

FIAP GRADUAÇÃO

Tecnologia em Análise e Desenvolvimento de Sistemas

Application Development For Databases

PROF. MILTON

Trabalhando com Tipos de Dados Compostos

Objetivos

Ao concluir esta lição, você será capaz de:

- Descrever conjuntos e registros PL/SQL
- Criar registros PL/SQL definidos pelo usuário
- Criar um registro PL/SQL com o atributo `%ROWTYPE`
- Criar arrays associativos
 - Tabela `INDEX BY`
 - Tabela de registros `INDEX BY`

Objetivos

Você já foi apresentado aos tipos de dados compostos. Nesta lição, você aprenderá mais sobre os tipos de dados compostos e seus usos.

Agenda

- Apresentando tipos de dados compostos
- Usando registros PL/SQL
 - Manipulando dados com registros PL/SQL
 - Vantagens do atributo `%ROWTYPE`
- Usando conjuntos PL/SQL
 - Examinando arrays associativos
 - Apresentando tabelas aninhadas
 - Apresentando `VARRAY`

Tipos de Dados Compostos

- Podem armazenar diversos valores (ao contrário dos tipos escalares)
- São de dois tipos:
 - Registros PL/SQL
 - Conjuntos PL/SQL
 - Array associativo (tabela INDEX BY)
 - Tabela aninhada
 - VARRAY

Tipos de Dados Compostos

Você aprendeu que variáveis de tipo de dados escalar só podem armazenar um valor, ao passo que uma variável de tipo de dados composto pode armazenar vários valores do tipo de dados escalar ou composto. Existem dois tipos de dados compostos:

- **Registros PL/SQL:** Os registros são usados para tratar dados relacionados, mas diferentes, como uma unidade lógica. Um registro PL/SQL pode ter variáveis de diferentes tipos. Por exemplo, você pode definir um registro para armazenar detalhes sobre o funcionário. Isso envolve armazenar o número do funcionário como NUMBER, nome e sobrenome como VARCHAR2 e assim por diante. Ao criar um registro para armazenar detalhes sobre o funcionário, você cria uma unidade de conjunto lógica. Isso facilita o acesso a dados e a manipulação.
- **Conjuntos PL/SQL:** Os conjuntos são usados para tratar os dados como uma só unidade. Existem três tipos de conjuntos:
 - Array associativo
 - Tabela aninhada
 - VARRAY

Por Que Usar Tipos de Dados Compostos?


Você terá todos os dados relacionados como uma só unidade. É possível acessar e modificar os dados facilmente. Será mais fácil gerenciar, relacionar e transportar dados se eles forem compostos. Uma analogia é ter uma única bolsa para todos os componentes do seu laptop em vez de uma bolsa separada para cada componente.

Registros ou Conjuntos PL/SQL?

- Use os registros PL/SQL quando desejar armazenar valores de diferentes tipos de dados, mas apenas uma ocorrência de cada vez.
- Use os conjuntos PL/SQL quando desejar armazenar valores do mesmo tipo de dados.

Conjunto PL/SQL:

Registro PL/SQL:

TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	---

1	SMITH
2	JONES
3	BENNETT
4	KRAMER

↑ PLS_INTEGER
↑ VARCHAR2

Registros ou Conjuntos PL/SQL?

Se tanto os registros PL/SQL quanto os conjuntos PL/SQL forem tipos compostos, como escolher qual deles será usado?

- Use os registros PL/SQL quando desejar armazenar valores de diferentes tipos de dados que são relacionados logicamente. Por exemplo, você pode criar um registro PL/SQL para armazenar detalhes sobre um funcionário e indicar que todos os valores armazenados estejam relacionados, pois eles fornecem informações sobre um determinado funcionário.
- Use os conjuntos PL/SQL quando desejar armazenar valores do mesmo tipo de dados. Observe que esse tipo de dados também pode ser do tipo composto, como registros. Você pode definir um conjunto para armazenar os nomes de todos os funcionários. Você pode ter n nomes armazenados no conjunto, porém o nome 1 não tem relação com o nome 2. A única relação entre esses nomes é que eles são nomes de funcionários. Esses conjuntos são semelhantes a arrays em linguagens de programação como C, C++ e Java.

Agenda

- Examinando tipos de dados compostos
- Usando registros PL/SQL
 - Manipulando dados com registros PL/SQL
 - Vantagens do atributo `%ROWTYPE`
- Usando conjuntos PL/SQL
 - Examinando arrays associativos
 - Apresentando tabelas aninhadas
 - Apresentando `VARRAY`

Registros PL/SQL

- Devem conter um ou mais componentes (chamados *campos*) de qualquer tipo de dados escalar, `RECORD`, ou tabela `INDEX BY`
- São semelhantes às estruturas na maioria das linguagens de terceira geração (incluindo C e C++)
- São definidos pelos usuários e podem ser um subconjunto de uma linha de uma tabela
- Tratam um conjunto de campos como uma unidade lógica
- São convenientes para extrair, com o comando `fetch`, uma linha de dados de uma tabela para processamento

Registros PL/SQL

Um registro é um grupo de itens de dados relacionados armazenados em campos, cada um com seu próprio nome e tipo de dados.

- Cada registro definido pode ter quantos campos forem necessários.
- É possível designar aos registros valores iniciais, e os registros podem ser definidos como `NOT NULL`.
- Os campos sem valores iniciais serão inicializados como `NULL`.
- A palavra-chave `DEFAULT`, bem como `:=` pode ser usada nos campos de inicialização.
- Você pode definir tipos `RECORD` e declarar registros definidos pelo usuário na parte declarativa de qualquer bloco, subprograma ou pacote.
- Você pode declarar e fazer referência a registros aninhados. Um registro pode ser o componente de outro registro.

