SQL e PL/SQL Oracle

Dicas de preparação para certificação

- **Selection**: para escolher linhas em uma tabela
- **Projection**: para escolher colunas em uma tabela
- **Join**: pode trazer simultaneamente dados que está armazenado em diferentes tabelas, criando um link entre elas.
- **SELECT**: uma lista de uma ou mais colunas. Usa-se para mostrar colunas específicas de uma tabela, especificando os nomes das colunas, separadas por vírgulas
- **FROM**: especifica a tabela que contém as colunas. Não pode usar operadores numéricos na cláusula FROM
- **DISTINCT**: suprime linhas duplicadas
- *****: seleciona todas as colunas
- **column**: seleciona as colunas específicas
- **alias**: seleciona colunas com cabeçalhos diferentes
- Uma **"palavra-chave"** refere-se a um elemento individual SQL. não podem ser separadas em quebra de linhas ou abreviadas, tipicamente são escritas em maiúsculas; todas as outras palavras, tal como nome de tabelas e colunas são escritas em minúsculo.
- Uma **cláusula** é uma parte de um comando SQL. são usualmente colocadas em linhas separadas para facilitar a leitura e edição
- Um **comando** é a combinação de duas ou mais cláusulas.
- Os comandos SQL não são case sensitive; podem ser escritos em uma ou muitas linhas;
- Dentro do SQL Plus, um comando SQL é escrito no prompt SQL, e as linhas subsequentes são numeradas. Isto é chamado de SQL buffer. Somente um comando pode ser corrente por vez no buffer.
- coloque um ponto-e-vírgula (;) no final da última cláusula
- coloque uma barra (/) na última linha do buffer
- usualmente um comando RUN do SQL Plus roda no prompt SQL.
- Default de cabeçalho de coluna: Maiúsculo
- Default de justificação de coluna: numéricos a direita; Data e caracteres a esquerda
- Cabeçalho numérico não pode ser truncado
- O SQL Plus ignora espaços brancos antes e depois dos operadores aritméticos
- Precedência de expressões aritméticas: *** / + -**
- Precedência de operadores lógicos: todos os operadores de comparação, NOT, AND e OR
- Zero é número, branco é caracter. Ambos são diferentes de nulo
- Para nomear colunas com nomes minúsculos, espaços ou caracteres especiais tem que usar aspas duplas
- Para concatenar colunas usa-se || barras verticais entre os campos
- Datas e caracteres literais vem entre ‘’ mas número não.
- Comandos SQL são armazenados no SQL Buffer e comandos do SQL Plus não
- **GET:** lista e executa
- START ou @: executa
- **SPOOL:** armazena resultados em um arquivo
- **DESCRIBE** mostra a estrutura de uma tabela
- **Operadores de comparação**: BETWEEN, IN, LIKE e IS NULL
- **BETWEEN** para selecionar uma faixa de valores inclusive
- **IN** parte qualquer de uma lista de valores

- **LIKE % qualquer seqüência de caracteres**
- **LIKE _ um caracter único**
- Precedência de operadores: todos os operadores de comparação, **NOT**, **AND**, **OR**
- **AND**: requer que ambas as condições são TRUE
- **OR**: requer que uma das opções seja TRUE
- **NOT**: negação
- **ORDER BY**: especifica a ordem na qual as linhas serão retornadas
- **CONCAT**: une strings
- **SUBSTR**: mostra uma seqüência de caracteres específicas a partir de uma posição específica
- **LENGTH**: informa o numero de caracteres uma string possui
- **INSTR**: informa em qual posição específica está um caracter específico
- **LPAD**: preenche uma quantidade de posições específicas com caracteres específicos justificado a direita
- **RPAD**: preenche uma quantidade de posições específicas com caracteres específicos justificado a esquerda
- **TRIM**: deleta caracteres específicos de uma string
- **MOD**: retorna resto da divisão
- **ROUND**: arredonda valores dual
- **TRUNC**: trunca valores – diferente de round
- **DUAL**: tabela do usuário SYS e que todos os usuários têm acesso
