

Aplicando regras de negócio

Site: [TCU Moodle Site](#)
Curso: Curso Básico de Apex
Livro: Aplicando regras de negócio

Impresso por: RUAN HELENO CORREA DA SILVA
Data: quarta, 30 Mar 2022, 02:40

Índice

1. Revisão básica de SQL

1.1. Otimizando o SQL

2. Revisão básica de PL/SQL

2.1. Definição e estrutura do PL/SQL

2.2. Laços de execução

3. Técnicas de debug

4. Aplicando as regras de negócio

4.1. Regra de Negócio RN 01

4.2. Regra de Negócio RN 02

4.3. Regra de Negócio RN 03

4.4. Regra de Negócio RN 04

4.5. Regra de Negócio RN 05

4.6. Resumo

1. Revisão básica de SQL

SQL (Structured Query Language ou Linguagem de Consulta Estruturada) é a principal linguagem de pesquisa, utilizada nos mais diversos bancos de dados relacionais. Como o APEX é intrinsicamente ligado ao banco de dados Oracle, essa é a principal linguagem utilizada para entrarmos em contato com os dados e os objetos presentes no banco.

Nós consideramos que você já conhece o básico do SQL, em especial do SELECT, INSERT, UPDATE e DELETE.

No APEX, podemos utilizar o SQL de várias formas possíveis, fazendo com que nossas aplicações atinjam um grau de maturidade inalcançável sem ele. Veremos a seguir algum dos comandos mais utilizados. Para as práticas desta lição, acesse "**SQL Workshop**" > "**Comandos SQL**".

Para uma referência mais completa sobre SQL, busque na internet SQL Oracle e irá encontrar várias apostilas (inclusive em português) que vão do básico ao avançado. Caso você seja do TCU, vá à seção [Links úteis da comunidade APEX](#).

a. SUM, AVG, COUNT, MAX, MIN

Estas funções de grupo operam em conjuntos de linhas para fornecer um resultado por grupo.

Cada função aceita um argumento. Esta tabela identifica as opções que você pode usar na sintaxe:

Function	Descrição
<i>AVG(expr)</i>	Valor médio de <i>expr</i> ; ignora valores nulos.
<i>COUNT(expr)</i> ou <i>COUNT(*)</i>	<i>COUNT(expr)</i> - conta o número de linhas, em que <i>expr</i> é avaliado como um valor diferente de nulo. <i>COUNT(DISTINCT expr)</i> - conta o número de valores diferentes de <i>expr</i> que sejam diferente de nulo. <i>COUNT(*)</i> - conta todas as linhas selecionadas, inclusive valores duplicados e linhas com valores nulos.
<i>MAX(expr)</i>	Valor máximo de <i>expr</i> ; ignora valores nulos.
<i>MIN(expr)</i>	Valor mínimo de <i>expr</i> ; ignora valores nulos.
<i>SUM(expr)</i>	Valores somados de <i>expr</i> ; ignora valores nulos.

Exemplo:

Passo 1 - Para deixar o exemplo mais claro, iremos alterar a nota do aluno com o COD=1 para null, com o comando:

```
UPDATE aluno_turma
SET    nota = null
WHERE  cod_aluno = 1;
```

↑

Comandos SQL

☒ **Commit Automático**

Linhas

?

```
UPDATE aluno_turma
SET nota = null
WHERE cod_aluno = 1
```

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

1 linha(s) atualizada(s).

0,01 segundos

Repare que no Comandos SQL não é necessário terminar a consulta com ponto e vírgula.

Passo 2 - O exemplo abaixo mostra a contagem de de linhas, contagem de valores, contagem de diferentes, média, máximo, mínimo e soma das notas dos alunos com o comando:

```
SELECT COUNT(*) AS linhas, -- Retorna a quantidade de linhas da tabela
COUNT(nota) AS notas_nao_nulas, -- Retorna a quantidade de notas não nulas
COUNT(DISTINCT nota) AS notas_diferentes, -- Retorna a quantidade de valores diferentes de nota
AVG (nota) AS media, -- Retorna a média das notas
MAX (nota) AS maximo, -- Retorna a maior nota
MIN (nota) AS minino, -- Retorna a menor nota
SUM (nota) AS soma -- Retorna o somatório de todas as notas
FROM aluno_turma;
```

Passo 3 - Veja se os valores obtidos no passo anterior fazem sentido com as notas na tabela ALUNO_TURMA.

Comandos SQL

Esquema APEX_CURSO

☒ Commit Automático
 Linhas 10

Limpar Comando

Localizar Tabelas

Salvar

Executar

```
SELECT *
FROM aluno_turma;
```

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

COD	COD_ALUNO	COD_TURMA	NOTA
4	4	3	8
3	3	3	8
1	1	1	-
2	2	1	10

4 linhas retornadas em 0,01 segundos

Fazer Download

b. DECODE

DECODE (expressão, busca1, retorno1, busca2, retorno2,..., valor_padrao)

Comando:

```
SELECT nome,
DECODE(IND_SEXO, 'M', 'Masculino', 'Feminino') AS SEXO
FROM aluno
```

Comandos SQL

EsquemaAPEX_CURSO

☒ Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

SELECT nome,
DECODE(IND_SEXO, 'M', 'Masculino', 'F', 'Feminino', 'Indefinido') AS SEXO
FROM aluno

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

NOME	SEXO
Alberto Domingues Esteves	Masculino
Ana Maria Alves	Feminino
Carlos Alberto Alves	Masculino
Cristina Pires Domingues	Feminino
David Cunha Goncalves	Masculino
Eusebio dos Santos Aguiar	Masculino
Fabrice Stephane Alves Pereira	Feminino
Fernanda Maria Rodrigues	Feminino
Joao Paulo Barros da Silva	Masculino
Tania Maria Pereira	Feminino

10 linhas retornadas em 0,00 segundosFazer Download

O DECODE também pode receber mais parâmetros! No exemplo abaixo temos uma simulação das luzes de um semáforo.

Exemplo: DECODE(ind_semaforo, 1, 'Verde', 2, 'Amarelo', 3, 'Vermelho', 4, 'Com defeito')

c. LENGTH

A função LENGTH retorna a quantidade de caracteres de uma string ou de um campo.

Exemplo: Se precisássemos saber a quantidade de caracteres de cada nome dos alunos cadastrados.

Comandos SQL

EsquemaAPEX_CURSO

☒ Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

SELECT nome,
LENGTH (nome) AS tamanho_nome
FROM aluno;

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

NOME	TAMANHO_NOME
Alberto Domingues Esteves	25
Ana Maria Alves	15
Carlos Alberto Alves	20
Cristina Pires Domingues	24
David Cunha Goncalves	21
Eusebio dos Santos Aguiar	25
Fabrice Stephane Alves Pereira	30
Fernanda Maria Rodrigues	24
Joao Paulo Barros da Silva	26
Tania Maria Pereira	19

10 linhas retornadas em 0,01 segundosFazer Download

Comando:

```
SELECT nome,
LENGTH (nome) AS tamanho_nome
FROM aluno;
```

Existem algumas funções que são normalmente usadas associadas a LENGTH, como INITCAP, REPLACE, TRANSLATE, SUBSTR, RTRIM, LTRIM, CONCAT, etc. Para saber mais sobre elas, basta visitar os links úteis da Comunidade APEX do TCU, ou pesquise na internet.

d. CASE WHEN

No CASE o Oracle busca pela primeiro WHEN .. THEN para o qual a expressão é igual ao valor de busca e retorna a expressão de retorno.

Exemplo: Imagine que fosse necessário classificar o nome dos alunos em relação à quantidade de letras. Onde o nome com menos de 20 letras é considerado curto, com 20 é médio e com mais de 20 letras é considerado longo. Para esse tipo de problema, temos o CASE para nos ajudar como no comando abaixo.

Comando:

```
SELECT nome,
CASE WHEN LENGTH (nome) < 20 THEN 'Nome Curto'
WHEN LENGTH (nome) = 20 THEN 'Nome Médio'
ELSE 'Nome Longo' END tipo_nome
FROM aluno;
```

Comandos SQL

EsquemaAPEX_CURSO

☒ Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

Salvar

Executar

SELECT nome,
CASE WHEN LENGTH (nome) < 20 THEN 'Nome Curto'
WHEN LENGTH (nome) = 20 THEN 'Nome Médio'
ELSE 'Nome Longo' END tipo_nome
FROM aluno;

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

NOME	TIPO_NOME
Alberto Domingues Esteves	Nome Longo
Ana Maria Alves	Nome Curto
Carlos Alberto Alves	Nome Médio
Cristina Pires Domingues	Nome Longo
David Cunha Goncalves	Nome Longo
Eusebio dos Santos Aguiar	Nome Longo
Fabrice Stephane Alves Pereira	Nome Longo
Fernanda Maria Rodrigues	Nome Longo
Joao Paulo Barros da Silva	Nome Longo
Tania Maria Pereira	Nome Curto

10 linhas retornadas em 0,01 segundosFazer Download

Dica 1: Repare que no SELECT foi colocado um apelido TIPO_NOME para a coluna do CASE. No APEX é importante colocar apelidos na colunas, pois caso não tivesse sido colocado o Oracle assumiria que o nome da coluna é: "CASEWHENLENGTH(NOME)20THEN'NOMECURTO'WHENLENGTH(NOME)=20THEN'NOMEMÉDIO'ELSE'NOMELONGO'END".

Isso pode gerar problemas no APEX! Portanto, procure utilizar apelidos nas colunas em SELECT.

Dica 2: Outro detalhe é que o Oracle sempre faz UPPERCASE. O apelido na consulta foi colocado como "**tipo_nome**" (com letras minúsculas), mas repare que a coluna mostra "**TIPO_NOME**" (com letras maiúsculas).

e. IS NULL

Deve-se ter muito cuidado com os valores nulos no Oracle.

Importante: No Oracle QUALQUER operador utilizado com um valor nulo retorna FALSO, ou seja, nulo não é igual a nulo, nulo não é diferente de nulo, nulo não é maior nem menor a qualquer número. Detalhe que a string vazia é tratada como nulo também.

Exemplo de consulta que não retorna informação:

```
SELECT 1 FROM dual WHERE null=null OR null = '' OR null != null OR null > 1 OR null < 1;
```

No item a) desta seção atualizamos uma das notas de um aluno para nulo. Como podemos fazer uma consulta na tabela ALUNO_TURMA para obter esta nota nula?

Tente executar:

```
SELECT * FROM aluno_turma WHERE nota = null;
```

Você verá que nenhuma informação é retornada.

Qual seria a maneira correta?

A maneira correta de fazer comparação com nulo é utilizando o operador IS NULL.

Agora tente executar o seguinte comando:

```
SELECT * FROM aluno_turma WHERE nota IS NULL;
```

Comandos SQL

EsquemaAPEX_CURSO

☒ Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

Salvar

Executar

SELECT * FROM aluno_turma WHERE nota **IS NULL**;

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

COD	COD_ALUNO	COD_TURMA	NOTA
1	1	1	-

1 linhas retornadas em 0,00 segundosFazer Download

Agora é com você!

- 1. Execute cada um dos códigos SQL mostrados até agora no "SQL Workshop" > "Comandos SQL".
- 2. Caso seja do TCU, explore os links úteis relacionados à SQL na Comunidade APEX do TCU.

Link da Comunidade APEX: <https://acesso1.tcu.gov.br/comunidade/apex>

1.1. Otimizando o SQL

Da mesma forma que um bom SQL resolve os principais problemas de sistema e de negócio em uma solução informatizada, um SQL pouco otimizado leva a um baixo desempenho e, portanto, frustração aos usuários finais.

Como otimização de SQL é assunto para um curso inteiro, vamos apresentar apenas algumas dicas de como melhorar um SQL.

a. Evite o produto cartesiano

Quando uma condição de join (junção) é inválida ou completamente omitida, o resultado é um produto cartesiano, no qual todas as combinações de linha são exibidas. Todas as linhas da primeira tabela são unidas a todas as linhas da segunda tabela.

Um produto cartesiano tende a gerar um grande número de linhas e o resultado raramente é útil. Inclua sempre uma condição de join válida, a menos que exista uma necessidade específica de combinar todas as linhas de todas as tabelas.

Exemplo: Um produto cartesiano será gerado se uma condição de join for omitida.