Criando um Registro PL/SQL

Sintaxe:

1

```
TYPE type_name IS RECORD  
    (field_declaration [, field_declaration]...);
```

2

```
identifier    type_name;
```

field_declaration:

```
field_name {field_type | variable%TYPE  
            | table.column%TYPE | table%ROWTYPE}  
            [[NOT NULL] {:= | DEFAULT} expr]
```

10

Criando um Registro PL/SQL

Os registros PL/SQL são tipos compostos definidos pelo usuário. Para usá-los, execute estas etapas:

1. Defina o registro na seção declarativa de um bloco PL/SQL. A sintaxe para definir o registro é mostrada no slide.
2. Declare (e, opcionalmente, inicialize) os componentes internos desse tipo de registro.

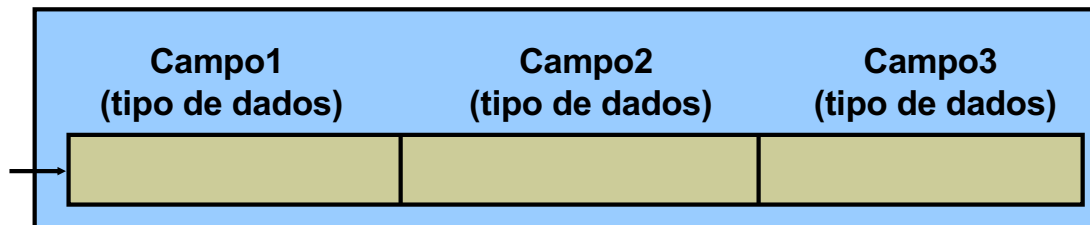
Na sintaxe:

<i>type_name</i>	É o nome do tipo RECORD (Esse identificador é usado para declarar registros.)
<i>field_name</i>	É o nome de um campo dentro do registro
<i>field_type</i>	É o tipo de dados do campo (representa qualquer tipo de dados PL/SQL, exceto REF CURSOR. É possível usar os atributos %TYPE e %ROWTYPE).
<i>expr</i>	É o <i>field_type</i> ou um valor inicial

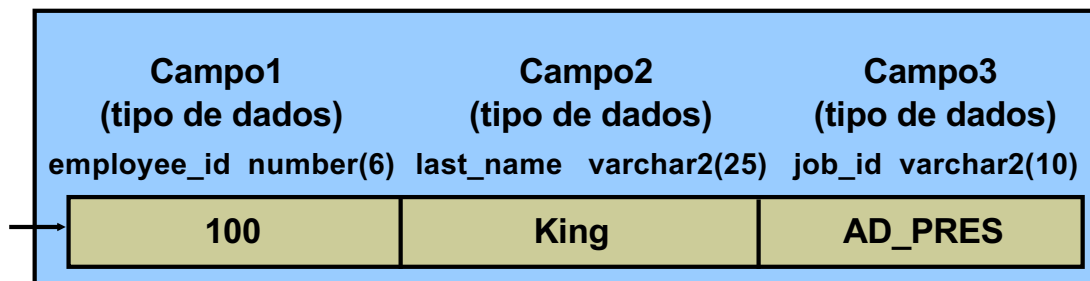
A constraint NOT NULL impede a atribuição de nulos aos campos especificados. Certifique-se de inicializar os campos NOT NULL.

Estrutura de Registros PL/SQL

Declarações de campo:



Exemplo:



Estrutura de Registros PL/SQL

Os campos de um registro são acessados pelo nome do registro. Para fazer referência ou inicializar um campo individual, use a notação de ponto:

```
record_name.field_name
```

Por exemplo, faça referência ao campo `job_id` do registro `emp_record` desta forma:

```
emp_record.job_id
```

Em seguida, você poderá designar um valor ao campo do registro:

```
emp_record.job_id := 'ST_CLERK';
```

Em um bloco ou subprograma, os registros definidos pelo usuário são instanciados quando você informa o bloco ou o subprograma. Eles deixam de existir quando você sai do bloco ou do subprograma.

Atributo %ROWTYPE

- Declare uma variável de acordo com um conjunto de colunas de uma view ou tabela de banco de dados.
- Use como prefixo de %ROWTYPE a view ou a tabela de banco de dados.
- Os campos do registro adotam os nomes e tipos de dados das colunas da view ou da tabela.

Sintaxe:

```
DECLARE
    identifier reference%ROWTYPE;
```

12

Atributo %ROWTYPE

Você aprendeu que %TYPE é usado para declarar uma variável do tipo de coluna. A variável tem o mesmo tipo de dados e tamanho de uma coluna da tabela. A vantagem de %TYPE é que não será necessário alterar a variável caso a coluna seja alterada. Além disso, se a variável for um número e for usada em qualquer cálculo, você não precisará se preocupar com a sua precisão.

O atributo %ROWTYPE é usado para declarar um registro que possa armazenar uma linha inteira de uma view ou tabela. Os campos do registro adotam os nomes e tipos de dados das colunas da view ou da tabela. O registro também pode armazenar uma linha inteira de dados extraídos de um cursor ou uma variável de cursor.

O slide mostra a sintaxe para se declarar um registro. Na sintaxe:

<i>identifier</i>	É o nome escolhido para o registro como um todo
<i>reference</i>	É o nome da tabela, view, cursor ou variável de cursor na qual o registro deve ser baseado (a tabela ou view deve existir para que essa referência seja válida).

Neste exemplo, um registro é declarado usando %ROWTYPE como um especificador de tipo de dados:

```
DECLARE
    emp_record employees%ROWTYPE;
    ...
```

Criando um Registro PL/SQL: Exemplo

```
DECLARE
  TYPE t_rec IS RECORD
    (v_sal number(8),
     v_minsal number(8) default 1000,
     v_hire_date employees.hire_date%type,
     v_rec1 employees%rowtype);
  v_myrec t_rec;
BEGIN
  v_myrec.v_sal := v_myrec.v_minsal + 500;
  v_myrec.v_hire_date := sysdate;
  SELECT * INTO v_myrec.v_rec1
    FROM employees WHERE employee_id = 100;
  DBMS_OUTPUT.PUT_LINE(v_myrec.v_rec1.last_name || ' ' ||
    to_char(v_myrec.v_hire_date) || ' ' || to_char(v_myrec.v_sal));
END;
```

```
anonymous block completed
King 16-FEB-09 1500
```

14

Criando um Registro PL/SQL: Exemplo

As declarações de campo usadas para definir um registro são como declarações de variáveis. Cada campo tem um nome exclusivo e um tipo de dados específico. Não existem tipos de dados predefinidos para registros PL/SQL, como existem para as variáveis escalares. Portanto, você deve criar primeiro o tipo de registro e, depois, declarar um identificador usando aquele tipo.