- **SYSDATE**: função que retorna a data e hora do sistema e deve ser obtida da tabela **DUAL**
- **NVL**: converte nulo para zero
- **DECODE**: funciona como If-Then-Else (ou case)
- **FM**: remove espaço em branco ou suprime zero. P. 3-29
- **JOIN**: é definida na cláusula WHERE
- **Equijoin**: (ou simple join ou inner join) uma coluna corresponde a uma coluna de outra tabela. Para fazer join com mais de duas tabelas, usa-se o AND
- **No_equijoin**: a coluna da tabela da esquerda não corresponde a da tabela direita
- **Outer_join**: retorna linhas ausentes na outra tabela. O sinal de adição (+) é colocado ao lado do join que não possui as informações. Esse tipo de relacionamento não pode usar operadores In ou OR
- **Self Join**: auto relacionamento
- **GROUP BY**: retorna uma linha para cada grupo encontrado. Não se pode usar com alias de coluna. Grupos podem ser excluídos usando cláusula HAVING
- **HAVING**: restrição de linha
- **AVG**: média
- **COUNT**: conta linhas
- **COUNT(*)**: retorna o número de linhas em uma tabela, incluindo linhas duplicadas e valores nulos
- **COUNT(***expr***)**: retorna o número de linhas não nulas em uma tabela, com as colunas identificadas pela **(***expr***)**
- **STDDEV**: desvio padrão
- **SUM**: soma
- **VARIANCE**: variação de determinado valor
- **DISTINCT**: elimina linhas duplicadas
- Funções AVG, SUM, VARIANCE e STDDEV só podem ser usados com dados numéricos. Para essas funções é necessário usar GROUP BY. Em caso de linhas restritas em resultados de um grupo de função, deve-se usar a cláusula HAVING ao invés da cláusula WHERE.
- **NVL**: força a inclusão de valores nulos
- A sequência da cláusulas é a seguinte: WHERE, GROUP BY E HAVING

- Subquery não pode ter ORDER BY.
- Quando uma consulta retorna mais que uma linha não se pode usar somente os operadores de comparação comuns (= > < <>) e sim os operadores IN, ANY e ALL ou conjuntamente com os comuns. O operador NOT não pode ser usado juntamente com IN, ANY e ALL.
- Em caso de subquery, não use NOT IN e sim !=ALL que é equivalente.
- O operador **IN** é equivalente a **=ANY**
- **ACCEPT:** Lê uma linha inserida pelo usuário e armazenada em uma variável.
- **DEFINE**: cria e especifica um valor para uma variável. Tipo CHAR. As variáveis são válida apenas para sessão corrente , para que sejam válidas em todas as sessões é necessário modificar o arquivo login.sql
- **&**: opção para o usuário inserir o valor
- **SET VERIFY ON**: força o SQL PLUS a mostrar um valor antes da substituição
- **&&**: para usar mais de uma vez o valor de variável dada pelo usuário
- **HIDE**: suprime o valor que o usuário insere. Ex.: o valor de uma senha inserida aparece com caracteres
- **UNDEFINE**: para limpar o valor definido com o DEFINE
- **BREAK**: suprime valores duplicados
- **TTITLE** e BTITLE: cabeçalho e rodapé. Quando o texto tem mais de uma palavra tem que colocar em aspas simples
- **TRANSACTION:** coleção de comandos DML
- **DML:** para adicionar linhas, modificar linhas e remover linhas em uma tabela
- **DDL:** CREATE, TRUNCATE (executam automático COMMIT)
- **DCL:** GRANT, REVOKE (executam automático COMMIT)
- **INSERT:** coloque em aspas simples para tipo de dados caracter e data. Dados numéricos não. Para inserir um formato de data que não o default, é necessário usar a função TO_DATE.
- **UPDATE:** modifica linhas
- **DELETE**: se omitir a cláusula WHERE, todas as linhas. Este comando elimina linhas mas não libera espaço.
- **COMMIT:** marca a transação como definitiva
- **SAVEPOINT:** permite um ROLLBACK até uma marca
- **ROLLBACK:** elimina dados pendentes
- **VALUES:** Quando junto com INSERT, adiciona apenas uma linha por vez ma tabela.