Comando:

```
SELECT a.nome, t.descricao, at.nota
FROM aluno a, turma t, aluno_turma at
```

↑ Comandos SQL

Esquema APEX_CURSO

☒ Commit Automático

Linhas 10

Limpar Comando

Localizar Tabelas

Salvar

Executar

SELECT a.nome, t.descricao, at.nota
FROM aluno a, turma t, aluno_turma at

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

NOME	DESCRICAO	NOTA
Alberto Domingues Esteves	Curso de APEX	8
Ana Maria Alves	Curso de APEX	8
Carlos Alberto Alves	Curso de APEX	8
Cristina Pires Domingues	Curso de APEX	8
David Cunha Goncalves	Curso de APEX	8
Eusebio dos Santos Aguiar	Curso de APEX	8
Fabrice Stephane Alves Pereira	Curso de APEX	8
Fernanda Maria Rodrigues	Curso de APEX	8
Joao Paulo Barros da Silva	Curso de APEX	8
Tania Maria Pereira	Curso de APEX	8

Há mais de 10 linhas disponíveis. Aumente o seletor de linhas para ver mais linhas.

10 linhas retornadas em 0,01 segundos

Fazer Download

Clicando na aba **Explicação** veremos o plano de execução, que é a forma que o banco de dados usa para recuperar as informações.

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

Salvar

Executar

SELECT a.nome, t.descricao, at.nota
FROM aluno a, turma t, aluno_turma at

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

Plano de Consulta

Operação	Opções	Objeto	Linhas	Hora	Custo	Bytes	Predicados de Filtro *	Predicados de Acesso
SELECT STATEMENT			120	1	27	6.480		
MERGE JOIN	CARTESIAN		120	1	27	6.480		
MERGE JOIN	CARTESIAN		12	1	9	360		
TABLE ACCESS	FULL	TURMA	3	1	3	81		
BUFFER	SORT		4	1	6	12		
TABLE ACCESS	FULL	ALUNO_TURMA	4	1	2	12		
BUFFER	SORT		10	1	25	240		
TABLE ACCESS	FULL	ALUNO	10	1	2	240		

* As colunas não indexadas são mostradas em vermelho

Em destaque vemos a palavra CARTESIAN que indica que aconteceu um produto cartesiano e o Custo que é o "preço" relativo que o banco de dados cobra para executar esta operação.

Nesta consulta o tempo de execução foi rápido, pois temos poucos registros. Lentidão em consulta é sentido de maneira muito intensa quando as tabelas são grandes.

Agora com o JOIN correto (evitando o produto cartesiano).

Comando:

```
SELECT a.nome, t.descricao, at.nota  
FROM aluno a, turma t, aluno_turma at  
WHERE a.cod = at.cod_aluno AND t.cod = at.cod_turma
```

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

Salvar

Executar

SELECT a.nome, t.descricao, at.nota
FROM aluno a, turma t, aluno_turma at
WHERE a.cod = at.cod_aluno AND t.cod = at.cod_turma

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

NOME	DESCRICAO	NOTA
Alberto Domingues Esteves	Curso de APEX	-
Ana Maria Alves	Curso de APEX	10
Carlos Alberto Alves	Curso de PL/SQL	8
Cristina Pires Domingues	Curso de PL/SQL	8

4 linhas retornadas em 0,01 segundosFazer Download

Clique na aba **Explicação** para ver o Plano de execução deste SELECT que agora NÃO contém o produto cartesiano.

Comandos SQL

EsquemaAPEX_CURSO

☒ Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

Salvar

Executar

```
SELECT a.nome, t.descricao, at.nota
FROM aluno a, turma t, aluno_turma at
WHERE a.cod = at.cod_aluno AND t.cod = at.cod_turma
```

Resultados	Explicação	Descrever	Instrução SQL Salva	Histórico
	ALUNO	ALUNO_PK		CODUNIQUEVALIDNORMALNO
	ALUNO_TURMA	ALUNO_TURMA_IDX_ALUNO		COD_ALUNONONUNIQUEVALIDNORMALNO

Plano de Consulta

Operação	Opções	Objeto	Linhas	Hora	Custo	Bytes	Predicados de Filtro *	Predicados de Acesso
SELECT STATEMENT			4	1	9	264		
HASH JOIN			4	1	9	264		"A"."COD" = "AT"."COD_ALUNO"
MERGE JOIN			4	1	6	156		
TABLE ACCESS	BY INDEX ROWID	TURMA	3	1	2	90		
INDEX	FULL SCAN	TURMA_PK	3	1	1			
SORT	JOIN		4	1	4	36	"T"."COD" = "AT"."COD_TURMA"	"T"."COD" = "AT"."COD_TURMA"
TABLE ACCESS	FULL	ALUNO_TURMA	4	1	3	36		
TABLE ACCESS	FULL	ALUNO	10	1	3	270		

* As colunas não indexadas são mostradas em vermelho

Comparando com a Explicação anterior, vemos que a palavra CARTESIAN sumiu e que o Custo para executar essa consulta caiu em mais de 60%. Isso aconteceu porque fizemos os joins e estes joins utilizam os índices da tabela, fazendo com que esta consulta retorne o resultado correto com bem menos custo para o banco e bem menos tempo.

Assim, sempre que um SQL estiver lento, fuja das construções cartesianas!

b. Use índices

Um índice no Oracle é um objeto de esquema que pode acelerar a recuperação de linhas.

É possível criar índices de forma explícita ou automática (quando for criada uma chave primária ou única).

Um índice permite acesso direto e rápido a linhas de uma tabela. Seu objetivo é reduzir a necessidade de I/O de disco usando um caminho indexado para localizar os dados com rapidez. O servidor Oracle usa e mantém automaticamente o índice.

Diretrizes para a Criação de um Índice

Crie um índice quando:	
<input checked="" type="checkbox"/>	Uma coluna contiver uma grande faixa de valores
<input checked="" type="checkbox"/>	Uma coluna contiver um grande número de valores nulos
<input checked="" type="checkbox"/>	Uma ou mais colunas forem usadas em conjunto com frequência em uma cláusula WHERE ou em uma condição de join
<input checked="" type="checkbox"/>	A tabela for grande e for esperado que a maioria das consultas recupere menos de 2% a 4% das linhas da tabela
Não crie um índice quando:	
<input checked="" type="checkbox"/>	As colunas não forem usadas com frequência como uma condição na consulta
<input checked="" type="checkbox"/>	A tabela for pequena ou for esperado que a maioria das consultas recupere mais de 2% a 4% das linhas da tabela
<input checked="" type="checkbox"/>	A tabela for atualizada com frequência
<input checked="" type="checkbox"/>	As colunas indexadas forem referenciadas como parte de uma expressão

c. Use bind variable (*:Pn_ITEM*)

O uso de variáveis de *bind* constitui numa boa prática que traz benefícios para a chamada de SQLs idênticos.

Imagine que se deseja fazer consultas a alunos pelo código.

```
SELECT * FROM aluno WHERE cod = 1;
```

```
SELECT * FROM aluno WHERE cod = 2;
```

....

Entretanto, ao usamos uma variável de bind, a consulta se torna a mesma e o Oracle "economiza" diversas etapas, tais como a verificação de sintaxe e semântica da consulta.

```
SELECT * FROM aluno WHERE cod = :cod_aluno;
```

Onde *:cod_aluno* (bind variable) significa que esse valor será informado pelo usuário.

Repare que com o uso de variáveis de bind, só precisamos definir o valor do COD_ALUNO, mas a consulta permanece a mesma. Quando a consulta for executada pela segunda vez, já estará em memória e a performance será muito mais rápida.

O APEX utiliza muito este recurso de bind variable passando o valor do itens (Ex: P3_COD) como variável de bind.

Faça essas consultas nos **Comandos SQL** do **SQL Workshop**!

d. Outras possibilidades

Uso ineficiente de cursores e do *shared pool*: A não reutilização de cursores em análises (pares) repetidas. Se uma variável de *bind* não for usada então acontecerá um hard parse de todas as instruções SQL. Isto tem um impacto enorme na performance, sendo totalmente não-escalável. Utilize os cursores com variáveis de *bind* que abrem o cursor e o executam várias vezes. Você verá mais sobre cursores na próxima lição.

Código SQL Ruim: É o código que usa mais recursos do que o necessário para os requisitos da aplicação. Isto pode acontecer numa consulta que roda por horas ou uma consulta em uma aplicação tradicional que leva mais de 1 minuto. Os SQLs que consomem uma quantidade significativa de recursos devem ser investigados para que possam ser ajustados.

Varredura Integral (*Full-table scan*): Varredura total na tabela em operações online podem indicar um design ineficiente da transação, falta de índices ou otimização ineficiente de SQL.

Erros nos *schema* e problemas no otimizador: Em vários casos, um usuário da aplicação usa muitos recursos porque o schema proprietário das tabelas não foi migrado com sucesso do ambiente de desenvolvimento ou da sua versão anterior. Como exemplo, podemos citar a falta de índices ou estatísticas incorretas. Estes erros podem deixar os planos de execução sem eficiência, com baixa performance para o usuário.

Mesma consulta repetida diversas vezes: Algumas informações são necessárias em vários momentos da aplicação, como por exemplo o código da unidade organizacional que uma pessoa trabalha, o código do usuário logado, etc. Como essas são informações que mudam com baixa frequência, pode ser interessante fazer a consulta uma vez e armazenar o valor em um item de aplicação no APEX.

2. Revisão básica de PL/SQL

PL/SQL é a linguagem estruturada da Oracle, utilizada para desenvolver operações e procedimentos que seriam impossíveis de serem implementados utilizando apenas SQL. Dentro do PL/SQL temos criação de variáveis, estruturas de decisão e repetição. Com esse conjunto de ferramentas, poderemos criar nossos processos, filtros e cálculos dentro do Apex.

Veremos alguns dos principais tópicos do PL/SQL.

2.1. Definição e estrutura do PL/SQL

A estrutura básica de um bloco PL/SQL está dividida em três sessões:

Sessão	Descrição	Inclusão
Declarativa (DECLARE)	Contém as declarações de todas as variáveis, constantes, cursores e exceções definidas pelo usuário que serão referenciadas na seção executável e na seção de exceção.	Opcional
Executável (BEGIN ... END)	Contém instruções SQL para retornar dados do banco de dados; contém instruções PL/SQL para manipular dados em um bloco.	Obrigatório
Exceção (EXCEPTION)	Especifica as ações que serão executadas quando acontecerem erros e condições anormais na sessão executável.	Opcional

Em um bloco PL/SQL as palavras-chave **DECLARE**, **BEGIN** e **EXCEPTION** não são terminadas com um ponto e vírgula [;]. Contudo a palavra-chave **END**, todas as instruções SQL e PL/SQL devem terminar com um ponto e vírgula [;].

Exemplo: Abaixo temos o exemplo da estrutura de um bloco anônimo, procedure e função.

Bloco Anônimo	Procedure	Função
<pre>[DECLARE] BEGIN --instruções [EXCEPTION] END;</pre>	<pre>PROCEDURE teste IS BEGIN --instruções [EXCEPTION] END;</pre>	<pre>FUNCTION teste RETURN tipo de dados IS BEGIN --instruções RETURN valor; [EXCEPTION] END;</pre>

Bloco anônimo – Deve ser utilizado em rotinas pontuais, que são executadas para atender uma demanda específica, como um processo, uma migração de dados ou correção dos mesmos, por exemplo.

Procedure (Procedimento) – É um procedimento que fica armazenado como um objeto no banco de dados. Com isso a sua reutilização pode ser facilmente feita, uma vez em que podemos colocar apenas o nome da procedure no nosso código PL/SQL e passar os parâmetros necessários para que a execução aconteça. Quanto aos parâmetros, as procedures podem receber ou não parâmetros de entrada e de saída, o que pode fazer com que ela tenha zero ou mais retornos de dados.

Exemplo: O procedimento abaixo **PROC_ATUALIZA_CURRICULO** está sendo criado para permitir atualizar todos os currículos dos professores com a frase 'Sem currículo cadastrado' quando os professores não tiverem nada cadastrado neste campo.

```
-- Nome da procedure, sem parâmetros de entrada ou de saída. Caso fosse necessário deveria aparecer, entre parênteses, o nome do
parâmetro, um indicador se ele é de entrada ou saída e o tipo de dado dele com vírgulas separando os parâmetros. Ex: (param1 IN NUMBER,
param2 IN VARCHAR2, param3 OUT DATE)
CREATE PROCEDURE proc_atualiza_curriculo
IS
BEGIN
  UPDATE professor -- Atualização de dados
  SET curriculo = 'Sem currículo cadastrado'
  WHERE curriculo IS NULL;

  COMMIT; -- Confirmação da alteração dos dados. A ausência do COMMIT pode causar problemas!!

END;
```

Uma procedure deve ser invocada em um bloco PLSQL.