No exemplo do slide, um registro PL/SQL é criado usando o processo de duas etapas necessário:

1. Um tipo de registro (t_rec) é definido
2. Um registro (v_myrec) do tipo t_rec é declarado

Observação

- O registro contém quatro campos: v_sal, v_minsal, v_hire_date e v_rec1.
- v_rec1 é definido usando o atributo %ROWTYPE, que é semelhante ao atributo %TYPE. Com %TYPE, um campo herda o tipo de dados de uma coluna especificada. Com %ROWTYPE, um campo herda os nomes de colunas e os tipos de dados de todas as colunas da tabela referenciada.
- Os campos do registro PL/SQL são referenciados por meio da notação <record>.<field> ou da notação <record>.<field>.<column> no caso dos campos que são definidos com o atributo %ROWTYPE.
- É possível adicionar a constraint NOT NULL a qualquer declaração de campo para impedir a designação de nulos a esse campo. Lembre-se de que esses campos que são declarados como NOT NULL devem ser inicializados.

Vantagens do Uso do Atributo %ROWTYPE

- O número e os tipos de dados das colunas subjacentes do banco de dados não precisam ser conhecidos e, de fato, devem mudar durante o runtime.
- O atributo %ROWTYPE é útil quando você deseja recuperar uma linha com:
 - A instrução `SELECT *`
 - Instruções `INSERT` e `UPDATE` em nível de linha

Vantagens do Uso de %ROWTYPE

As vantagens de usar o atributo %ROWTYPE estão listadas no slide. Use o atributo %ROWTYPE quando não tiver certeza da estrutura da tabela subjacente do banco de dados.

A principal vantagem de usar %ROWTYPE é que ele simplifica a manutenção. A utilização de %ROWTYPE assegura que os tipos de dados das variáveis declaradas com esse atributo serão dinamicamente alterados caso a tabela subjacente seja alterada. Se uma instrução DDL alterar as colunas de uma tabela, a unidade do programa PL/SQL será invalidada. Quando o programa for recompilado, ele refletirá automaticamente o novo formato de tabela.

Os atributos %ROWTYPE serão particularmente úteis se você quiser recuperar uma linha inteira de uma tabela. Na ausência desse atributo, você teria que declarar uma variável para cada coluna recuperada pela instrução `SELECT`.

Outro Exemplo do Atributo %ROWTYPE

```

DECLARE
    v_employee_number number:= 124;
    v_emp_rec    employees%ROWTYPE;
BEGIN
    SELECT * INTO v_emp_rec FROM employees
    WHERE employee_id = v_employee_number;
    INSERT INTO retired_emps(empno, ename, job, mgr,
                           hiredate, leavedate, sal, comm, deptno)
    VALUES (v_emp_rec.employee_id, v_emp_rec.last_name,
            v_emp_rec.job_id, v_emp_rec.manager_id,
            v_emp_rec.hire_date, SYSDATE,
            v_emp_rec.salary, v_emp_rec.commission_pct,
            v_emp_rec.department_id);
END;
/

```

SELECT * FROM retired_emps;

Results: Script Output Explain Autotrace DBMS Output OWA Output

	EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124	Mourgos	ST_MAN	100	16-NOV-99	16-JUN-09	5800	(null)	50

16

Outro Exemplo do Atributo %ROWTYPE

Outro exemplo do atributo %ROWTYPE é mostrado no slide. Se um funcionário estiver se aposentando, as informações sobre ele serão adicionadas à tabela que armazena informações sobre funcionários aposentados. O usuário fornece o número do funcionário. O registro do funcionário especificado pelo usuário é obtido da tabela employees e armazenado na variável emp_rec, que foi declarada com o atributo %ROWTYPE.

A instrução CREATE, que cria a tabela retired_emps é:

```

CREATE TABLE retired_emps
(EMPNO      NUMBER(4), ENAME      VARCHAR2(10),
 JOB        VARCHAR2(9), MGR       NUMBER(4),
 HIREDATE    DATE, LEAVEDATE    DATE,
 SAL         NUMBER(7,2), COMM     NUMBER(7,2),
 DEPTNO      NUMBER(2))

```

Observação

- O registro inserido na tabela retired_emps é mostrado no slide.
- Para ver a saída mostrada no slide, coloque o cursor na instrução SELECT, na parte inferior do código de exemplo no SQL Developer e pressione F9.
- O código de exemplo completo está em code_6_14_n-s.sql.

Inserindo um Registro Usando %ROWTYPE

```

...
DECLARE
    v_employee_number number:= 124;
    v_emp_rec retired_emps%ROWTYPE;
BEGIN
    SELECT employee_id, last_name, job_id, manager_id,
    hire_date, hire_date, salary, commission_pct,
    department_id INTO v_emp_rec FROM employees
    WHERE employee_id = v_employee_number;
    INSERT INTO retired_emps VALUES v_emp_rec;
END;
/
SELECT * FROM retired_emps;

```

Results

Script Output

Explain

Autotrace

DBMS Output

OWA Output

Results:

EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124 Mourgos	ST_MAN	100	16-NOV-99	16-NOV-99	5800	(null)	50

17

Inserindo um Registro Usando %ROWTYPE

Compare a instrução INSERT do slide anterior com a instrução INSERT deste slide. O registro emp_rec é do tipo retired_emps. O número de campos no registro deve ser igual ao número de nomes de campos contido na cláusula INTO. É possível usar esse registro para inserir valores em uma tabela. Isso torna o código mais legível.

