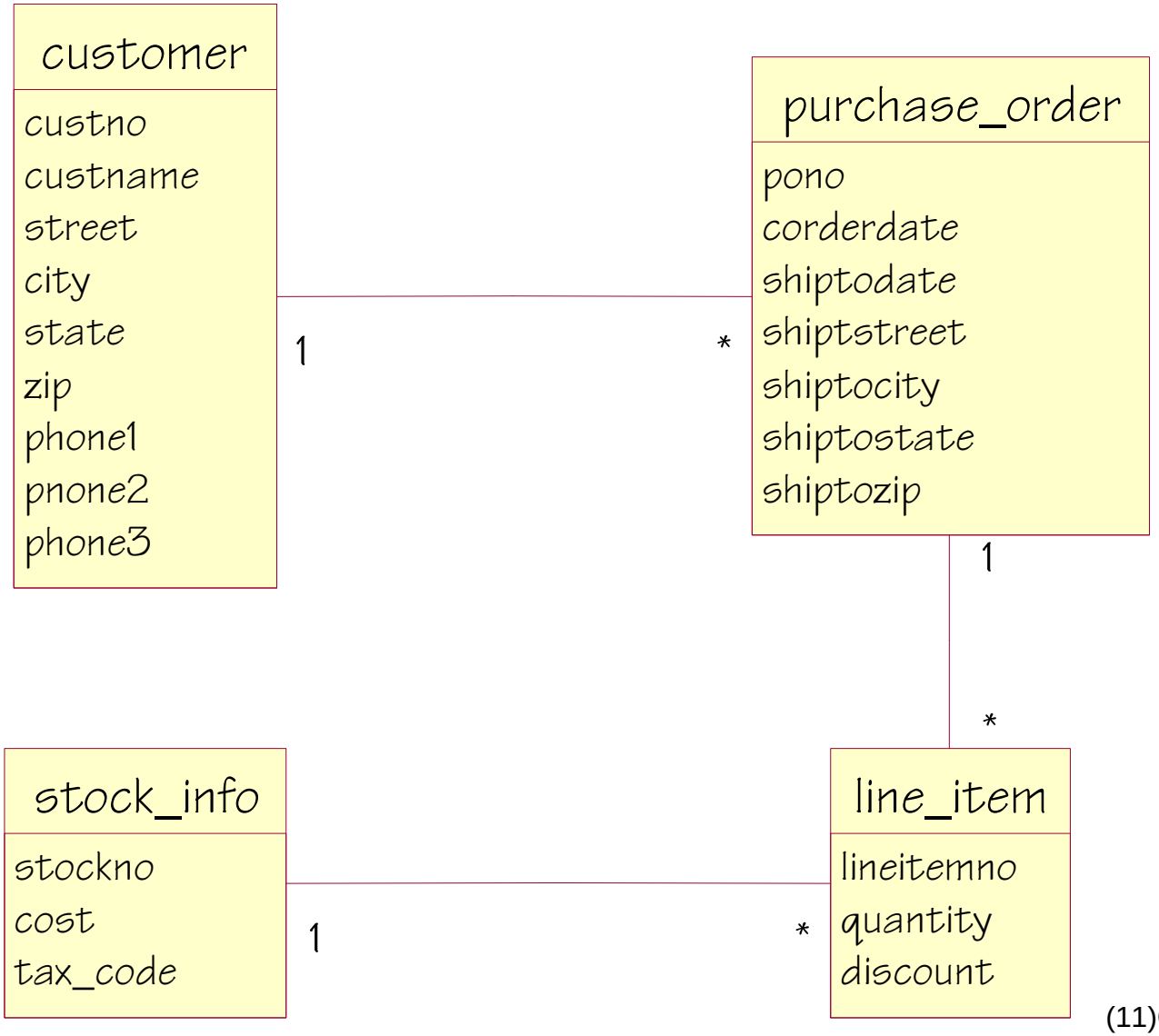
Exemplo de Oracle8 - modelo conceitual



Definições preliminares

- **Inicialmente são definidas alguns tipos (forward definition) para uso futuro**

```
CREATE TYPE line_item_t ;
CREATE TYPE purchase_order_t ;
CREATE TYPE stock_info_t ;
```

Lista de telefones

- **A lista de telefones será armazenada em um VARRAY**
- **No máximo 10 telefones são admitidos**

```
CREATE TYPE phone_list_t AS
                VARRAY(10) OF VARCHAR2(20) ;
```