Comando:

```
BEGIN
proc_atualiza_curriculo
END;
```

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

```
BEGIN
proc_atualiza_curriculo;
END;
```

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

Instrução processada.

0,01 segundos

Function (Função) – É uma função que fica armazenada no banco de dados e tem como objetivo retornar obrigatoriamente um valor, possuindo ou não parâmetros de entrada. Para utilizar uma função, podemos colocar uma variável recebendo o retorno desta ou usá-la diretamente dentro da nossa expressão SQL.

Exemplo: A função abaixo retorna a quantidade de alunos de uma determinada turma, que é passada através de um parâmetro de entrada

```
-- Nome da função com um parâmetro de entrada numérico
CREATE FUNCTION f_qtd_participante_turma (p_cod_turma NUMBER)
RETURN NUMBER -- Indica que a função irá retornar um valor numérico
IS
v_qtd_participante    NUMBER; -- Esta área entre o IS e o BEGIN é a área de declaração de variáveis, não constando a palavra reservada DECLARE.
BEGIN
SELECT COUNT (*)      -- Corpo da função, que neste caso é uma consulta que retorna
INTO v_qtd_participante -- A quantidade de alunos cujo o COD_TURMA seja igual ao parâmetro passado
FROM aluno_turma -- Jogando este resultado na variável v_qtd_participante
WHERE cod_turma = p_cod_turma;

RETURN v_qtd_participante; -- Informa que a variável v_qt_participante será o valor de retorno desta função
END;
```

Podemos fazer uma chamada à função tanto em uma coluna de uma consulta, e um filtro de uma consulta ou em um bloco PLSQL.

O exemplo abaixo faz uma consulta mostrando a quantidade de participantes e filtrando para mostrar apenas as turmas com mais de zero participantes inscritos.

Comando:

```
SELECT t.descricao, f_qtd_participante_turma (t.cod) PARTICIPANTES
FROM turma t
WHERE f_qtd_participante_turma (t.cod) > 0
```

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

```
SELECT t.descricao, f_qtd_participante_turma (t.cod) PARTICIPANTES
FROM turma t
WHERE f_qtd_participante_turma (t.cod) > 0
```

ResultadosExplicaçãoDescreverInstrução SQL SalvaHistórico

DESCRICAO	PARTICIPANTES
Curso de APEX	2
Curso de PL/SQL	2

2 linhas retornadas em 0,00 segundosFazer Download

Packages (Pacotes)

Resumidamente, Packages são procedures e functions de banco empacotadas por assunto, facilitando assim a organização dos códigos PL/SQL. Ela é composta pela PACKAGE (área responsável pela interface da package com os códigos PL/SQL, armazenando aqui além do cabeçalho dos procedimentos e funções, declaração de variáveis que serão visíveis a todas as partes da package) e o PACKAGE BODY (que como o nome sugere, é o corpo da package, local onde constam todos os códigos das funções e procedimentos).

Vantagens do uso de pacotes

Modularidade e Facilidade de Manutenção: através do encapsulamento das estruturas relacionadas num módulo nomeado. Cada package deve ser fácil de entender e a interface entre packages deve ser simples, clara e bem definida.

Design de aplicação fácil: A codificação e a compilação da especificação e do corpo da package podem ser feitas separadamente.

Melhor performance: Na primeira chamada de um subprograma dentro de uma package, toda a package é carregada na memória. Após isto, pode acontecer das chamadas seguintes aos subprogramas de uma package não precisarem de I/O de disco, pois apenas uma cópia da package em memória é necessária para todos os usuários.

Exemplo: Abaixo temos um exemplo de package.

```
CREATE OR REPLACE PACKAGE pck_professor
IS
PROCEDURE proc_atualiza_curriculo; -- mesma procedure criada anteriormente, agora numa package

FUNCTION f_qtd_professores RETURN NUMBER; -- função que retorna a quantidade de professores cadastrados
END;

----Abaixo o código da package body.
CREATE OR REPLACE PACKAGE BODY pck_professor
IS
PROCEDURE proc_atualiza_curriculo
IS
BEGIN
UPDATE professor
SET curriculo = 'Sem currículo cadastrado'
WHERE curriculo IS NULL;

COMMIT;
END;

FUNCTION f_qtd_professores
RETURN NUMBER
IS
v_qtd_professores    NUMBER;
BEGIN
SELECT COUNT (*)
INTO v_qtd_professores
FROM professor;

RETURN v_qtd_professores;
END;
END;
```

Quando você for referenciar a package em um SQL ou processo, use **PCK_PROFESSOR.PROC_ATUALIZA_CURRICULO** ou **PCK_PROFESSOR.F_QTD_PROFESSORES** (O Oracle não é case sensitive).

```
SELECT INTO
```

Com o PL/SQL é possível declarar variáveis e usá-las em outras instruções. As variáveis são usadas para armazenar dados temporariamente e para manipular estes valores armazenados. Um uso prático dessas variáveis é ser o retorno de uma função.

Em suma, é a forma mais simples de transferir uma informação vinda de um SELECT no banco para uma variável programável.

A reusabilidade é outra vantagem da declaração de variáveis. Após a sua declaração, as variáveis podem ser usadas repetidamente em uma aplicação através de sua referência em instruções.

Repare que no código da função que se encontra dentro do pacote (package) foi utilizada uma variável para armazenar a quantidade de professores e que foi utilizado um comando **SELECT INTO**

Vamos ver um exemplo que usa o SELECT INTO e imprime o retorno.

```
DECLARE
v_qtd_professores NUMBER;

BEGIN

SELECT COUNT (*)
INTO v_qtd_professores
FROM professor;

htp.p('Quantidade de professores:' || v_qtd_professores);

END;
```

Agora é com você!

- 1) Faça a criação e execução da procedure mostrada nesta seção.
- 2) Faça a criação e execução da função mostrada nesta seção.
- 3) Faça a criação e execução do pacote (package) mostrado nesta seção.
- 4) Acesse o **SQL Workshop > Browser de Objetos** e encontre o procedimento (procedure), a função e o pacote criados nesta seção.

2.2. Laços de execução

Nesta seção veremos vários tópicos relacionados a laços de execução no PL/SQL.

Muito cuidado no uso de loops, pois um erro de lógica pode levar a um loop infinito. No PL/SQL apenas o administrador do banco de dados pode parar a execução de um loop infinito.

a. LOOP END

Esta é a forma mais simples de fazer uma estrutura de LOOP, que engloba uma sequência de instruções entre as palavras-chave LOOP e END LOOP. A cada vez que o fluxo de execução atinge a instrução END LOOP o controle é retornado para a instrução LOOP correspondente.

Exemplo: O exemplo abaixo cria uma turma e insere 5 alunos nessa turma.

Código:

```
DECLARE
    v_cod_turma NUMBER;
    v_cod_aluno NUMBER;
    v_contador  NUMBER:= 0;
BEGIN
    --Inserimos uma turma e obtemos o código dela pelo comando RETURNING INTO
    INSERT INTO turma (descricao, periodo, horario, sala, qtd_vagas, valor_curso, ind_situacao)
    VALUES ('Novo curso de APEX','Tarde', '14h', 'Laboratório 1', 25, 2300, 'A')
    RETURNING cod INTO v_cod_turma;
    --Vamos fazer um loop para matricular 5 alunos nesta turma recém-criada
    LOOP
        --Buscamos o aluno de menor código que ainda não está matriculado na turma recém-criada
        SELECT MIN(cod)
        INTO v_cod_aluno
        FROM aluno
        WHERE cod NOT IN (SELECT cod_aluno FROM aluno_turma WHERE cod_turma = v_cod_turma);
        --Insere o aluno na turma
        INSERT INTO aluno_turma (cod_aluno, cod_turma, nota)
        VALUES (v_cod_aluno, v_cod_turma, v_contador*2);
        --Incrementa o contador
        v_contador := v_contador + 1;
        EXIT WHEN v_contador = 5; --Sai do LOOP quando tiver inserido 5 registros
    END LOOP;
END;
```

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

```
DECLARE
  v_cod_turma NUMBER;
  v_cod_aluno NUMBER;
  v_contador  NUMBER:= 0;

BEGIN
  --Inserimos uma turma e obtemos o código dela pelo comando RETURNING INTO
  INSERT INTO turma (descricao, periodo, horario, sala, qtd_vagas, valor_curso, ind_situacao)
  VALUES ('Novo curso de APEX', 'Tarde', '14h', 'Laboratório 1', 25, 2300, 'A')
  RETURNING cod INTO v_cod_turma;

  --Vamos fazer um loop para matricular 5 alunos nesta turma recém-criada
  LOOP
    --Buscamos o aluno de menor código que ainda não está matriculado na turma recém-criada
    SELECT MIN(cod)
    INTO v_cod_aluno
    FROM aluno
    WHERE cod NOT IN (SELECT cod_aluno FROM aluno_turma WHERE cod_turma = v_cod_turma);

    --Insere o aluno na turma
    INSERT INTO aluno_turma (cod_aluno, cod_turma, nota)
    VALUES (v_cod_aluno, v_cod_turma, v_contador*2);

    --Incrementa o contador
    v_contador := v_contador + 1;

    EXIT WHEN v_contador = 5; --Sai do LOOP quando tiver inserido 5 registros
  END LOOP;
END;
```

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

1 linha(s) inserida(s).

0,04 segundos

Ao final confira se os registros foram inseridos com o comando:

```
SELECT *
FROM aluno_turma;
```

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

```
SELECT *
FROM aluno_turma
```

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

COD	COD_ALUNO	COD_TURMA	NOTA
4	4	3	8
3	3	3	8
1	1	1	-
2	2	1	10
63	1	24	0
64	2	24	2
65	3	24	4
66	4	24	6
67	5	24	8

9 linhas retornadas em 0,00 segundosFazer Download

Note que foram inseridos 5 registros na turma com maior código. O código da turma foi para um número muito acima da sequência que se tinha na tabela de turmas. No caso de nossa execução o código foi o 24, mas na sua execução pode ser diferente. A sequência apesar do nome é um objeto no Oracle que não necessariamente é inserida na ordem e pode pular valores. **Não se prenda a querer que os valores gerados ao objeto sequência**

(SEQUENCE) do Oracle seja realmente uma sequência numérica! (Exceto se sua regra de negócio exija isso)

b. FOR LOOP

Os loops FOR têm a mesma estrutura geral de um loop básico. Além disso, o FOR possui uma instrução de controle antes da palavra-chave LOOP para definir o número de iterações (repetições) que o PL/SQL irá executar.

Exemplo: Vamos inserir os pagamentos relativos aos alunos já matriculados. O valor final do curso a ser pago depende da nota do aluno. Se a nota do aluno for maior que 8, ele tem desconto de 10%, caso contrário, ele tem que pagar mais 10% pela aula de reforço. Para que a data de pagamento não fique a mesma, consideramos a data de pagamento o dia de hoje (SYSDATE) e subtraímos o código em dias.

Comando:

```
DECLARE
v_valor_final NUMBER;
BEGIN
--FOR percorrendo um CURSOR
FOR cur IN (SELECT at.cod, at.nota, t.valor_curso
            FROM aluno_turma at
            JOIN turma t ON t.cod = at.cod_turma)
LOOP
    IF cur.nota >= 8 THEN v_valor_final := cur.valor_curso * 0.9;
    ELSE v_valor_final := cur.valor_curso * 1.1;
    END IF;
    --Insere na data atual (SYSDATE) subtraído do código em dias.
    INSERT INTO pagamento (cod_aluno_turma, data, valor) VALUES (cur.cod, SYSDATE-cur.cod, v_valor_final);

    http.p('Inseriu um pagamento no valor de '||v_valor_final);
END LOOP;
END;
```

Comandos SQL

EsquemaAPEX_CURSO

☒ Commit Automático

Linhas10

Limpar Comando

Localizar Tabelas

SalvarExecutar

```
DECLARE
v_valor_final NUMBER;
BEGIN
--FOR percorrendo um CURSOR
FOR cur IN (SELECT at.cod, at.nota, t.valor_curso
            FROM aluno_turma at
            JOIN turma t ON t.cod = at.cod_turma)
LOOP
    IF cur.nota >= 8 THEN v_valor_final := cur.valor_curso * 0.9;
    ELSE v_valor_final := cur.valor_curso * 1.1;
    END IF;

    INSERT INTO pagamento (cod_aluno_turma, data, valor) VALUES (cur.cod, SYSDATE-cur.cod, v_valor_final);--Insere o pagamento na data atual (SYSDATE) subtraído do código em dias.

    http.p('Inseriu um pagamento no valor de '||v_valor_final);
END LOOP;
END;
```

Resultados	Explicação	Descrever	Instrução SQL Salva	Histórico
Inseriu um pagamento no valor de 900				
Inseriu um pagamento no valor de 1100				
Inseriu um pagamento no valor de 855				
Inseriu um pagamento no valor de 855				
Inseriu um pagamento no valor de 2530				
Inseriu um pagamento no valor de 2530				
Inseriu um pagamento no valor de 2530				
Inseriu um pagamento no valor de 2070				
Inseriu um pagamento no valor de 2530				
1 linha(s) inserida(s).				
0,01 segundos				

Podemos utilizar o FOR da maneira mais tradicional que é percorrendo uma faixa de valores. Como exemplo, vamos imprimir os números de 6 a 10.