Observe a instrução SELECT no slide. Selecione hire_date duas vezes e insira o valor referente a hire_date no campo leavedate de retired_emps. Nenhum funcionário se aposenta na data de admissão. O registro inserido é mostrado no slide. (No próximo slide, você verá como atualizar isso.)

Observação: Para ver a saída mostrada no slide, coloque o cursor na instrução SELECT, na parte inferior do código de exemplo no SQL Developer, e pressione F9.

Atualizando uma Linha da Tabela Usando um Registro

```

SET VERIFY OFF
DECLARE
    v_employee_number number:= 124;
    v_emp_rec retired_emps%ROWTYPE;
BEGIN
    SELECT * INTO v_emp_rec FROM retired_emps;
    v_emp_rec.leavedate:=CURRENT_DATE;
    UPDATE retired_emps SET ROW = v_emp_rec WHERE
        empno=v_employee_number;
END;
/
SELECT * FROM retired_emps;

```

Results

Script Output

Explain

Autotrace

DBMS Output

OWA Output

Results:

EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124 Mourgos	ST_MAN	100	16-NOV-99	16-NOV-99	5800	(null)	50

18

Atualizando a Linha de uma Tabela Usando um Registro

Você aprendeu a inserir uma linha usando um registro. O slide mostra como atualizar uma linha usando um registro.

- A palavra-chave `ROW` é usada para representar a linha inteira.
- O código mostrado no slide atualiza o campo `leavedate` do funcionário.
- O registro é atualizado como mostrado no slide.

Observação: Para ver a saída mostrada no slide, coloque o cursor na instrução `SELECT`, na parte inferior do código de exemplo no SQL Developer, e pressione F9.

Agenda

- Examinando tipos de dados compostos
- Usando registros PL/SQL
 - Manipulando dados com registros PL/SQL
 - Vantagens do atributo `%ROWTYPE`
- Usando conjuntos PL/SQL
 - Examinando arrays associativos
 - Apresentando tabelas aninhadas
 - Apresentando `VARRAY`

Agenda

Como mencionado anteriormente, os conjuntos PL/SQL são usados quando você deseja armazenar valores do mesmo tipo de dados. Esse tipo de dados também pode ser do tipo composto (como registros).

Portanto, os conjuntos são usados para tratar os dados como uma unidade única. Existem três tipos de conjuntos:

- Array associativo
- Tabela aninhada
- `VARRAY`

Observação: Desses três conjuntos, o array associativo é o foco desta lição. A tabela aninhada e `VARRAY` são apresentados apenas com a finalidade de comparação. Esses dois conjuntos são abordados em detalhes no curso *Oracle Database 10g: Advanced PL/SQL* ou *Oracle Database 11g: Advanced PL/SQL* (conforme for adequado para a versão que você estiver usando).

Arrays Associativos (Tabelas INDEX BY) FIAP

Um array associativo é um conjunto PL/SQL com duas colunas:

- Chave primária de tipo de dados inteiro ou string
- Coluna de tipo de dados escalar ou de registro

Chave	Valores
1	JONES
2	HARDEY
3	MADURO
4	KRAMER

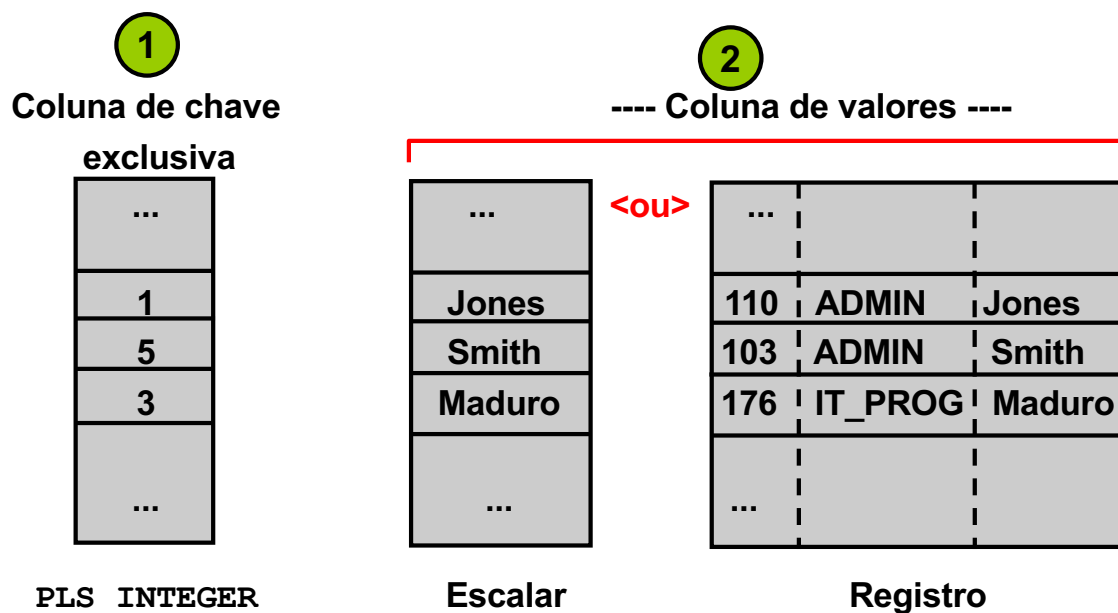
Arrays Associativos (Tabelas INDEX BY)

Um array associativo é um tipo de conjunto PL/SQL. Ele é um tipo de dados composto e é definido pelo usuário. Os arrays associativos são conjuntos de pares de valor/chave. Eles podem armazenar dados usando um valor de chave primária como o índice, em que os valores da chave não são necessariamente sequenciais. Os arrays associativos também são conhecidos como tabelas *INDEX BY*.