- Quando um comando não cabe na mesma linha, deve-se usar um traço – para continuar na próxima linha
- Schema: coleção de objetos
- **DROP**: se uma coluna é dropada, ela não pode ser recuperada. Este comando elimina linhas e a estrutura da tabela. Quando se dropa uma tabela, os índices correspondentes são dropados.
- **TRUNCATE**: remove linhas e libera espaço em disco
- **USER_TABLES**: tabelas pertencentes ao usuário
- **USER_OBJECTS**: objetos do usuário
- **USER_CATALOG**: tabelas, views, sinônimos e sequências pertencentes ao usuário
- Se uma tabela já contém linhas quando uma coluna é adicionada, a nova coluna é inicialmente nula para todas as linhas
- **UNUSED**: marcar a coluna como indisponível
- Pode ser adicionado uma coluna com o nome de uma coluna UNUSED
- **Tipos de Constraints**: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY E CHECK. Uma constraint pode ser adicionada, eliminada ou desabilitada, mas não se pode modificar sua estrutura. Pode ser usado o MODIFY apenas para adicionar uma constraint NOT NULL através do comando ALTER TABLE.

- **NOT NULL**: não pode conter valores nulos. Só pode ser especificada a nível de coluna, de tabela não é possível.
- **UNIQUE**: valores devem ser únicos. Quando envolve mais de uma coluna é chamada de *composite unique key.*
- **PRIMARY KEY**: identifica unicamente cada linha da tabela. Um index unique é automaticamente criado para uma coluna Primary Key
- **FOREIGN KEY (**ou *referenctial integry constraint***)**: estabelece relacionamentos entre tabelas.
- **REFERENCES**: usado com FK identifica a coluna da tabela pai
- **ON DELETE CASCADE**: usado com FK permite deleção na tabela pai e das linhas dependentes na tabela filha
- **CHECK**: especifica que a condição tem que ser TRUE. Não são permitidas referências a CURRVAL, NEXTVAL, LEVEL e pseudocolunas ROWNUM
- **DISABLE**: pode ser usado tanto no comando CREATE TABLE quanto no ALTER TABLE. Usando também a cláusula CASCADE, é desabilitada a integridade das constraints dependentes.
- **Base Tables**: são tabelas na qual uma view é baseada
- **VIEW**: representação lógica de dados de uma ou mais tabelas. Linhas podem ser removidas de uma View, desde que não contenha grupos de funções, a cláusula GROUP BY e a pseudocoluna ROWNUM
- **ROWNUM**: retorna o número de cada linha no Select
- **WITH CHECK POINT OPTION**: não permite a criação de linhas que a VIEW não seleciona
- **WITH READ ONLY:** view somente para leitura
- **DROP ANY VIEW:** para dropar view, mas só é permitido aos donos da view ou usuários com acesso DROP ANY VIEW
- Analysys Top-N: Para consultar os maiores valores. É necessário incluir ROWNUM, ORDER BY, DESC e a cláusula WHERE com os operadores ou < ou <=.
- **SEQUENCE**: para gerar valores sequenciais únicos. Qdo for usado para PK não deve usar a opção CYCLE.
- **NEXTVAL**: retorna o próximo valor da seqüência.
- **CURRVAL**: obtém o valor corrente da seqüência
- Com as pseudocolunas NEXTVAL e CURRVAL, não se pode usar uma lista Select de uma view, um comando Select com DISTINCT, com GROUP BY, com HAVING ou ORDER BY. Também não se pode usar em uma subquery com os comandos SELECT, DELETE ou UPDATE, nem a expressão DEFAULT em um CREATE TABLE ou ALTER TABLE.
- Privilégios típicos de DBA: CREATE USER, DROP USER, DROP ANY TABLE, BACKUP ANY TABLE
- Para o desenvolvimento de uma aplicação são necessários os seguintes privilégios de sistema: Create session, create table, create sequence, create view, create procedure
- **ROLE**: é um grupo de privilégios que podem ser dados ao usuário
- **GRANT juntamente com WITH GRANT OPTION**: permite que o usuário que recebeu estes privilégios repasse para outros.