Comando:

```
BEGIN
FOR contador IN 6..10 LOOP
http.p(contador);
END LOOP;
END;
```

c. Cursor

Da mesma maneira que o SELECT INTO permite transferir um dado das tabelas do banco para uma variável, o cursor permite transferir muitos dados ao mesmo tempo para manipulação programática.

Um cursor nada mais é que um ponteiro para uma área de memória alocada no servidor Oracle. Pode ser implícito (criados e gerenciados pelo servidor Oracle) ou explícito (declarado explicitamente pelo programador).

O servidor Oracle aloca uma área de memória privada chamada de “Área de Contexto” para processamento de instruções SQL. As instruções são analisadas (parse) e processadas nesta área. As informações necessárias para o processamento e as informações retornadas após o processamento são todas armazenadas nesta área.

De maneira resumida, o cursor pega informações e coloca numa área em que poderá ser percorrida linha a linha.

Mostraremos neste curso duas sintaxes para uso de cursor. A sintaxe com FOR foi apresentada no item anterior. Agora vamos ver a sintaxe em que o cursor é declarado explicitamente.

Exemplo: O exemplo abaixo extrai as colunas da tabela ALUNO e as coloca num cursor chamado ALUNO_REC.

OBS: O atributo %ROWTYPE especifica um tipo de registro que representa uma linha em uma tabela.

Comando:

```
DECLARE
  CURSOR c1 IS -- declaração do cursor
    SELECT COD, NOME, DATA_NASCIMENTO, IND_SEXO,
           ENDERECO_COMPLETO, TELEFONE, EMAIL
    FROM ALUNO
    WHERE COD < 11;
  ALUNO_REC c1%ROWTYPE;
BEGIN
  OPEN c1; -- abertura do cursor
  LOOP
    FETCH c1 INTO ALUNO_REC; -- recuperação da linha
    EXIT WHEN c1%NOTFOUND;
    http.p(ALUNO_REC.NOME||' - '||ALUNO_REC.EMAIL);
  END LOOP;
END;
```

Comandos SQL

EsquemaAPEX_CURSO

Commit Automático

Linhas

10

Limpar Comando

Localizar Tabelas

SalvarExecutar

```
DECLARE
  CURSOR c1 IS -- declaração do cursor
    SELECT COD, NOME, DATA_NASCIMENTO, IND_SEXO,
           ENDERECO_COMPLETO, TELEFONE, EMAIL
    FROM ALUNO
    WHERE COD < 11;
  ALUNO_REC c1%ROWTYPE;
BEGIN
  OPEN c1; -- abertura do cursor
  LOOP
    FETCH c1 INTO ALUNO_REC; -- recuperação da linha
    EXIT WHEN c1%NOTFOUND;
    http.p(ALUNO_REC.NOME||' - '||ALUNO_REC.EMAIL);
  END LOOP;
END;
```

Resultados	Explicação	Descrever	Instrução SQL Salva	Histórico
Alberto Domingues Esteves - alberto@apex.com Ana Maria Alves - ana@apex.com Carlos Alberto Alves - carlos@apex.com Cristina Pires Domingues - cristina@apex.com David Cunha Goncalves - david@apex.com Eusebio dos Santos Aguiar - eusebio@apex.com Fabrice Stephane Alves Pereira - fabrice@apex.com Fernanda Maria Rodrigues - fernanda@apex.com Joao Paulo Barros da Silva - joao@apex.com Tania Maria Pereira - tania@apex.com				
Instrução processada.				
0,01 segundos				

Você viu muitos conceitos de PL/SQL. Para desenvolver no APEX você não precisa ser um especialista em PL/SQL, mas para construir algumas regras de negócio de cálculos, navegação e relatórios mais sofisticados, um pouco de PL/SQL será importante.

Agora é com você!

1) Execute os códigos apresentados nesta seção.

3. Técnicas de debug

Sabe quando tudo parece estar correto e você não consegue entender porque a sua página ou sua rotina não está fazendo o que deveria fazer? Sempre que se deparar com esse tipo de situação, procure uma técnica de debug. Em geral elas expõem informações que normalmente são ocultadas para que o processamento seja algo mais transparente para o usuário.

Neste módulo veremos algumas delas.

a. Execução de trechos de maneira separada

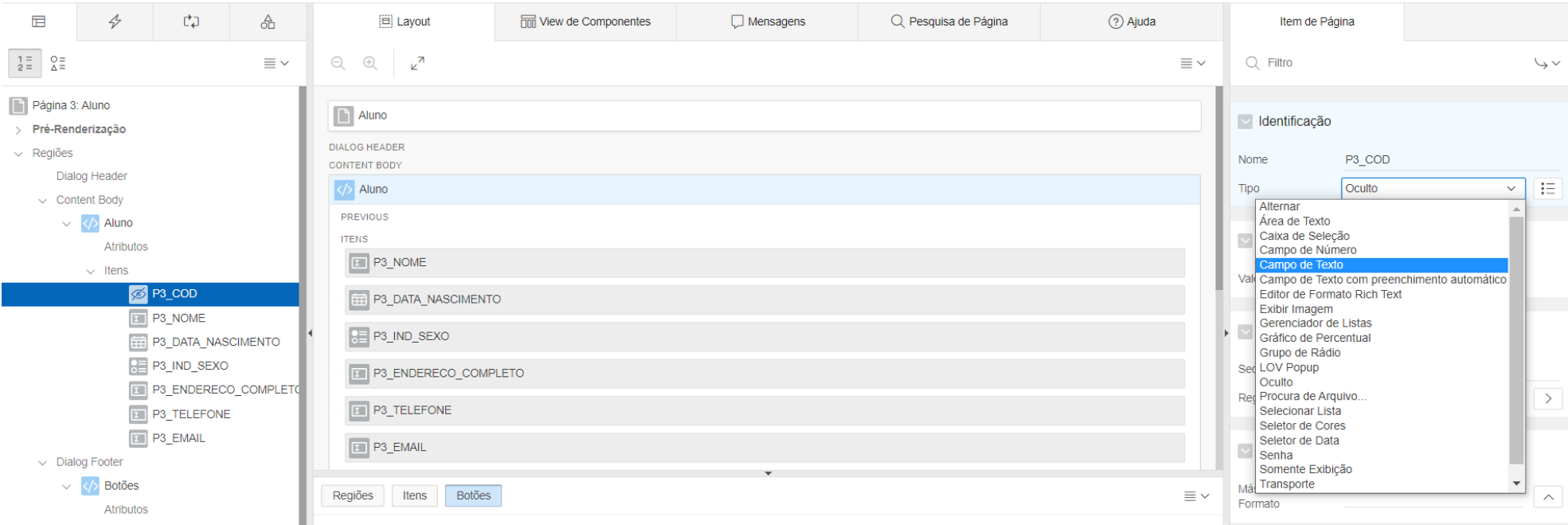
Caso uma consulta ou um código PL/SQL não esteja com a performance ou o comportamento desejado, vá para o **Comandos SQL** e faça a execução do código de maneira separada. Execute trechos de código e veja o resultado até encontrar a origem do problema.

Lembre-se o **Comandos SQL** é seu amigo!

b. Exibir itens ocultos

Uma maneira identificar problemas pode ser exibindo os itens ocultos do APEX. Por exemplo no formulário de alunos, a chave primária é um item Oculto e não é apresentada. Vamos ensinar agora como mostrar o valor do item de forma a facilitar a depuração.

Para isso, acesse o formulário de alunos e altere o item P3_COD de Oculto para "**Campo de Texto**". Salve e volte no relatório e clique em algum registro.



Repare que agora o item que foi criado como Oculto está sendo exibido. Exibir valores de itens ocultos pode ser útil para entender o que está acontecendo na aplicação.

Aluno ✕

Código ?

Nome ?

Data de nascimento ?

Sexo *

Endereço completo ?

Telefone ?

E-mail ?

2

Ana Maria Alves

30/12/1981

Masculino

Feminino

?

SQN 406 bloco D apart 107

ana@apex.com

Cancelar

Excluir

Aplicar Alterações

Ao final retorne o campo para Oculto a fim de evitar que o usuário altere o mesmo.

Esta técnica apresenta o valor que os itens possuem no lado cliente, ou seja, na página HTML.

c. Exibir valores do Estado da Sessão

Uma técnica interessante para entender o que está acontecendo é exibir o estado da sessão. Esta técnica apresenta o valor dos itens que estão no servidor de banco de dados.

Acesse o relatório de alunos, digite uma data de início e uma data de fim. Até este momento estes valores se encontram apenas no cliente, ou seja, na página HTML.

Quando clicamos no botão "**Pesquisar**", os valores são enviados para o servidor e ficam disponíveis no estado da sessão do APEX.

Vamos agora verificar os valores no estado da sessão?

Para isso, acesse na barra do desenvolvedor a opção "**Sessão**".

Minha Escola

aluno

Início

Aluno

Edição em grade de alunos

Professor

Turma

Aluno

Filtro

Data de início

01/01/1980

Data de fim

31/12/1985

Pesquisar

Nome

Data de nascimento

Sexo

Endereço completo

E-mail

Ana Maria Alves

30/12/1981

Feminino

SQN 406 bloco D apart 107

ana@apex.com

Carlos Alberto Alves

27/03/1980

Masculino

QNL 30 TAGUATINGA NORTE DF

carlos@apex.com

David Cunha Goncalves

03/04/1984

Masculino

SQS 116 BLOCO E APT 401

david@apex.com

Eusebio dos Santos Aguiar

15/05/1985

Masculino

SHCES QUADRA 1405 BLOCO H APARTAMENTO 304

eusebio@apex.com

Tania Maria Pereira

11/08/1984

Feminino

HIGS 709 BLOCO C CASA 21 ASA SUL

tania@apex.com

1 - 5

Início

Aplicativo 600186

Editar Página 2

Sessão

Exibir Depuração

Depurar

Informações da Página

Edição Rápida

Rolador do Tema

Após isso, por padrão o valor dos itens da página atual são apresentados, mas você pode escolher outra página; pode visualizar itens de aplicação; pode visualizar todo o estado da sessão e coletas. (Coletas são conhecidos como APEX Collections e não fazem parte do escopo deste curso!)

Estado da Sessão - Google Chrome

contas.tcu.gov.br/ords/f?p=4000:34:862103477923:PAGE:NO:34:F4000_P34_SESSION,F4000_P34_FLOW,F4000_P34_PAGE,FB_FLOW_ID:110463768737704,6...