Eles possuem apenas duas colunas, sendo que nenhuma delas pode ser nomeada:

- A primeira coluna, de tipo inteiro ou string, funciona como a chave primária.
- A segunda coluna, de tipo de dados escalar ou de registro, armazena valores.

Estrutura de Array Associativo



Estrutura de Array Associativo

Como mencionado anteriormente, os arrays associativos possuem duas colunas. A segunda coluna armazena um valor por linha ou vários valores.

Coluna de Chave Exclusiva: O tipo de dados da coluna de chave pode ser:

- Numérico, seja `BINARY_INTEGER` ou `PLS_INTEGER`. Esses dois tipos de dados numéricos requerem menos armazenamento do que `NUMBER`, e as operações aritméticas nesses tipos de dados são mais rápidas do que a aritmética `NUMBER`.
- `VARCHAR2` ou um dos seus subtipos

Coluna "Value": A coluna de valor pode ser de um tipo de dados escalar ou de registro. Uma coluna com tipo de dados escalar só pode armazenar um valor por linha, ao passo que uma coluna com tipo de dados de registro pode armazenar vários valores por linha.

Outras Características

- Um array associativo não é preenchido no momento da declaração. Ele não contém chaves ou valores e não pode ser inicializado na sua declaração.
- Uma instrução executável explícita é necessária para preencher o array associativo.
- Assim como o tamanho de uma tabela do banco de dados, o tamanho de um array associativo não tem restrição, ou seja, o número de linhas pode aumentar dinamicamente, de forma que o array associativo cresce à medida que novas linhas são adicionadas. Observe que as chaves não precisam ser sequenciais e podem ser positivas e negativas.

Etapas para a Criação de um Array Associativo

Sintaxe:

```

1  TYPE type_name IS TABLE OF
    {column_type | variable%TYPE
    | table.column%TYPE} [NOT NULL]
    | table%ROWTYPE
    | INDEX BY PLS_INTEGER | BINARY_INTEGER
    | VARCHAR2(<size>);
2  identifier type_name;
  
```

Exemplo:

```

...
TYPE ename_table_type IS TABLE OF
    employees.last_name%TYPE
    INDEX BY PLS_INTEGER;
...
ename_table ename_table_type;
  
```

22

Etapas para a Criação de um Array Associativo

Há duas etapas na criação de um array associativo:

1. Declare um tipo de dados TABLE usando a opção INDEX BY.
2. Declare uma variável com esse tipo de dados.

Sintaxe

type_name É o nome do tipo de dados TABLE (Este nome é usado na declaração subsequente do identificador do array.)

column_type É qualquer qualquer tipo de dados escalar ou composto como VARCHAR2, DATE, NUMBER, ou %TYPE (é possível usar o atributo %TYPE para fornecer o tipo de dados da coluna.)

identifier É o nome do identificador que representa um array associativo inteiro

Observação: A constraint NOT NULL impede que nulos sejam designados ao array associativo.

Exemplo

No exemplo, um array associativo com o nome de variável *ename_table* é declarado para armazenar os sobrenomes dos funcionários.

Criando e Acessando Arrays Associativos

```

...
DECLARE
  TYPE ename_table_type IS TABLE OF
    employees.last_name%TYPE
    INDEX BY PLS_INTEGER;
  TYPE hiredate_table_type IS TABLE OF DATE
    INDEX BY PLS_INTEGER;
  ename_table          ename_table_type;
  hiredate_table       hiredate_table_type;
BEGIN
  ename_table(1)       := 'CAMERON';
  hiredate_table(8)    := SYSDATE + 7;
  IF ename_table.EXISTS(1) THEN
    INSERT INTO ...
    ...
END;
/
...

```

```

anonymous block completed
ENAME                HIREDT
-----
CAMERON              23-JUN-09

1 rows selected

```

23

Criando e Acessando Arrays Associativos

O exemplo do slide cria dois arrays associativos, com os identificadores `ename_table` e `hiredate_table`.

A chave de cada array associativo é usada para acessar um elemento do array, usando a seguinte sintaxe:

```
identifier(index)
```

Em ambos os arrays, o valor `index` pertence ao tipo `PLS_INTEGER`.

- Para fazer referência à primeira linha do array associativo `ename_table`, especifique:
`ename_table(1)`
- Para fazer referência à oitava linha do array associativo `hiredate_table`, especifique:
`hiredate_table(8)`

Observação

- A faixa de magnitude de um `PLS_INTEGER` é de `-2.147.483.647` a `2.147.483.647`, portanto, o valor da chave primária pode ser negativo. A indexação não precisa iniciar em 1.
- O método `exists(i)` retorna `TRUE` se uma linha com o índice `i` for retornada. Use o método `exists` para impedir a ocorrência de erro em relação a um elemento não existente na tabela.
- O código de exemplo completo está em `code_6_21_s.sql`.

Usando Métodos da Tabela INDEX BY

Os seguintes métodos facilitam o uso de arrays associativos:

- EXISTS
- COUNT
- FIRST
- LAST
- PRIOR
- NEXT
- DELETE

Usando Métodos da Tabela INDEX BY

Um método da tabela INDEX BY é um procedure ou uma função incorporados que executam operações em um array associativo e são chamados por meio da notação de ponto.

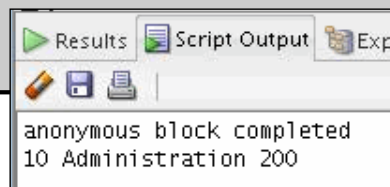
Sintaxe: `table_name.method_name[(parameters)]`

Método	Descrição
EXISTS (<i>n</i>)	Retorna TRUE caso o elemento <i>enésimo</i> exista em um array associativo
COUNT	Retorna o número de elementos que um array associativo contenha atualmente
FIRST	<ul style="list-style-type: none">• Retorna o primeiro (o menor) número de índice em um array associativo• Retorna NULL se o array associativo estiver vazio
LAST	<ul style="list-style-type: none">• Retorna o último (o maior) número de índice em um array associativo• Retorna NULL se o array associativo estiver vazio
PRIOR (<i>n</i>)	Retorna o número de índice que precede o índice <i>n</i> em um array associativo
NEXT (<i>n</i>)	Retorna o número de índice que sucede o índice <i>n</i> em um array associativo
DELETE	<ul style="list-style-type: none">• DELETE remove todos os elementos de um array associativo.• DELETE (<i>n</i>) remove o <i>enésimo</i> elemento de um array associativo.• DELETE (<i>m</i>, <i>n</i>) remove todos os elementos na faixa <i>m</i> ... <i>n</i> de um array associativo.