- **Foi escolhido VARRAY ao invés de tabela aninhada:**
  - ordem dos números pode ser importante (tabela aninhada não é ordenada)
  - número de telefones é pequeno - VARRAY é armazenado junto com a linha que o contém
  - não há consultas sobre telefones

Endereço

- **Endereço será tratado como um objeto separado**

```
CREATE TYPE address_t AS OBJECT (
   street VARCHAR2(200),
   city VARCHAR2(200),
   state CHAR(2),
   zip VARCHAR2(20) ) ;
```

Definição de objetos tipo cliente

- **A instruçaõ abaixo cria um objeto tipo cliente (customer_info_t) a partir dos tipos definidos acima**

```
CREATE TYPE customer_info_t AS OBJECT (
  custno     NUMBER,
  custname   VARCHAR2(200),
  address    address_t,
  phone_list phone_list_t,

  ORDER MEMBER FUNCTION
    cust_order(x IN customer_info_t) RETURN INTEGER,

  PRAGMA RESTRICT_REFERENCES (
    cust_order,  WNDS, WNPS, RNPS, RNDS)
  ) ;
```

Tipo cliente

- **Um objeto tipo cliente é um agregado, pois contém outros objetos:**

```
...
  address    address_t,
  phone_list phone_list_t,
...
```

- O método ORDER serve para comparar dois clientes.

- A declaração PRAGMA contém informações sobre como os métodos acessam a base de dados (não discutido aqui).

Tipo linha de item

- **Uma linha de item é definida pelo tipo de objeto abaixo**

```
CREATE TYPE line_item_t AS OBJECT (
   lineitemno NUMBER,
   stockref   REF stock_info_t,
   quantity   NUMBER,
   discount   NUMBER
) ;
```

- **Observar que `stockref` é uma referência a um objeto de tipo `stock_info_t` (ainda não definido)**

Criação de um tipo de tabela

- **O comando abaixo cria um tipo de tabela, a partir de um tipo de objeto já definido:**

```
CREATE TYPE line_item_list_t AS
                        TABLE OF line_item_t ;
```

- **Este tipo é usado para definir uma tabela aninhada**

Tabela aninhada versus VARRAY

- **Optou-se por tabela aninhada ao invés de VARRAY pelos seguintes motivos:**
  - Provavelmente haverão consultas sobre os objetos item de linha - consultas são ineficientes sobre VARRAYs já que exigem acesso à linha pai
  - Indexação sobre as linhas de item é necessária - não é possível em VARRAY
  - Ordenação não é relevante - campo número da linha pode ser usado para ordenação
  - Não há limite superior para o número de linhas de item de um pedido

Definião do tipo pedido de compra

```
CREATE TYPE purchase_order_t AS OBJECT (
  pono            NUMBER,
  custref         REF customer_info_t,
  orderdate       DATE,
  shipdate        DATE,
  line_item_list  line_item_list_t,
  shiptoaddr      address_t,

  MAP MEMBER FUNCTION
    ret_value RETURN NUMBER,
    PRAGMA RESTRICT_REFERENCES (
      ret_value, WNDS, WNPS, RNPS, RNDS),

  MEMBER FUNCTION
    total_value RETURN NUMBER,
    PRAGMA RESTRICT_REFERENCES (total_value, WNDS, WNPS)
  ) ;
```

Tipo pedido

- Pedido contém uma referência para o cliente do pedido
- Pedido contém como tabela aninhada, o conjunto de itens do pedido
- O método ret_value (MAP método) é usado pelo SGBD para comparar dois objetos do tipo por igualdade
- O método total_value é um método de domínio de problema (calcular o total do pedido)

Definição do tipo produto

```
CREATE TYPE stock_info_t AS OBJECT (
  stockno    NUMBER,
  cost       NUMBER,
  tax_code   NUMBER
  ) ;
```

Corpo do método total_value de pedido

```
CREATE OR REPLACE TYPE BODY purchase_order_t AS
MEMBER FUNCTION total_value RETURN NUMBER IS
   i           INTEGER;
   stock       stock_info_t;
   line_item   line_item_t;
   total       NUMBER := 0;
   cost        NUMBER;
BEGIN
   FOR i IN 1..SELF.line_item_list.COUNT LOOP
      line_item := SELF.line_item_list(i);
      SELECT DEREF(line_item.stockref) INTO stock FROM DUAL;
      total := total + line_item.quantity * stock.cost ;
   END LOOP;
   RETURN total;
END;
END;
```