Itens

Páginas

Consultas

Tabelas

PL/SQL

Depurar

Sessão

Erros

Página

2

Linhas

50

Localizar

Exibir

Itens da Página

Definir

Aplicativo: 600186 Minha Escola

Sessão: 110463768737704

Usuário: ALUNO

Espaço de Trabalho: 1366930195384506648

Idioma do Browser: pt-br

Itens da Página

Aplicativo	Página	Nome do Item	Visualizar	Valor do Item	Status	Criptografado
600186	2	P2_DATA_INICIO	Seletor de Data	01/01/1980	Inserido	Não
600186	2	P2_DATA_FIM	Seletor de Data	31/12/1985	Atualizado	Não

1 - 2

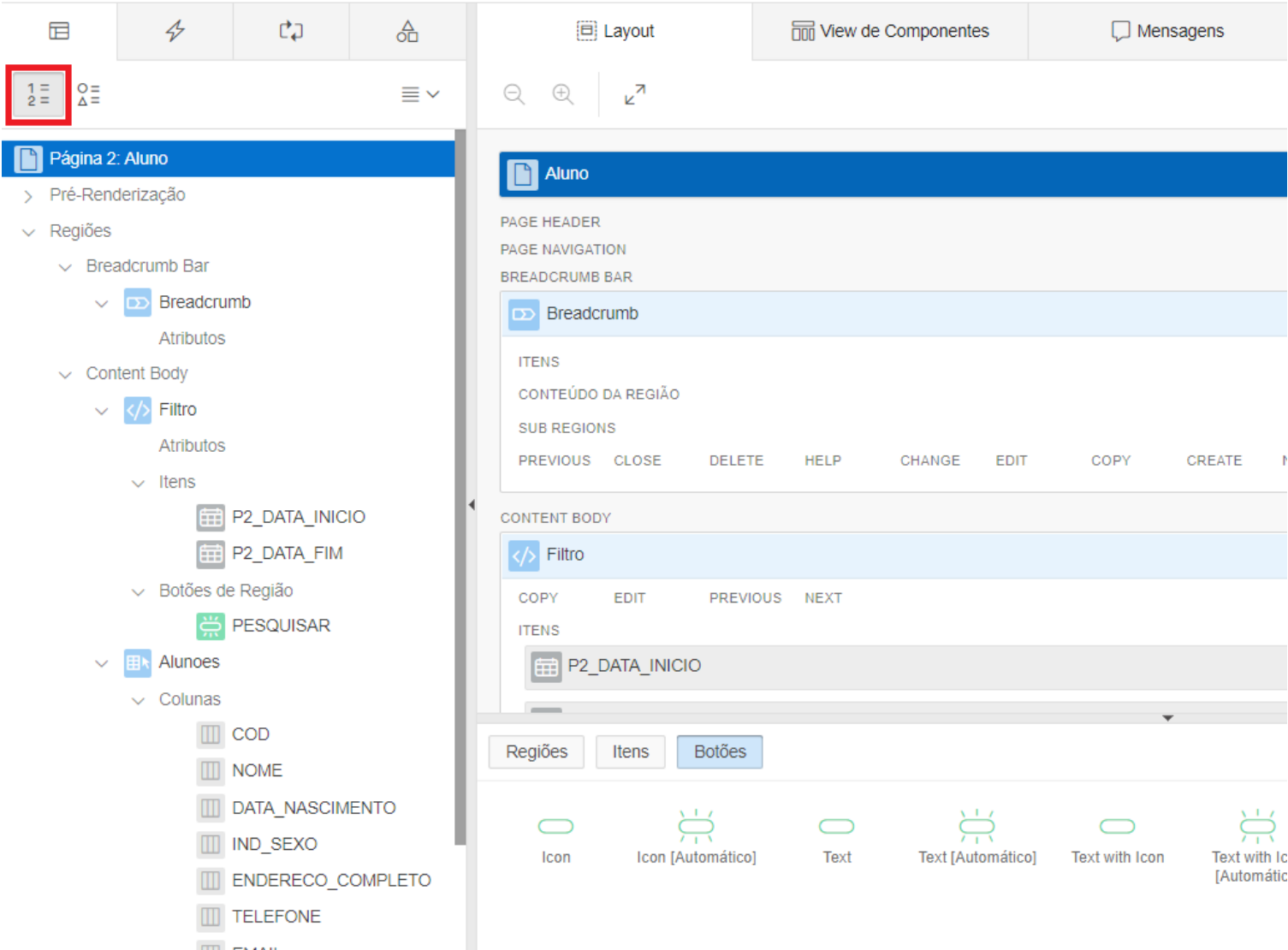
Repare que o valor dos itens de data que enviamos para o servidor estão no Estado da Sessão.

d. Descobrimdo a ordem correta das ações

Agora imagine outro problema comum: uma ação que deveria acontecer em certo momento, está acontecendo antes ou depois do momento desejado.

Como saber e como resolver?

Primeiro confirme se as ações desejadas foram criadas na ordem correta. O Designer de página apresenta a ordem de execução de cada elemento tanto na renderização quanto no processamento da página. Marque o ícone que apresenta os itens na ordem de processamento.



Repare que neste exemplo, a primeira região a ser renderizada é o "Breadcrumb", depois a "Filtro" e dentro desta região, primeiro é renderizado o item P2_DATA_INICIO, depois o item P2_DATA_FIM e então o botão "PESQUISAR". Logo em seguida é renderizada a região de Alunos.

Esta técnica é bem útil, pois a ordem dos elementos pode fazer diferença, principalmente quando forem processos de página.

e. Utilizando o mecanismo de depuração do APEX

O APEX possui um mecanismo de depuração que gera mensagens de debug.

Primeiro você deve ativar a depuração e depois exibir as mensagens geradas.

Para isso, clique no link "**Depurar**" para que a tela seja recarregada no modo de depuração. A partir de agora, todas as ações realizadas pelo APEX gerarão mensagens (log de ações) para permitir a depuração.

Minha Escola

aluno

Início

Aluno

Edição em grade de alunos

Professor

Turma

Aluno

Filtro

Data de início

01/01/1980

Data de fim

31/12/1985

Pesquisar

Q

Ir

Ações

Redefinir

Criar

	Nome	Data de nascimento	Sexo	Endereço completo	E-mail
	Ana Maria Alves	30/12/1981	Feminino	SQN 406 bloco D apart 107	ana@apex.com
	Carlos Alberto Alves	27/03/1980	Masculino	QNL 30 TAGUATINGA NORTE DF	carlos@apex.com
	David Cunha Goncalves	03/04/1984	Masculino	SQS 116 BLOCO E APT 401	david@apex.com
	Eusebio dos Santos Aguiar	15/05/1985	Masculino	SHCES QUADRA 1405 BLOCO H APARTAMENTO 304	eusebio@apex.com
	Tania Maria Pereira	11/08/1984	Feminino	HIGS 709 BLOCO C CASA 21 ASA SUL	tania@apex.com

1 - 5

Início

Aplicativo 600186

Editar Página 2

Sessão

Exibir Depuração

Depurar

Informações da Página

Edição Rápida

Rolador do Tema

Obs: Quando a depuração está ligada, a página demora mais a fazer as ações, pois além do tempo de executar a ação, o APEX também está gerando mensagens das ações executadas.

Após isso, clique na opção **"Exibir Depuração"** na barra do desenvolvedor para acessar os dados das mensagens de depuração geradas.

Depurar Dados da Mensagem - Google Chrome

contas.tcu.gov.br/ords/f?p=4000:19:862103477923:::RIR,19:IR_APPLICATION_ID,IR_PAGE_ID:600186,2

Itens

Páginas

Consultas

Tabelas

PL/SQL

Depurar

Sessão

Erros

Q

Ir

Ações

Redefinir

Aplicativo = 600186

Página = 2

Exibir Identificador	Id da Sessão	Usuário	Aplicativo	Página	Informações do Caminho	Entradas	Timestamp	Segundos
281875072	110463768737704	ALUNO	600186	2	show	179	4 segundos atrás	1,7192
281875069	110463768737704	ALUNO	600186	2	accept PESQUISAR	62	5 segundos atrás	0,0311
281875066	110463768737704	ALUNO	600186	2	show	179	15 segundos atrás	1,7404
281875063	110463768737704	ALUNO	600186	2	show	179	Há 6 minutos	1,9761

1 - 4

Identifique qual a execução que deseja detalhar as mensagens de depuração e clique no número identificador desejado. Com isso vamos visualizar cada ação executada pelo APEX na ordem realizada e o tempo de execução de cada ação, além de outras informações.

Depurar Dados da Mensagem - Google Chrome

contas.tcu.gov.br/ords/f?p=4000:939:862103477923::NO:939:P939_PAGE_VIEW_ID:281875072

Ajuda

Itens

Páginas

Consultas

Tabelas

PL/SQL

Depurar

Sessão

Erros

Identificador de View da Página

281875072

?

Definir

Selecionar Identificador de Exibição da Página

Redefinir

Aplicativo:

600186

?

Página:

2

?

Tempo Decorrido:

1,71924

?

Tempo Máximo de Execução

1,63273

?

Q

Ir

Ações

Decorrido	Execução	Mensagem	Nível	Gráfico
0.00365	0.00087	Reset NLS settings	4	
0.00452	0.00061	alter session set NLS_LANGUAGE='BRAZILIAN PORTUGUESE' NLS_TERRITORY='BRAZIL' NLS_CALENDAR='GREGORIAN' NLS_SORT='WEST_EUROPEAN' NLS_COMP='BINARY'	4	
0.00513	0.00004	...NLS: Set Decimal separator=", "	4	
0.00517	0.00052	...NLS: Set NLS Group separator="."	4	
0.00569	0.00005	...NLS: Set g_nls_date_format="DD/MM/RR"	4	
0.00574	0.00057	...NLS: Set g_nls_timestamp_format="DD/MM/RR HH24:MI:SSXFF"	4	
0.00631	0.00186	...NLS: Set g_nls_timestamp_tz_format="DD/MM/RR HH24:MI:SSXFF TZR"	4	
0.00817	0.00007	NLS of database and client differs, character set conversion needed	4	
0.00824	0.00011	...Setting session time_zone to -03:00	4	
0.00834	0.00459	R E Q U E S T show	4	
0.01293	0.00085	Language derived from: FLOW_PRIMARY_LANGUAGE, current browser language: pt-br	4	
0.01378	0.00004	alter session set nls_language='BRAZILIAN PORTUGUESE' nls_territory='BRAZIL'	4	

Com essas informações pode ser mais fácil descobrir o que está acontecendo.

f. Identificar erros de compilação de objetos PL/SQL

Sempre que houver uma funcionalidade nativa, evite programar em PL/SQL, pois o melhor é utilizar os recursos nativos. Entretanto, caso precise criar um objeto PL/SQL, quando criamos funções ou procedimentos podem acontecer erros de compilação.

Crie a seguinte função no Comandos SQL e veja que o APEX informa que houve um erro de compilação.

Comando:

```
create or replace FUNCTION f_valor_pago (p_cod_aluno_turma NUMBER)
RETURN NUMBER
IS
    v_total    NUMBER;
BEGIN
    SELECT SUM (valor)
    INTO v_total
    FROM pagamentos
    WHERE cod_aluno_turma = p_cod_aluno_turma;
    RETURN v_total;
END;
```

Comandos SQL

Esquema

APEX_CURSO

?

Commit Automático

Linhas

10

?

Limpar Comando

Localizar Tabelas

Salvar

Executar

create or replace FUNCTION f_valor_pago (p_cod_aluno_turma NUMBER)

RETURN NUMBER

IS

v_total NUMBER;

BEGIN

SELECT SUM (valor)

INTO v_total

FROM pagamentos

WHERE cod_aluno_turma = p_cod_aluno_turma;

RETURN v_total;

END;

Resultados

Explicação

Descrever

Instrução SQL Salva

Histórico

Erro na linha 6: PL/SQL: SQL Statement ignored

Para corrigir este erro de compilação, acesse o **SQL Workshop** > **Browser de Objetos** > **Funções** e clique em "**Salvar e Compilar**". Os eventuais erros aparecerão em destaque. O número com a linha do erro fica clicável, destacando a linha problemática diretamente no código.

↑ Browser de Objetos

Funções

F_QTD_PARTICIPANTE_TURMA

F_VALOR_PAGO

F_VALOR_PAGO

Código

Dependências

Erros

Concessões

Salvar e Compilar

Fonte para Download

Eliminar

Falha de compilação; linha 8 (13:03:04)
PL/SQL: ORA-00942: a tabela ou view não existeFalha de compilação; linha 6 (13:03:04)
PL/SQL: SQL Statement ignored

A

1

create or replace FUNCTION f_valor_pago (p_cod_aluno_turma NUMBER)

2

RETURN NUMBER

3

IS

4

v_total NUMBER;

5

BEGIN

6

SELECT SUM (valor)

7

INTO v_total

8

FROM pagamentos

9

WHERE cod_aluno_turma = p_cod_aluno_turma;

10

11

RETURN v_total;

12

END;

13

Agora é com você!

- 1) Execute as técnicas mostradas nesta seção.
- 2) Corrija o problema na função f_valor_pago e recompile a função.

4. Aplicando as regras de negócio

Neste momento, muito da nossa aplicação Minha Escola já está pronta. Logo, devemos voltar ao nosso problema e verificar se todas as regras do negócio já foram atendidas. Retornando à lição 2, veremos as seguintes regras de negócio:

RN 01: Toda turma tem um ou mais professores, uma sala de aula, e vários alunos.