Opção da Tabela de Registros INDEX BY

Defina um array associativo para armazenar uma linha inteira de uma tabela.

```
DECLARE
  TYPE dept_table_type IS TABLE OF
    departments%ROWTYPE INDEX PLS_INTEGER;
  dept_table dept_table_type;
  -- Each element of dept_table is a record
Begin
  SELECT * INTO dept_table(1) FROM departments
    WHERE department_id = 10;
  DBMS_OUTPUT.PUT_LINE(dept_table(1).department_id || ' ' ||
    dept_table(1).department_name || ' ' ||
    dept_table(1).manager_id);
END;
/
```



25

Opção da Tabela de Registros INDEX BY

Como abordado anteriormente, um array associativo que é declarado como uma tabela de tipo de dados escalar pode armazenar os detalhes apenas de uma coluna de uma tabela do banco de dados. Entretanto, com frequência, é necessário armazenar todas as colunas recuperadas por uma consulta. A opção da tabela de registros INDEX BY permite a definição de um array para armazenar as informações sobre todos os campos de uma tabela do banco de dados.

Criando e Fazendo Referência a uma Tabela de Registros

Como mostrado no exemplo do array associativo no slide, você pode:

- Usar o atributo %ROWTYPE para declarar um registro que represente uma linha de uma tabela do banco de dados.
- Fazer referência a campos contidos no array dept_table, porque cada elemento do array é um registro

As diferenças entre o atributo %ROWTYPE e o registro PL/SQL de tipo de dados composto são as seguintes:

- Os tipos de registro PL/SQL podem ser definidos pelo usuário, ao passo que %ROWTYPE define implicitamente o registro.
- Os registros PL/SQL permitem que você especifique campos e seus tipos de dados ao declará-los. Se usar %ROWTYPE, você não poderá especificar os campos. O atributo %ROWTYPE representa a linha de uma tabela com todos os campos baseados na definição dessa tabela.
- Os registros definidos pelo usuário são estáticos, mas os registros %ROWTYPE são dinâmicos — eles são baseados em uma estrutura de tabela. Se a estrutura da tabela for alterada, a estrutura do registro também assimilará a alteração.

Opção da Tabela de Registros INDEX BY: Exemplo 2

```
DECLARE
  TYPE emp_table_type IS TABLE OF
    employees%ROWTYPE INDEX BY PLS_INTEGER;
  my_emp_table emp_table_type;
  max_count    NUMBER(3) := 104;
BEGIN
  FOR i IN 100..max_count
  LOOP
    SELECT * INTO my_emp_table(i) FROM employees
    WHERE employee_id = i;
  END LOOP;
  FOR i IN my_emp_table.FIRST..my_emp_table.LAST
  LOOP
    DBMS_OUTPUT.PUT_LINE(my_emp_table(i).last_name);
  END LOOP;
END;
/
```

26

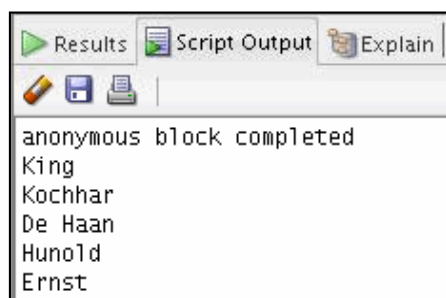
Tabela de Registros INDEX BY: Exemplo 2

O exemplo do slide declara um array associativo, usando a opção da tabela de registros INDEX BY, para armazenar temporariamente os detalhes dos funcionários cujos IDs estão entre 100 e 104. O nome da variável para o array é `emp_table_type`.

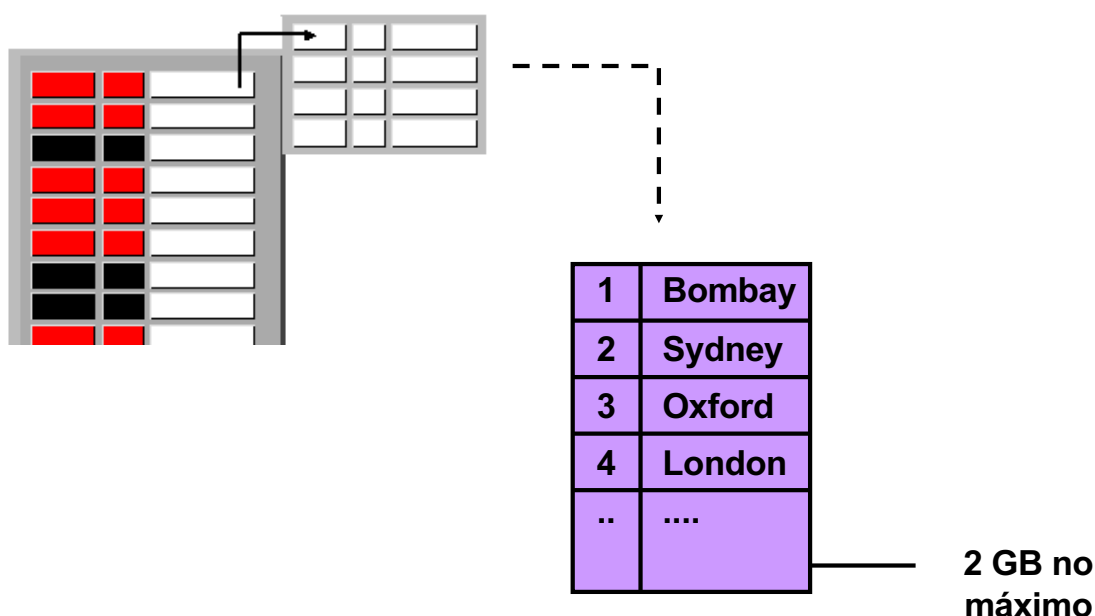
Usando um loop, as informações sobre os funcionários da tabela EMPLOYEES são recuperadas e armazenadas no array. Outro loop é usado para imprimir os sobrenomes do array. Observe o uso dos métodos `first` e `last` no exemplo.

