111,’Reducao do investimento nao permitida’);

END IF;

END;

1. **TRIGGER de linha sem ​condição**
2. **TRIGGER de linha com ​condição**

Sendo utilizada no item 72

1. **TRIGGER de comando**

CREATE OR REPLACE TRIGGER Demite BEFORE DELETE ON Professor

DECLARE

dia VARCHAR2(2);

BEGIN

dia:= EXTRACT(day FROM sysdate);

IF dia != ‘1’ THEN

RAISE \_APPLICATION\_ERROR(-20017,’Demissoes sao permitadas apenas no inicio do mes’);

END IF;

END Demite;

1. **Uso de NEW em TRIGGER de inserção**

CREATE OR REPLACE TRIGGER verifica\_Investimento

BEFORE INSERT OR UPDATE ON Projeto

REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW DECLARE

BEGIN

IF :NEW.Investimento > 90000 THEN

RAISE\_APPLICATION\_ERROR (-20202,'Valor excessivo de investimento');

END IF;

END;

1. **Uso de OLD em TRIGGER de deleção**

CREATE OR REPLACE TRIGGER deleta\_Investimento

BEFORE DELETE ON Projeto

REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW DECLARE

BEGIN

IF :OLD.Investimento > 5000 THEN

RAISE\_APPLICATION\_ERROR (-20202,'Investimento muito baixo para ser deletado');

END IF;

END;

1. **Uso de NEW e OLD em TRIGGER de atualização**

Sendo utilizada no item 72

1. **Uso de TRIGGER para impedir inserção em tabela**

CREATE OR REPLACE TRIGGER max\_alunos

BEFORE INSERT ON Aula

DECLARE a\_count number;

BEGIN

SELECT COUNT(CPF\_aluno) INTO a\_count from Aula;

IF a\_count > 40 THEN

RAISE\_APPLICATION\_ERROR (-20202, 'numero maximo de alunos por sala atingido');

END IF; END;

1. **Uso de TRIGGER para impedir atualização em tabela**

Sendo utilizada no item 72

1. **Uso de TRIGGER para impedir deleção em tabela**

Ta sendo utilizada no item 76

1. **Uso de TRIGGER para inserir valores em outra tabela**
2. **Uso de TRIGGER para atualizar valores em outra tabela**
3. **Uso de TRIGGER para apagar valores em outra tabela**

**PL - avançadas**

1. **Uso de função dentro de uma consulta SQL ​(pode-se usar uma das funções criadas anteriormente)**
2. **Registro como parâmetro de função ou procedimento**
3. **Função com registro como retorno**
4. **Pacote com funções ou procedimentos que usem outros componentes do mesmo pacote**
5. **INSTEAD OF TRIGGER**

CREATE OR REPLACE TRIGGER red\_investimento

INSTEAD OF Investimento ON Projeto

FOR EACH ROW BEGIN

IF(:OLD.Investimento > :NEW.Investimento) THEN

RAISE\_APPLICATION\_ERROR (-200111,’Reducao do investimento nao permitida’);

END IF;

END;