RN 02: As notas são classificadas de 0 (sem rendimento) a 10 (excelente rendimento).

RN 03: O aluno só pode entrar em uma turma que possua vagas.

RN 04: Todos os alunos têm nota em cada uma das turmas que participa.

RN 05: Todos os valores monetários são em reais (R\$), podendo eventualmente ser gratuitos.

Analizando e aplicando as regras de negócio

Vamos agora entender as regras de negócio. Para isso, o ideal é pegar apenas uma regra por vez e tentar desmembrá-las no menor tamanho possível. Faremos isso agora, regra por regra.

4.1. Regra de Negócio RN 01

"Toda turma tem um ou mais professores, uma sala de aula, e vários alunos."

Olhando no nosso modelo de dados, podemos identificar que essa regra atinge as tabelas TURMA, PROFESSOR_TURMA e ALUNO_TURMA.

a. Primeira parte

Para checar se todas as condições já foram atendidas no momento da criação do nosso modelo, vamos testar a primeira parte da regra, “**Toda a turma tem um ou mais professores**”.

Browser de Objetos

Tabelas

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

Esquema

APEX_CURSO

PROFESSOR_TURMA

Tabela

Dados

Índices

Modelo

Constraints

Concessões

Estatísticas

Padrões da UI

Triggers

Dependências

SQL

Adicionar Coluna

Modificar Coluna

Renomear Coluna

Eliminar Coluna

Renomear

Copiar

Eliminar

Truncar

Criar Tabela de Consulta

Nome da Coluna	Tipo de Dados	Anulável	Padrão	Chave Primária
COD	NUMBER	Não	-	1
COD_PROFESSOR	NUMBER	Não	-	-
COD_TURMA	NUMBER	Não	-	-

Fazer Download

Imprimir

Repare na imagem acima, que a tabela PROFESSOR_TURMA permite a associação de mais de um professor a uma turma e vemos que os campos COD_PROFESSOR e COD_TURMA estão com o preenchimento obrigatório (Não Anulável). Com isso temos a possibilidade (pelas constraints) que cada turma tenha no mínimo um professor o que já nos possibilita dizer que essa primeira parte da regra está atendida. Caso haja mais de um professor para esta turma, teremos mais um registro na tabela repetindo o código da turma e alterando o do professor.

b. Segunda parte

A segunda parte da regra de negócio 1 diz que toda turma tem que ter "**uma sala de aula**".

Browser de Objetos

Tabelas

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

Esquema

APEX_CURSO

TURMA

Tabela

Dados

Índices

Modelo

Constraints

Concessões

Estatísticas

Padrões da UI

Triggers

Dependências

SQL

Adicionar Coluna

Modificar Coluna

Renomear Coluna

Eliminar Coluna

Renomear

Copiar

Eliminar

Truncar

Criar Tabela de Consulta

Nome da Coluna	Tipo de Dados	Anulável	Padrão	Chave Primária
COD	NUMBER	Não	-	1
DESCRICAO	VARCHAR2(50)	Não	-	-
PERIODO	VARCHAR2(50)	Não	-	-
HORARIO	VARCHAR2(50)	Não	-	-
SALA	VARCHAR2(20)	Sim	-	-
QTD_VAGAS	NUMBER	Não	-	-
VALOR_CURSO	NUMBER	Não	-	-
IND_SITUACAO	CHAR(1)	Não	-	-

Fazer Download

Imprimir

Como podemos observar, o campo SALA da tabela TURMA aceita valor nulo, o que contradiz a nossa regra de negócio. Para atendê-la, iremos alterar a tabela, impossibilitando assim a presença de valores nulos.

Passo 1 – Entre no **Browser de Objetos** e escolha a tabela TURMA. Em seguida clique no botão **Modificar Coluna**. Escolha a Coluna "**SALA (VARCHAR2)**" e opção Anulável para "**NOT NULL (requer um valor)**".

↑

Browser de Objetos

Esquema

APEX_CURSO

Tabelas

Q

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

TURMA

+ ▼

Modificar Coluna

Use esta página para aumentar o tamanho dos tipos de dados da coluna de caracteres.

Esquema:

APEX_CURSO

Tabela:

TURMA

Coluna

SALA (VARCHAR2)

Tipo de Dados

VARCHAR2

Tamanho

20

Precisão

Escala

Anulável

NOT NULL (requer um valor)

NOT NULL (requer um valor)

NULL (não requer um valor)

Cancelar

Próximo >

Passo 2 – Clique no botão "**Próximo** >" em seguida finalize clicando no botão "**Finalizar**". Com isso você verá que a nova estrutura da tabela ficará como mostrada na imagem abaixo.

↑

Browser de Objetos

Esquema

APEX_CURSO

Tabelas

Q

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

TURMA

+ ▼

Tabela

Dados

Índices

Modelo

Constraints

Concessões

Estatísticas

Padrões da UI

Triggers

Dependências

SQL

Adicionar Coluna

Modificar Coluna

Renomear Coluna

Eliminar Coluna

Renomear

Copiar

Eliminar

Truncar

Criar Tabela de Consulta

Nome da Coluna	Tipo de Dados	Anulável	Padrão	Chave Primária
COD	NUMBER	Não	-	1
DESCRICAO	VARCHAR2(50)	Não	-	-
PERIODO	VARCHAR2(50)	Não	-	-
HORARIO	VARCHAR2(50)	Não	-	-
SALA	VARCHAR2(20)	Não	-	-
QTD_VAGAS	NUMBER	Não	-	-
VALOR_CURSO	NUMBER	Não	-	-
IND_SITUACAO	CHAR(1)	Não	-	-

Fazer Download

Imprimir

Agora é sua vez!

Agora implemente esta alteração no banco.

c. Terceira parte

A última parte da regra diz que a turma deve ter "**vários alunos**". Para isso checaremos a estrutura da tabela ALUNO_TURMA e verificaremos se a coluna COD_ALUNO é de preenchimento obrigatório ou não.

↑

Browser de Objetos

Esquema

APEX_CURSO

?

Tabelas

Q

↶

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

ALUNO_TURMA

+ ▾

Tabela

Dados

Índices

Modelo

Constraints

Concessões

Estatísticas

Padrões da UI

Triggers

Dependências

SQL

Adicionar Coluna

Modificar Coluna

Renomear Coluna

Eliminar Coluna

Renomear

Copiar

Eliminar

Truncar

Criar Tabela de Consulta

Nome da Coluna	Tipo de Dados	Anulável	Padrão	Chave Primária
COD	NUMBER	Não	-	1
COD_ALUNO	NUMBER	Não	-	-
COD_TURMA	NUMBER	Não	-	-
NOTA	NUMBER	Sim	-	-

Fazer Download

|

Imprimir

Como vimos, todas as regras citadas na RN 01 estão atendidas!

4.2. Regra de Negócio RN 02

"As notas são classificadas de 0 (sem rendimento) a 10 (excelente rendimento)."

As notas devem ser limitadas entre os valores 0 e 10.

Para isso vamos ajustar os valores permitidos na aplicação.

Obs: O ideal é também criar uma constraint no banco de dados também, mas neste curso vamos apenas criar na aplicação.

Passo 1 - Acesse a página de "Cadastro de aluno em turma", e edite o item PX_NOTA.

The screenshot shows the APEX Designer de Página interface. On the left, the 'Regiões' pane shows the 'Cadastro de aluno em turma' region selected. The 'Itens' pane shows the 'P10_NOTA' item selected. The main pane displays the 'Valor Máximo' configuration for the item. The 'Definições' section is highlighted with a red box, showing the 'Valor Mínimo' set to 0 and the 'Valor Máximo' set to 10. The 'Identificação' section shows the item name as 'P10_NOTA' and the type as 'Campo de Número'. The 'Label' section shows the label as 'Nota final'.

Passo 2 - Para executar a página, acesse no menu da aplicação: "**Turma**" > "**Matriculados por turma**". Clique no ícone de edição para inserir a nota que está com valor nulo.

The screenshot shows the application interface. The top navigation bar has 'Minha Escola' and a user profile 'aluno'. The left sidebar has a menu with 'Início', 'Aluno', 'Professor', 'Turma', and 'Matriculados por turma'. The main content area shows the 'Matriculados por turma' page. The page has a search bar, a 'Ir' button, and an 'Ações' dropdown. A table lists students and their courses. The table has columns for 'Aluno', 'Turma', and 'Nota'. The 'Nota' column shows values 8, 8, 10, 0, 2, 4, 6, and 8. The 'Turma' column shows 'Curso de PL/SQL' and 'Curso de APEX'. The 'Aluno' column shows names like 'Cristina Pires Domingues', 'Carlos Alberto Alves', 'Alberto Domingues Esteves', 'Ana Maria Alves', and 'David Cunha Goncalves'. A 'Nova matrícula' button is in the top right corner.

	Aluno	Turma	Nota
	Cristina Pires Domingues	Curso de PL/SQL	8
	Carlos Alberto Alves	Curso de PL/SQL	8
	Alberto Domingues Esteves	Curso de APEX	
	Ana Maria Alves	Curso de APEX	10
	Alberto Domingues Esteves	Novo curso de APEX	0
	Ana Maria Alves	Novo curso de APEX	2
	Carlos Alberto Alves	Novo curso de APEX	4
	Cristina Pires Domingues	Novo curso de APEX	6
	David Cunha Goncalves	Novo curso de APEX	8

Passo 3 - No formulário modal, tente inserir a nota 20 (fora do intervalo válido) e veja que um erro é apresentado.

Minha Escola

Início

Aluno

Professor

Turma

Professores por turma

Matriculados por turma

Turma \

Matriculados por turma

Cadastro de aluno em turma

Aluno

Alberto Domingues Esteves

Turma

Curso de APEX

Nota final

20

Cancelar

Excluir

Aplicar Alterações

Ocorreu 1 erro

Nota final não está entre o intervalo válido de 0 e 10.

Nova matrícula

	Nota
	8
	8
	10
	0
	2
	4
	6
	8

1 - 9

Passo 4 - Agora insira a nota 10 e clique "Aplicar Alterações". Certifique-se de que não nenhuma nota inválida e nem nula.

Minha Escola

Início

Aluno

Professor

Turma

Professores por turma

Matriculados por turma

Turma \

Matriculados por turma

Q

Ir

Ações

Nova matrícula

	Aluno	Turma	Nota
	Cristina Pires Domingues	Curso de PL/SQL	8
	Carlos Alberto Alves	Curso de PL/SQL	8
	Alberto Domingues Esteves	Curso de APEX	10
	Ana Maria Alves	Curso de APEX	10
	Alberto Domingues Esteves	Novo curso de APEX	0
	Ana Maria Alves	Novo curso de APEX	2
	Carlos Alberto Alves	Novo curso de APEX	4
	Cristina Pires Domingues	Novo curso de APEX	6
	David Cunha Goncalves	Novo curso de APEX	8

1 - 9

4.3. Regra de Negócio RN 03

"O aluno só pode entrar em uma turma que possua vagas."

Essa é a regra mais complexa deste sistema. Para facilitar a compreensão da solução, iremos subdividir esta regra em problemas menores.

Os subproblemas são:

- a) Qual o número de vagas de uma turma?
- b) Quantos alunos inscritos há nesta turma?
- c) O número de vagas é maior que a quantidade de inscritos?

Vamos responder agora as duas primeiras perguntas:

a) Qual o número de vagas de uma turma?