Observação: O slide demonstra um modo de trabalhar com um array associativo que usa o método da tabela de registros INDEX BY. Entretanto, você pode fazer o mesmo de forma mais eficiente usando cursores. Os cursores são explicados na lição “Usando Cursores Explícitos”.

Os resultados do código de exemplo são os seguintes:



Tabelas Aninhadas



27

Tabelas Aninhadas

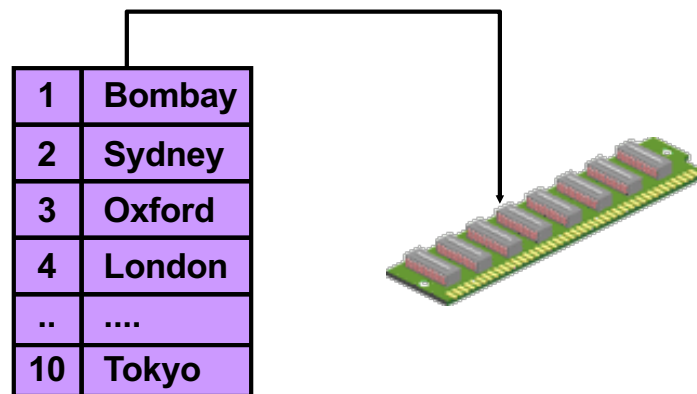
A funcionalidade das tabelas aninhadas é semelhante à dos arrays associativos; no entanto, há diferenças na implementação da tabela aninhada.

- Uma tabela aninhada é um tipo de dados válido em uma tabela em nível de esquema, mas um array associativo não é. Portanto, ao contrário dos arrays associativos, as tabelas aninhadas podem ser armazenadas no banco de dados.
- O tamanho de uma tabela aninhada pode aumentar dinamicamente, embora o tamanho máximo seja de 2 GB.
- A “chave” não pode ter um valor negativo (ao contrário do array associativo). Embora seja feita uma referência à primeira coluna como chave, não há chave em uma tabela aninhada. Há uma coluna com números.
- Os elementos podem ser deletados de qualquer lugar de uma tabela aninhada deixando a tabela desordenada com “chaves” não sequenciais. As linhas de uma tabela aninhada não estão em uma ordem específica.
- Se você recuperar valores de uma tabela aninhada, as linhas receberão subscripts consecutivos, começando em 1.

Sintaxe

```
TYPE type_name IS TABLE OF
    {column_type | variable%TYPE
    | table.column%TYPE} [NOT NULL]
    | table.%ROWTYPE
```

VARRAY



VARRAY

Um array de tamanho variável (VARRAY) é semelhante a um array associativo, a diferença é que um VARRAY tem restrição de tamanho.

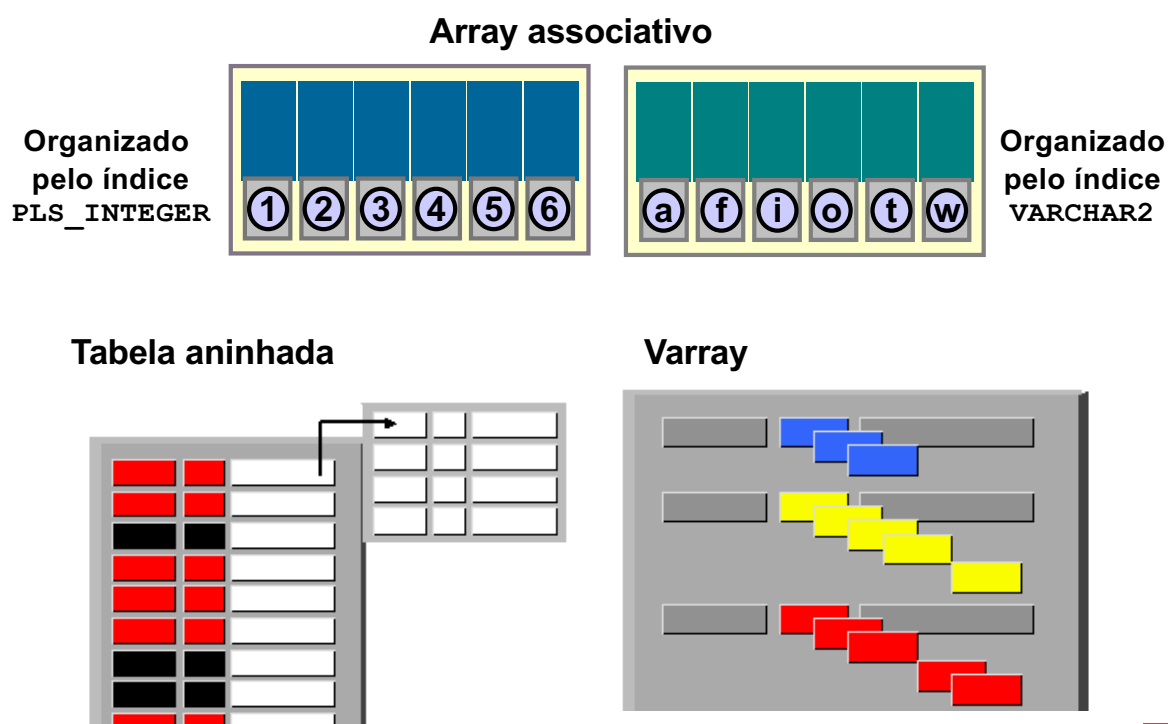
- Um VARRAY é válido em uma tabela em nível de esquema.
- Itens do tipo VARRAY são chamados de VARRAYs.
- Os VARRAYs têm um limite superior fixo. Você deverá especificar o limite superior ao declará-los. Essa característica é semelhante à dos arrays na linguagem C. O tamanho máximo de um VARRAY é de 2 GB, como nas tabelas aninhadas.
- A diferença entre a tabela aninhada e o VARRAY é o modo de armazenamento físico. Os elementos de um VARRAY são armazenados em linha com os dados da tabela, exceto se o tamanho do VARRAY for superior a 4 KB. Compare-os com as tabelas aninhadas, que são sempre armazenadas fora da linha.
- Com o código SQL, é possível criar um tipo VARRAY no banco de dados.