- **REVOKE**: remove privilégios
- **Estrutura de um bloco PL/SQL**: Seção Declarativa, seção executável (mandatória), seção de tratamento de exceções. Para encerrar o bloco use um (;). Para rodar o bloco use (/), para fechar o buffer use um ponto(.)
- **Tipos de blocos**: anonymous, procedure e function.
- **PROCEDURES e FUNCTIONS**: são também conhecidos como *subprogramas*. A diferença entre eles é que as Functions retornam valores.

- **Programs constructs**: anonymous (é acessível de todos os ambientes PL/SQL), stores procedure (Oracle Server), application procedure ou function (Oracle Developer), packages (Oracle Server e Oracle Developer), database trigger (Oracle Server) e application trigger Oracle Developer)
- **Variáveis PL/SQL**: Scalar, composite (collections), Reference (ou pointers), LOB (ou Locator)
- Parâmetros de subprogramas: IN, OUT e IN OUT
- **Para inicializar variáveis**: usar (:=). Por default são inicializadas como NULL.
- Variáveis não podem Ter mais que 30 caracteres, o 1º caracter tem que ser uma letra, o restante podem ser caracteres, letras, números ou símbolos especiais.
- Tipos scalares: são classificados nas seguintes categorias: number, character, date e boolean.
- Tipos composite: TABLE, RECORD, NESTED TABLE e VARRAY
- Tipos LOB: CLOB, BLOB, BFILE e NCLOB
- Tipos booleanos: TRUE, FALSE e NULL
- PRINT: não pode ser usado dentro de um bloco PL/SQL
- Variáveis (host)que não são PL/SQL, devem conter no seu prefixo dois-pontos (:)
- Comentários: quando estiver em apenas uma linha, usar dois traços (--), quando ocupar mais de uma linha colocar entre os símbolos /*
- **Fmt**: é o formato usado para converter valores
- **%TYPE**: do mesmo tipo
- **Cursor**: Área de memória aberta pelo Oracle Server na qual o comando é analisado e executado. Podem ser implícitos ou explícitos. Os implícitos o Oracle Server usa para analisar e executar os comandos SQL. Os explícitos são declarados pelo programador.
- **Atributos do cursor**: SQL%ROWCONT (n.º de linhas afetadas pelo mais recente comando SQL), SQL%FOUND (avalia se afetou uma ou mais linhas), SQL%NOTFOUND (avalia se afetou nenhuma linha), SQL%ISOPEN.
- **Tipos de IF**: IF-THEN-END IF; IF-THEN-ELSE-END IF; IF-THEN-ELSIF-END IF;
- **Condições booleanas com Operadores lógicos** : AND - retorna TRUE somente se ambos os operadores forem TRUE e o OR retorna FALSE se ambos os operadores forem FALSE.
- **Tipos de LOOP**: Basic (ações repetitivas sem condições globais), FOR (controle interativo das ações baseadas em uma conta), WHILE (controle interativo das ações baseadas em uma condição).
- **REVERSE**: é usado para o loop FOR contar do maior para o menor
- **Tabela PL/SQL**: precisa conter uma pk BINARY_INTEGER para indexar a tabela, uma coluna scalar ou do tipo record na qual armazena elementos PL/SQL.
- A diferença entre o atributo %ROWTYPE e um tipo de dado composite RECORD é que este último permite especificar tipos de dados no registro ou declarar campos da qual for dono.
- **Active Set**: linhas retornadas de uma consulta de linhas múltiplas
- **Default de OPEN-CURSORS**: 50
- **%ISOPEN**: retorna o status do cursor
- **NOWAIT**: se a sessão estiver alocada para outro o Oracle retorna erro
- **WHERE CURRENT OF**: para referenciar uma linha corrente de um cursor explícito
- **Tipos de exceções**: Pré-definida pelo Oracle Server, não-pré-definida pelo Oracle Server e definida pelo usuário
- **Erros mais comuns**: NO_DATA_FOUND e TOO_MANY_ROWS
- **PRAGMA**: também chamado pseudoinstructions
- **Funções de exceções**: SQLCODE: retorna número do código do erro e SQLERRM: retorna a mensagem associada ao erro.
- **Procedure de tratamento de erros**: é usada na seção executável e na seção de exceções.