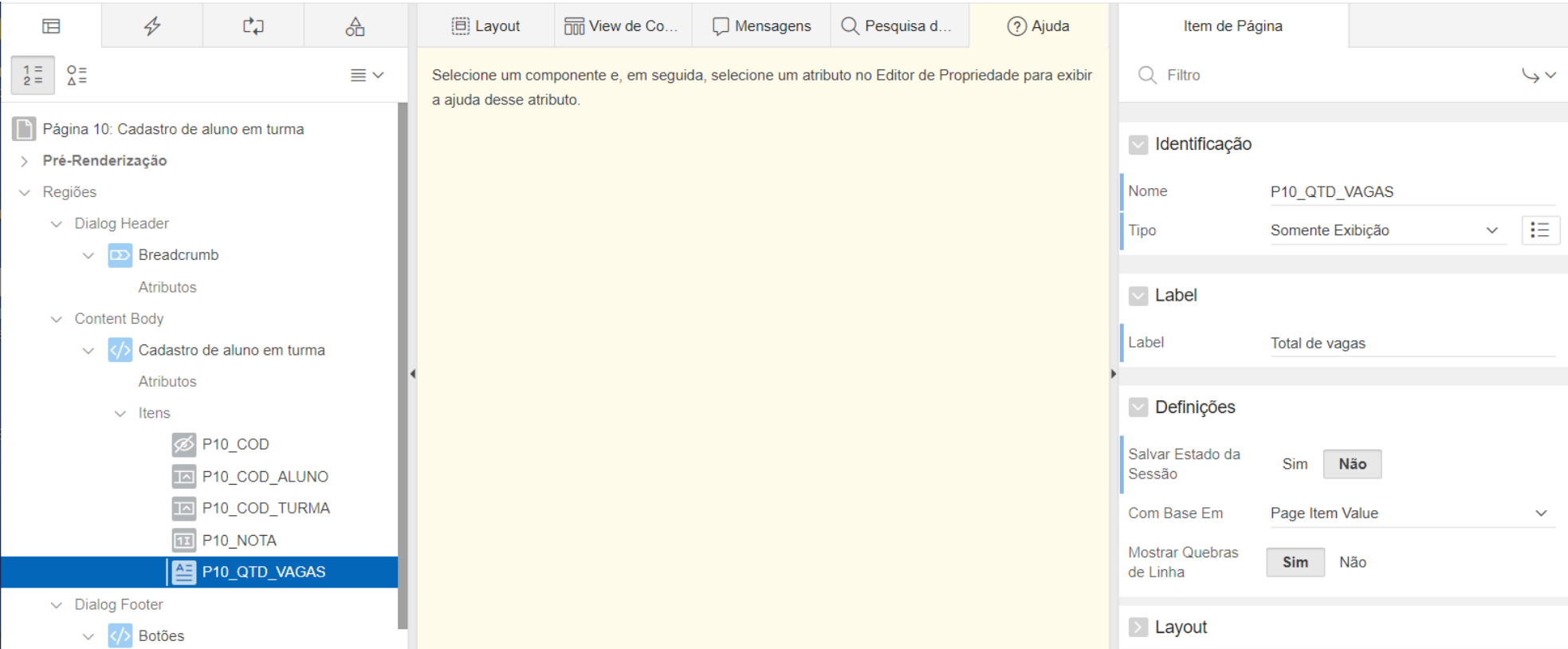
b) Quantos alunos inscritos há nesta turma?

Para saber o número de vagas de uma turma, devemos verificar o campo QTD_VAGAS na tabela TURMA.

Para saber a quantidade de inscritos, utilizamos a função **F_QTD_PARTICIPANTE_TURMA** que foi criada anteriormente.

Agora é com você!

Passo 1 - Acesse o formulário de "**Cadastro de aluno em turma**", na região de mesmo nome crie um item chamado P10_QTD_VAGAS (Obs: o número da página pode variar). Defina o Tipo: "**Somente Exibição**", Label: "**Total de vagas**", Salvar Estado da Sessão: "**Não**" e na seção Origem o Tipo: "**Nulo**".



1=

2=

Δ=

≡

▼

Página 10: Cadastro de aluno em turma

> Pré-Renderização

▼ Regiões

▼ Dialog Header

▼ Breadcrumb

Atributos

▼ Content Body

▼ </> Cadastro de aluno em turma

Atributos

▼ Itens

P10_COD

P10_COD_ALUNO

P10_COD_TURMA

P10_NOTA

P10_QTD_VAGAS

▼ Dialog Footer

▼ </> Botões

Atributos

▼ Botões de Região

Layout

View de Co...

Mensagens

Pesquisa d...

Ajuda

Tipo

Selecione o tipo de origem no qual se baseie o valor desse item.

Observação: Se a opção **Origem: Usado** estiver definida como *Apenas quando o valor atual em estado de sessão for nulo*, a origem não será utilizada se um valor tiver sido definido no estado de sessão para este item.

As opções disponíveis incluem:

Valor Estático

Definir como o texto inserido no *Valor Estático*.

Coluna do Banco de Dados

Definido para o valor de coluna inserido na *Coluna do Banco de Dados* que é recuperado de um processo de extração de linha incorporado.

Item

Definir como o valor mantido no estado da sessão para o *Item* selecionado.

Consulta SQL (retornar valor único)

Definir como o primeiro valor retornado da *Consulta SQL* informada.

Consulta SQL (retornar valor separado por dois-pontos)

Definir como um valor único separado por vírgulas, que concatena cada valor de linha, retornado da *Consulta SQL* informada.

Expressão PL/SQL

Definir como o valor que resulta da execução da *Expressão PL/SQL* informada.

Corpo da Função PL/SQL

Definir como o valor retornado do *Corpo de Função PL/SQL* informado.

Item de Página

Filtro

↶

▼

▼ Identificação

Nome

P10_QTD_VAGAS

Tipo

Somente Exibição

▼

⋮

> Label

> Definições

> Layout

> Aparência

> Lista de Valores

> Avançado

▼ Origem

Tipo

Nulo

▼

⋮

Usado

Apenas quando o valor atual em estad

▼

Passo 2 - Clique com o botão direito do mouse sobre o item P10_QTD_VAGAS e escolha a opção de "**Duplicar**". No item duplicado, coloque o Nome: "**P10_QTD_INSCRITOS**" e Label: "**Alunos inscritos**".

1=

2=

Δ=

≡

▼

Página 10: Cadastro de aluno em turma

> Pré-Renderização

▼ Regiões

▼ Dialog Header

▼ Breadcrumb

Atributos

▼ Content Body

▼ </> Cadastro de aluno em turma

Atributos

▼ Itens

P10_COD

P10_COD_ALUNO

P10_COD_TURMA

P10_NOTA

P10_QTD_VAGAS

P10_QTD_INSCRITOS

▼ Dialog Footer

Layout

View de Co...

Mensagens

Pesquisa d...

Ajuda

Label

Informe o label do campo para o item da página. O label será exibido na página somente se o item for exibido. Dependendo do alinhamento e da exibição da página, o label será exibido em sua própria célula da tabela HTML ou na mesma célula.

Labels de itens podem incluir atalhos usando a sintaxe "SHORTCUT_NAME".

Informações Adicionais

- Tipo: Texto
- Substituições Suportadas: Aplicativo, Itens da Página e Variáveis do Sistema

Item de Página

Filtro

↶

▼

▼ Identificação

Nome

P10_QTD_INSCRITOS

Tipo

Somente Exibição

▼

⋮

▼ Label

Label

Alunos inscritos

> Definições

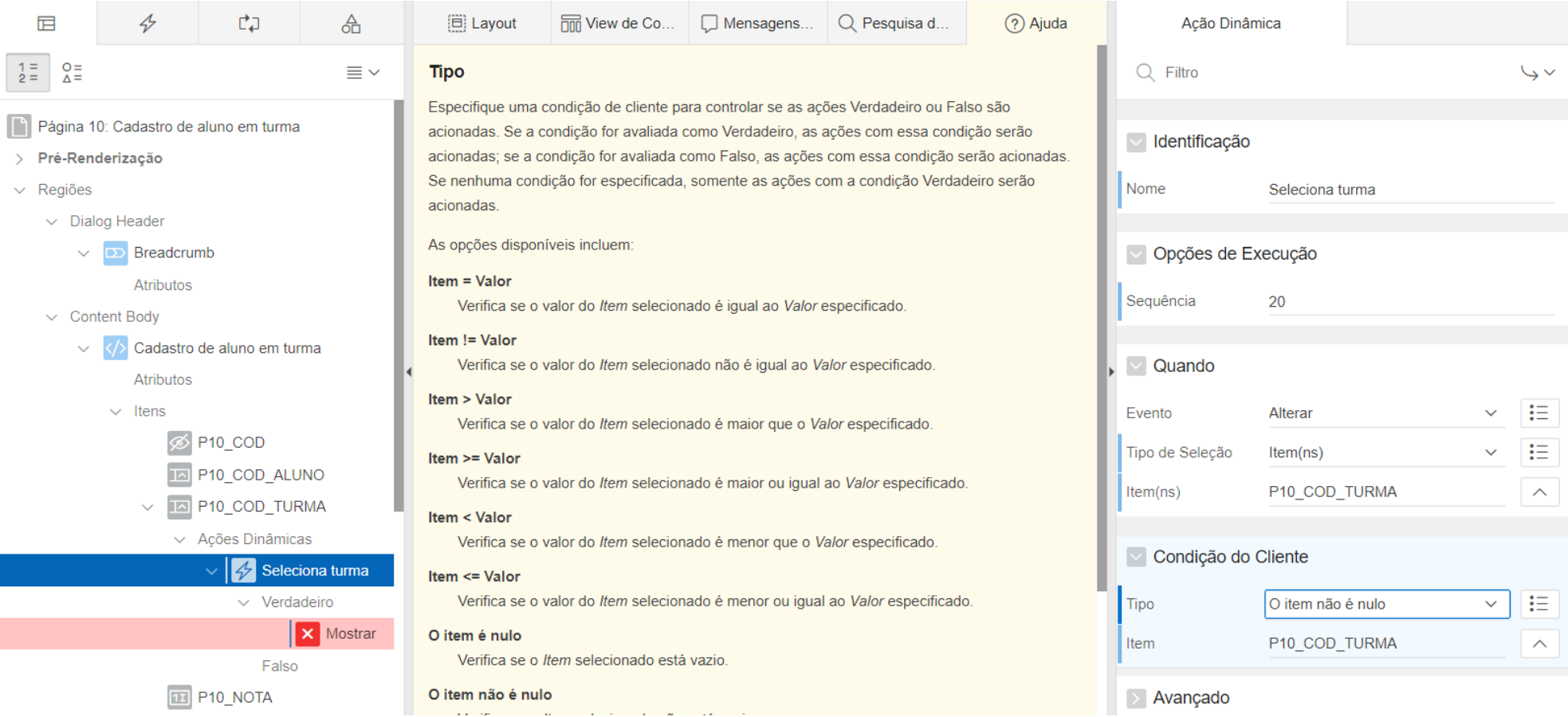
> Layout

> Aparência

> Lista de Valores

> Avançado

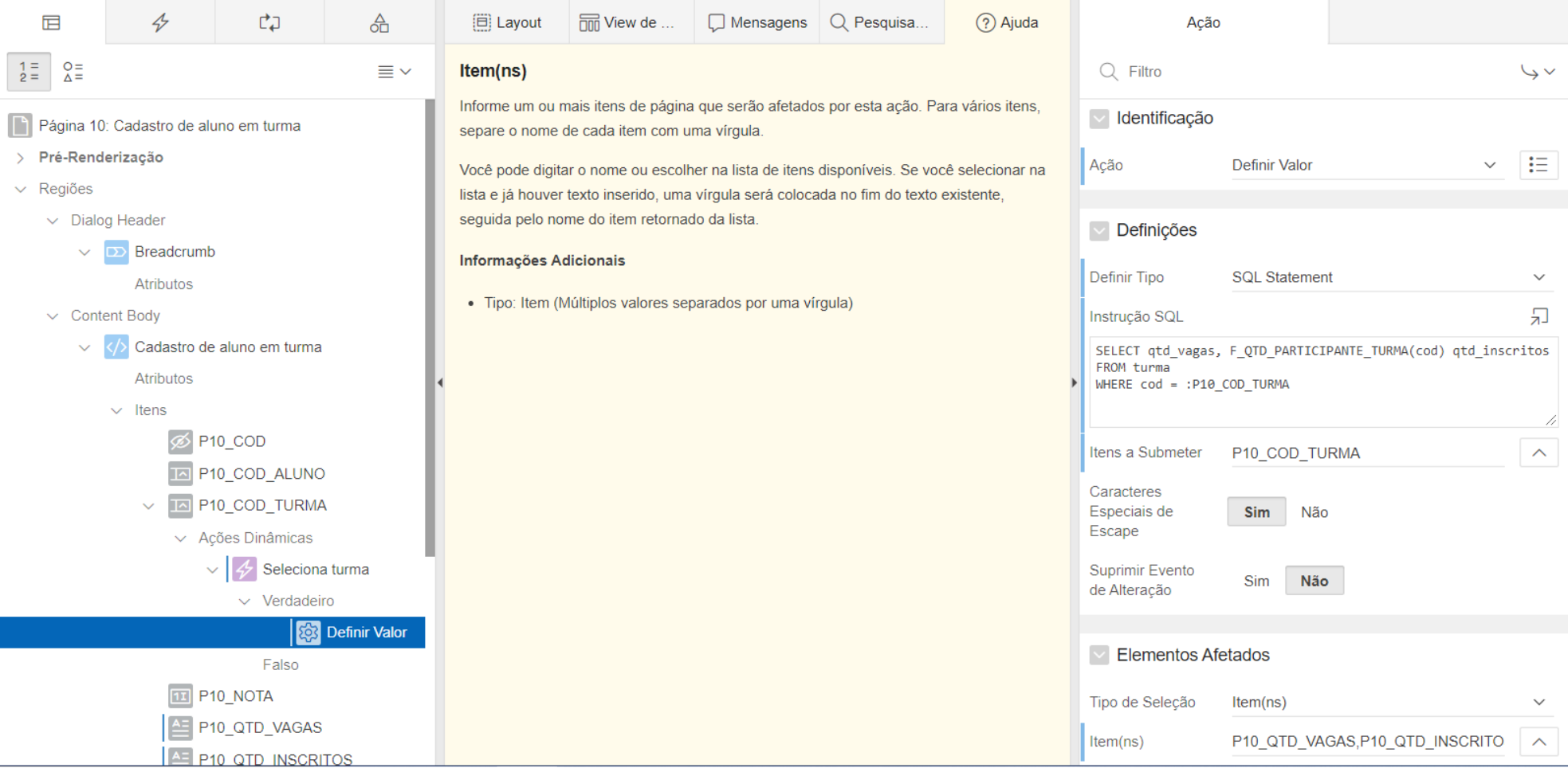
Passo 3 - Agora vamos criar a ação dinâmica para definir o valor dos itens criados. Para isso, clique com o botão direito do mouse no item "**P10_COD_TURMA**" e selecione a opção "**Criar Ação Dinâmica**". Defina na Ação dinâmica o Nome : "**Seleciona turma**", confirme se na seção Quando está definido o Evento: "**Alterar**", Tipo de Seleção: "**Item(ns)**", Item(ns): "**P10_COD_TURMA**".