Exemplo:

```
TYPE location_type IS VARRAY(3) OF locations.city%TYPE;  
offices location_type;
```

O tamanho desse VARRAY é restrito a 3. Você pode inicializar um VARRAY usando construtores. Se você tentar inicializar o VARRAY com mais de três elementos, será exibida a mensagem de erro “Subscript outside of limit”.

Resumo dos Tipos de Conjunto



30

Resumo dos Tipos de Conjunto

Arrays Associativos

Os arrays associativos são conjuntos de pares de valor/chave, em que cada chave é exclusiva e usada para localizar um valor correspondente no array. A chave pode basear-se em um número inteiro ou em um caractere. O array pode ter um valor de tipo de dados escalar (único valor) ou de tipo de dados de registro (vários valores).

Como o objetivo dos arrays associativos é armazenar dados temporários, você não poderá usá-los com instruções SQL como `INSERT` e `SELECT INTO`.

Tabelas Aninhadas

Uma tabela aninhada armazena um conjunto de valores. Em outras palavras, é uma tabela dentro de outra. Não há limite para as tabelas aninhadas, ou seja, o tamanho delas pode aumentar dinamicamente. As tabelas aninhadas estão disponíveis no código PL/SQL e no banco de dados. No código PL/SQL, as tabelas aninhadas são como arrays unidimensionais cujo tamanho pode aumentar dinamicamente.

Varrays

Os arrays de tamanho variável, ou varrays, também são conjuntos de elementos homogêneos que armazenam um número fixo de elementos (embora você possa alterar o número de elementos no runtime). Eles usam números sequenciais como subscripts. Você pode definir tipos SQL equivalentes, permitindo dessa forma o armazenamento de varrays em tabelas do banco de dados.

Identifique situações em que você possa usar o atributo `%ROWTYPE`.

- a. Quando você não tiver certeza da estrutura da tabela subjacente do banco de dados
- b. Quando você desejar recuperar uma linha inteira de uma tabela
- c. Quando você desejar declarar uma variável de acordo com outra declarada anteriormente ou com uma coluna do banco de dados

Resposta: a, b

Vantagens do Uso do Atributo `%ROWTYPE`

Use o atributo `%ROWTYPE` quando você não tiver certeza da estrutura da tabela subjacente do banco de dados.

A principal vantagem de usar `%ROWTYPE` é que ele simplifica a manutenção. A utilização de `%ROWTYPE` assegura que os tipos de dados das variáveis declaradas com esse atributo serão dinamicamente alterados caso a tabela subjacente seja alterada. Se uma instrução DDL alterar as colunas de uma tabela, a unidade do programa PL/SQL será invalidada. Quando o programa for recompilado, ele refletirá automaticamente o novo formato de tabela.

O atributo `%ROWTYPE` é particularmente útil se você quiser recuperar uma linha inteira de uma tabela. Na ausência desse atributo, você teria que declarar uma variável para cada coluna recuperada pela instrução `SELECT`.

Nesta lição, você aprendeu a:

- Definir e fazer referência a variáveis PL/SQL de tipos de dados compostos
 - Registro PL/SQL
 - Array associativo
 - Tabela `INDEX BY`
 - Tabela de registros `INDEX BY`
- Definir um registro PL/SQL usando o atributo `%ROWTYPE`
- Comparar os três tipos de conjunto PL/SQL:
 - Array associativo
 - Tabela aninhada
 - `VARRAY`

Sumário

Um registro PL/SQL é um conjunto de campos individuais que representam uma linha de uma tabela. Usando registros, você pode agrupar os dados em uma estrutura e, posteriormente, manipular essa estrutura como uma entidade ou unidade lógica. Isso ajuda a diminuir a codificação e facilita a compreensão e a manutenção do código.

Assim como os registros PL/SQL, um conjunto PL/SQL é outro tipo de dados composto. Os conjuntos PL/SQL incluem:

- Arrays associativos (também conhecidos como tabelas `INDEX BY`). Eles são objetos do tipo `TABLE` e são semelhantes às tabelas do banco de dados, mas com uma pequena diferença. As chamadas tabelas `INDEX BY` usam uma chave primária para prover acesso às linhas de modo semelhante a arrays. O tamanho de um array associativo não tem restrição.
- Tabelas aninhadas. A chave de tabelas aninhadas não pode ter um valor negativo, ao contrário das tabelas `INDEX BY`. A chave também deve estar em uma sequência.
- Arrays de tamanho variável (`VARRAY`). Um `VARRAY` é semelhante a um array associativo, a diferença é que um `VARRAY` tem restrição de tamanho.

Exercício 6: Visão Geral

Este exercício aborda os seguintes tópicos:

- Declarando arrays associativos
- Processando dados usando arrays associativos
- Declarando um registro PL/SQL
- Processando dados usando um registro PL/SQL

Exercício 6: Visão Geral

Neste exercício, você definirá, criará e usará arrays associativos e registros PL/SQL.