Corpo do método ret_value de pedido

```
CREATE OR REPLACE TYPE BODY purchase_order_t AS
   MAP MEMBER FUNCTION ret_value RETURN NUMBER IS
      BEGIN RETURN pono;
   END;
END;
```

- **Significa que o SGBD usa o número do pedido (pono) para compara dois pedidos por igualdade**

Corpo do método cust_order de cliente

```
CREATE OR REPLACE TYPE BODY customer_info_t AS

  ORDER MEMBER FUNCTION
  cust_order (x IN customer_info_t) RETURN INTEGER
  IS
  BEGIN
    RETURN custno - x.custno;
  END;

END;
```

- **Significa que um cliente é considerado maior que o outro (para fins de ordenação) caso seu número (`custno`) seja maior**

Criação de tabelas

- Comandos até este ponto são estritamente comandos de esquema
- Apenas tipos (classes) são definidos
- Para armazenar dados é necessário criar tabelas com objetos dos tipos definidos

Tabela de clientes

- **É criada com o comando**

```
CREATE TABLE customer_tab OF customer_info_t
 (custno PRIMARY KEY);
```

- **O comando define o tipo de linha da tabela**
- **A tabela possui uma chave primária**
  - Observar que a chave é da tabela e não do tipo
  - Se houvesse outra tabela com linhas do tipo, poderia ter outra chave primária

Endereço em cliente

- A coluna address contém objetos do tipo address_t.
- Objetos deste tipo possuem atributos de tipos pré-definidos
- O SGBD cria uma coluna para cada atributo de address_t
- Colunas de address_t são referenciadas com a notação de pontos:

```
address.street
```

Telefones de cliente

- A coluna phone_list contém os VARRAYs com on múmeros de telefone
- Cada VARRAY cotém no máximo 200 caracteres
- O SGBD armazena o VARRAY dentro da linha de cliente
- VARRAYS de mais de 4.000 caracteres são tratados como BLOBs

Tabela de produtos

```
CREATE TABLE stock_tab OF stock_info_t
 (stockno PRIMARY KEY) ;
```

Tabela de pedidos

```
CREATE TABLE purchase_tab OF purchase_order_t (
  PRIMARY KEY (pono),
  SCOPE FOR (custref) IS customer_tab
  )
  NESTED TABLE line_item_list STORE AS po_line_tab ;
```

- **Cada linha de pedido contém uma tabela aninhada line_item_list**
- **Esta tabela é armazenada em uma tabela separada (po_line_tab)**
- **É definido um escopo para a referência custref - ela referência apenas objetos da tabela customer_tab**

Completando o esquema

- **Delimitar o escopo dos REFs dentro de po_line_tab**

```
ALTER TABLE po_line_tab
  ADD (SCOPE FOR (stockref) IS stock_tab) ;
```

- **Parâmetros de armazenamento de po_line_tab**

```
ALTER TABLE po_line_tab
  STORAGE (NEXT 5K PCTINCREASE 5 MINEXTENTS 1
  MAXEXTENTS 20) ;
```

Criação de um índice para a tabela aninhada

```
CREATE INDEX po_nested_in
  ON         po_line_tab (NESTED_TABLE_ID) ;
```

- A tabela de armazenamento de uma coluna de tabela aninhada contém uma coluna oculta chamada NESTED_TABLE_ID
- Esta é uma referência à linha pai de cada linha aninhada
- O índice permite acesso eficiente às linhas aninhadas a partir da linha pai

Criação de um índice para a tabela aninhada

```
CREATE UNIQUE INDEX po_nested
  ON                  po_line_tab
                (NESTED_TABLE_ID, lineitemno) ;
```

- **Índice foi definido como UNIQUE**
- **É uma forma de especificar que a cada linha de item é identificada pelo seu pedido (`NESTED_TABLE_ID`) e pelo número da linha (`lineitemno`)**
- **Solução pouco elegante:**
  - mistura aspectos conceituais (chave primária) com questões de performance (índice)

Carga de valores - tabela de produtos

```
INSERT INTO stock_tab VALUES(1004, 6750.00, 2);
INSERT INTO stock_tab VALUES(1011, 4500.23, 2);
INSERT INTO stock_tab VALUES(1534, 2234.00, 2);
INSERT INTO stock_tab VALUES(1535, 3456.23, 2);
```