Passo 4 - Clique na ação verdadeira (por padrão é criada como "Mostrar"), defina a Ação para "**Definir Valor**", em Definir Tipo selecione "**SQL Statement**" em Instrução SQL copie e cole o código abaixo, em Itens a Submeter escolha "**P10_COD_TURMA**", no Tipo de Seleção escolha os itens: "**P10_QTD_VAGAS,P10_QTD_INSCRITOS**".

Instrução SQL:

```
SELECT qtd_vagas, F_QTD_PARTICIPANTE_TURMA(cod) qtd_inscritos
FROM turma
WHERE cod = :P10_COD_TURMA
```



A consulta SQL é executada no servidor, portanto, precisamos submeter o valor do item P10_COD_TURMA que foi selecionado no cliente (browser). Como a consulta SQL retorna dois valores, estes são definidos nos itens selecionados na seção "Elementos Afetados".

Passo 5 - Execute a aplicação, acesse o formulário de Cadastro de aluno em turma, selecione uma Turma e veja que a aplicação agora calcula e mostra o Total de vagas e a quantidade de alunos inscritos.

Minha Escola

Início

Aluno

Professor

Turma

Professores por turma

Matriculados por turma

Turma \

Matriculados por turma \ Cadastro de aluno em turma

Aluno

Turma

Novo curso de APEX

Nota final

Total de vagas

25

Alunos inscritos

5

Cancelar

Criar

Nova matrícula

	Nota
	8
	8
	10
	10
	0
	2
	4
	6
	8

1 - 9

Pronto! As duas primeiras perguntas estão respondidas.

Agora vamos responder à pergunta:

c) O número de vagas é maior que a quantidade de inscritos?

Para fazer isso faremos uma comparação do número de vagas com o número de inscritos e caso não esteja mostraremos uma mensagem de erro.

Agora é com você!

Passo 1 - Ainda na página de formulário "**Cadastro de aluno em turma**", acesse a aba "**Processamento**", clique com o botão direito do mouse em "**Validando**" e selecione "**Criar Validação**".

Aplicativo 600186 \ Designer de Página

10

Ir

Salvar

Layout

View de ...

Mensagens

Pesquisa...

Ajuda

Selecione um componente e, em seguida, selecione um atributo no Editor de Propriedade para exibir a ajuda desse atributo.

Editor de Propriedades

Filtro

Nenhum componente selecionado no momento.

Após a Submissão

Validando

Processamento

Processos

Process Row of ALUNO_TURMA

reset page

Close Dialog

Após o Processamento

Ajax Callback

Criar Ramificação

Criar Validação

Expandir Todos os Itens a Seguir

Contrair Todos os Itens a Seguir

Passo 2 - Na validação criada defina o Nome: "**Valida quantidade de vagas**", Tipo: "**Linhas retornadas**", em Consulta SQL insira o código abaixo, na Mensagem de Erro digite "**Não há vagas nesta turma!**", em Local de Exibição escolha "**Em Linha com Campo e em Notificação**", em Item Associado: "**P10_COD_TURMA**" e clique "**Salvar**".

1=2=

O=Δ=

Após a Submissão

Validando

Validações

Valida quantidade de vagas

Processamento

Processos

Process Row of ALUNO_TURMA

reset page

Close Dialog

Após o Processamento

Ajax Callback

Layout

View de ...

Mensagens

Pesquisa...

Ajuda

Item Associado

Selecione o item no qual essa mensagem de erro de validação será exibida. Se você selecionar um item e o *Local de exibição da mensagem de erro* incluir "Em Linha com Campo", a mensagem de erro será exibida abaixo do label do item especificado. Se o *Local de exibição da mensagem de erro* não incluir "Em linha com Campo", esse atributo não afetará a exibição da mensagem de erro.

Validação

Filtro

Nome

Valida quantidade de vagas

Opções de Execução

Validação

Região Editável

- Selecionar -

Tipo

Linhas retornadas

Consulta SQL

SELECT 1
FROM turma
WHERE cod = :P10_COD_TURMA
AND qtd_vagas > F_QTD_PARTICIPANTE_TURMA(cod)

Executar Sempre

Sim

Não

Erro

Mensagem de Erro

Não há vagas nesta turma!

Local de Exibição

Em Linha com Campo e em Notificação

Item Associado

P10_COD_TURMA

Colocamos o item P10_COD_TURMA para que a mensagem seja exibida também ao lado deste item. Lembre-se que a validação é uma regra que você quer que aconteça e caso a regra não seja atendida a mensagem de erro é exibida.

Passo 3 - Acesse a aplicação no menu Turma e altera a quantidade vagas da turma **"Novo curso de APEX"** para **"5"**. Após isso, acesse a aba **"Matriculados por turma"**, clique no botão **"Nova matrícula"** e tente cadastrar um aluno na turma **"Novo curso de APEX"** e veja que um erro é exibido.

Minha Escola

aluno

Início

Aluno

Professor

Turma

Professores por turma

Matriculados por turma

Turma \

Matr

Cadastro de aluno em turma

Ocorreu 1 erro

Não há vagas nesta turma!

Aluno

Eusebio dos Santos Aguiar

Turma

Novo curso de APEX

Não há vagas nesta turma!

Nota final

10

Total de vagas

5

Alunos inscritos

5

Cancelar

Criar

Nova matrícula

	Nota
	8
	8
	10
	10
	0
	2
	4
	6
	8
1 - 9	

Pronto! Todas as perguntas referentes a essa regra de negócio foram respondidas.

Assim fechamos a regra de negócio RN 03!

4.4. Regra de Negócio RN 04

"**Todos os alunos têm nota em cada uma das turmas que participa**"

Para ver se esta regra já está ou não atendida, devemos checar se o campo NOTA aceita ou não valores nulos. Primeiramente, vamos acessar a tabela para ver se o campo é anulável. Depois temos que implementar uma validação na aplicação que impeça inserção de nulos no campo de nota.

Agora é com você!

Passo 1 – Acesse o **Browser de Objetos**, clique sobre a tabela ALUNO_TURMA e veja que o campo NOTA aceita valores nulos. Portanto, a regra de negócio ainda não está sendo atendida.

↑ Browser de Objetos

EsquemaAPEX_CURSO?

Tabelas

Q

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

ALUNO_TURMA

+ ▼

TabelaDadosÍndicesModeloConstraintsConcessõesEstatísticasPadrões da UITriggersDependênciasSQL

Adicionar Coluna

Modificar Coluna

Renomear Coluna

Eliminar Coluna

Renomear

Copiar

Eliminar

Truncar

Criar Tabela de Consulta

Nome da Coluna	Tipo de Dados	Anulável	Padrão	Chave Primária
COD	NUMBER	Não	-	1
COD_ALUNO	NUMBER	Não	-	-
COD_TURMA	NUMBER	Não	-	-
NOTA	NUMBER	Sim	-	-

[Fazer Download](#) | [Imprimir](#)

Passo 2 - Clique em "**Modificar Coluna**", selecione a Coluna "NOTA (NUMBER)" e altere o atributo Anulável para "NOT NULL" e prossiga até completar a alteração.

↑ Browser de Objetos

EsquemaAPEX_CURSO?

Tabelas

Q

ALUNO

ALUNO_TURMA

HTMLDB_PLAN_TABLE

PAGAMENTO

PROFESSOR

PROFESSOR_TURMA

TURMA

ALUNO_TURMA

+ ▼

Modificar Coluna

Use esta página para aumentar o tamanho dos tipos de dados da coluna de caracteres.

Esquema:

APEX_CURSO?

Tabela:

ALUNO_TURMA?

Coluna:

NOTA (NUMBER) ?

Tipo de Dados:

NUMBER ▼ ?

Tamanho:

?

Precisão:

?

Escala:

?

Anulável:

NOT NULL (requer um valor) ▼ ?

NOT NULL (requer um valor)

NULL (não requer um valor)

Cancelar

Próximo >

Passo 3 - Agora vamos alterar a aplicação. Para isso, acesse o formulário "**Cadastro de aluno em turma**", clique no item "**P10_NOTA**", defina a propriedade Valor Necessário para "**Sim**" e clique "**Salvar**".

1

2

Página 10: Cadastro de aluno em turma

> Pré-Renderização

> Regiões

> Dialog Header

> Breadcrumb

Atributos

> Content Body

> Cadastro de aluno em turma

Atributos

> Itens

P10_COD

P10_COD_ALUNO

P10_COD_TURMA

P10_NOTA

P10_QTD_VAGAS

P10_QTD_INSCRITOS

> Dialog Footer

> Botões

Layout

View de ...

Mensagens

Pesquisa...

Ajuda

Item de Página

Filtro

Valor Necessário

Se for definido como **Sim** e o item da página estiver visível, o Application Express executará automaticamente uma validação NÃO NULA quando a página for submetida.

Ao definir uma mensagem chamada **APEX.PAGE_ITEM_IS_REQUIRED** em Componentes Compartilhados> Mensagens de Texto, o texto do erro predefinido poderá ser substituído por uma mensagem de erro específica do aplicativo.

O local de exibição da mensagem será definido pela definição no nível do aplicativo **Local de exibição do erro padrão**.

Identificação

NomeP10_NOTA

TipoCampo de Número

Label

Nota final

Definições

Layout

Aparência

Validação

Valor NecessárioSimNão

Tamanho Máximo22caracteres

Passo 4 - Execute a aplicação, tente cadastrar um nova matrícula deixar a Nota com valor nulo. Ao se clicar em **Criar**, a mensagem de erro será exibida.

Minha Escola

aluno

Início

Aluno

Professor

Turma

Professores por turma

Matriculados por turma

Turma \

Matr

Cadastro de aluno em turma

Turma \ Matriculados por turma \ Cadastro de aluno em turma

Aluno

Joao Paulo Barros da S

Turma

Curso de APEX

Nota final

Preencha este campo.

Total de vagas

10

Alunos inscritos

2

Cancelar

Criar

Corrija os erros antes de salvar.

OK

Nova matrícula

Nota

8

8

10

10

0

2

4

6

8

1 - 9

Com isso fechamos a regra de negócio RN 4!

4.5. Regra de Negócio RN 05

Todos os valores monetários são em reais (R\$) e podem ser zero

Além de contextualizar os dados, garantindo que todos os valores mostrados pelo sistema são em reais, há também a restrição de que o valor pode ser zero.

Pergunta: há algo mais a fazer em relação a esta regra de negócio?

4.6. Resumo

Vimos neste capítulo:

- O que é SQL e seus principais comandos;
- O que é PL/SQL e suas principais estruturas;
- Algumas técnicas de debug; e
- Como implementar todas as regras de negócio.