Carga de valores - tabela de clientes

```
INSERT INTO customer_tab
  VALUES (
    1, ‘Jean Nance’,
    address_t(‘2 Avocet Drive’, ‘Redwood Shores’, ‘CA’, ‘95054’)
    phone_list_t(‘415-555-1212’)
    ) ;

INSERT INTO customer_tab
  VALUES (
    2, ‘John Nike’,
    address_t(‘323 College Drive’, ‘Edison’, ‘NJ’, ‘08820’),
    phone_list_t(‘609-555-1212’,‘201-555-1212’)
    ) ;
```

Carga de valores - tabela de clientes

- **Observar o uso de construtores para montar os objetos endereço e lista de telefones:**

```
...
address_t('323 College Drive', 'Edison', 'NJ', '08820'),
phone_list_t('609-555-1212','201-555-1212')
...
```

Carga de valores - tabela de pedidos

```
INSERT INTO purchase_tab
  SELECT  1001, REF(C),
          SYSDATE,'10-MAY-1997',
          line_item_list_t(),
          NULL
   FROM   customer_tab C
   WHERE  C.custno = 1 ;
```

- **Observar**
  - uso de consulta para obter a referência a cliente
  - construtor da tabela aninhada vazia

Carga de valores - linhas da tabela aninhada

```
INSERT INTO THE (
  SELECT  P.line_item_list
   FROM   purchase_tab P
   WHERE  P.pono = 1001
  )
  SELECT  01, REF(S), 12, 0
   FROM   stock_tab S
   WHERE  S.stockno = 1534;
```

- **Observar**
  - cláusula THE para indicar que a inserção deve ser feita na tabela aninhada - linha pai é indicada por SELECT
  - uso de SELECT para obter o REF ao produto

Carga de valores - inserção de mais linhas aninhadas

```
INSERT INTO purchase_tab
  SELECT  2001, REF(C),
          SYSDATE,'20-MAY-1997',
          line_item_list_t(),
          address_t('55 Madison
  Ave','Madison','WI','53715')
   FROM   customer_tab C
   WHERE  C.custno = 2;

INSERT INTO THE (
  SELECT  P.line_item_list
   FROM   purchase_tab P
   WHERE  P.pono = 1001
  )
  SELECT  02, REF(S), 10, 10
   FROM   stock_tab S
   WHERE  S.stockno = 1535;
```

Carga de valores - inserção de mais linhas aninhadas

```
INSERT INTO THE (
  SELECT  P.line_item_list
   FROM   purchase_tab P
   WHERE  P.pono = 2001
  )
  SELECT  10, REF(S), 1, 0
   FROM   stock_tab S
   WHERE  S.stockno = 1004;

INSERT INTO THE (
  SELECT  P.line_item_list
   FROM   purchase_tab P
   WHERE  P.pono = 2001
  )
  VALUES( line_item_t(11, NULL, 2, 1) ) ;
```

Alteração de valores

```
UPDATE THE (
  SELECT  P.line_item_list
   FROM   purchase_tab P
   WHERE  P.pono = 2001
  ) plist

  SET plist.stockref =
   (SELECT REF(S)
     FROM  stock_tab S
     WHERE S.stockno = 1011
     )

  WHERE plist.lineitemno = 11 ;
```

- **Altera o produto da linha 11 do pedido de número 2001**

Uso de método de ordenação

```
SELECT  p.pono
 FROM   purchase_tab p
 ORDER BY VALUE(p);
```

- **Exemplo de uso de método de ordenação**
- **O SGBD utiliza o método definido pelo usuário para ordenação**

Exemplo de consulta - uso de DEREF

```
SELECT  DEREF(p.custref), p.shiptoaddr, p.pono,
        p.orderdate, line_item_list

 FROM   purchase_tab p

 WHERE  p.pono = 1001 ;
```

- obtém o cliente e os dados do pedido de número 1001

Exemplo de consulta - uso de método

```
SELECT   p.pono, p.total_value()

 FROM    purchase_tab p ;
```

Consulta - uso de Cursor

```
SELECT   po.pono, po.custref.custno,

         CURSOR (
           SELECT  *
            FROM   TABLE (po.line_item_list) L
            WHERE  L.stockref.stockno = 1004
           )

 FROM    purchase_tab po ;
```

Exclusão

```
DELETE
 FROM   purchase_order
 WHERE  pono = 1001 ;
```

- **Exclui o pedido e suas linhas aninhadas**