

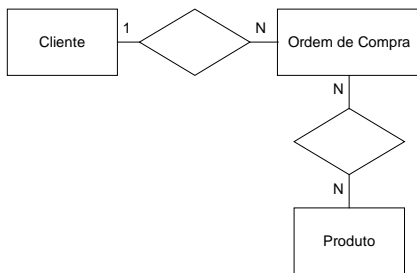
ORDBMS Oracle

Estudo de um caso exemplo

Descrição da aplicação

- Sistemas de pedidos
 - Clientes
 - Produtos
 - Ordens de compra
- Cliente tem um relacionamento 1:N com ordens de compra porque um cliente pode ter várias compras, mas uma dada ordem de compra se refere a apenas um cliente
- Ordem de compra tem um relacionamento N:N com produtos porque uma ordem de compra pode conter vários produtos, e um produto pode aparecer em várias ordens de compra

Diagrama ER



Modelo Relacional

- A maneira usual de mapear relacionamentos N:N, entre ordens de compra e produtos é introduzir uma entidade chamada item de compra.
- Uma ordem de compra pode ter um número qualquer de itens de compra, mas cada item de compra esta relacionado com apenas uma ordem de compra
- Um produto pode aparecer em vários itens de compra, mas cada item de compra refere-se a apenas um produto.

Modelo Relacional

Cliente	Informações para contato
Produto	Identificação, preço
Ordem de compra	Cliente, endereço, data da entrega
Item de compra	Produto, quantidade, preço, desconto

Tabelas no Modelo Relacional (em inglês)

Cliente	customer_info
Ordem de compra	purchase_order
Produto	stock_info
Item de Compra	line_items

Tabela Clientes

(em inglês)

```
CREATE TABLE customer_info (  
  custno NUMBER,  
  custname VARCHAR2(200),  
  street VARCHAR2(200),  
  city VARCHAR2(200),  
  state CHAR(2),  
  zip VARCHAR2(20),  
  phone1 VARCHAR2(20),  
  phone2 VARCHAR2(20),  
  phone3 VARCHAR2(20),  
  PRIMARY KEY (custno)  
) ;
```

Tabela Ordem de Compra

(em inglês)

```
CREATE TABLE purchase_order (  
  pono NUMBER,  
  custno NUMBER REFERENCES customer_info,  
  orderdate DATE,  
  shiptodate DATE,  
  shiptostreet VARCHAR2(200),  
  shiptocity VARCHAR2(200),  
  shiptostate CHAR(2),  
  shiptozip VARCHAR2(20),  
  PRIMARY KEY (pono)  
) ;
```

Tabela Produto

(em inglês)

```
CREATE TABLE stock_info (  
  stockno NUMBER PRIMARY KEY,  
  cost NUMBER,  
  tax_code NUMBER  
) ;
```

Tabela Item da Compra

(em inglês)

```
CREATE TABLE line_items (  
  lineitemno NUMBER,  
  pono NUMBER REFERENCES  
  purchase_order,  
  stockno NUMBER REFERENCES  
  stock_info,  
  quantity NUMBER,  
  discount NUMBER,  
  PRIMARY KEY (pono, lineitemno)  
) ;
```

Modelo Relacional

X

Modelo Objeto-Relacional

- Endereço e lista de telefones de contato dos clientes
 - Modelo Relacional
 - colunas independentes nas tabelas
 - Modelo Objeto-Relacional
 - Novos tipos são definidos
- Itens de Compra
 - Modelo Relacional
 - Tabela separa
 - Modelo Objeto-Relacional
 - Nested table junto a ordem de compra

Modelo Objeto-Relacional

- Objetos do modelo Objeto-Relacional
 - Cliente
 - Produto
 - Ordem de Compra

Criação dos novos tipos

- CREATE TYPE **line_item_t**;
- CREATE TYPE **purchase_order_t**;
- CREATE TYPE **stock_info_t**;

Criação do tipo **phone_list_t**

- CREATE TYPE **phone_list_t** AS
VARRAY(10) OF VARCHAR2(20) ;
 - Poderia ser Nested Table?
 - VARRAY é a melhor escolha:
 - Lista ordenada
 - Número pequeno e conhecido de valores
 - Nenhuma consulta será feita sobre a lista de telefones

Criação do tipo **address_t**

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

Criação do tipo **customer_info_t**

```
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)  
) ;
```

Criação dos tipos **line_item_t** e **line_item_list_t**

```
CREATE TYPE line_item_t AS OBJECT (  
    lineitemno NUMBER,  
    STOCKREF REF stock_info_t,  
    quantity NUMBER,  
    discount NUMBER  
) ;  
CREATE TYPE line_item_list_t AS TABLE OF  
line_item_t ;  
– Nested Table é a melhor escolha  
• Pode ser feita alguma consulta sobre a lista de itens  
• Pode ser indexada  
• A ordem dos itens não importa  
• Não é conhecido o tamanho máximo
```

Criação do tipo **purchase_order_t**

```
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)  
) ;
```

Criação do tipo `stock_info_t`

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

Criação dos métodos de `purchase_order_t`

```
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;  
MAP MEMBER FUNCTION ret_value RETURN NUMBER IS  
BEGIN  
    RETURN pono;  
END;  
END;
```

Criação dos métodos de `customer_info_t`

```
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;
```

Criação das TABELAS

- Os tipos definidos não permitem o armazenamento de dados
- Somente as tabelas permitem que sejam armazenados dados
- Como objetos se relacionam com tabelas:
 - Classe, que representam entidades, são mapeadas em tabelas;
 - Atributos são as colunas
 - Objetos são as linhas

Criação da tabela `customer_tab`

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

- A tabela `CUSTOMER_TAB` tem as colunas `CUSTNO`, `CUSTNAME`, `ADDRESS`, `PHONE_LIST` e `PO_LIST`
- E cada linha é um objeto do tipo `CUSTOMER_INFO_T`
- Esta noção de objeto (*row object*) agrega funcionalidades
- Reusabilidade através do tipo `customer_info_t`
 - Ex.: `customer_tab2`
- As constraints aplicam-se às tabelas e não aos tipos

Criação da tabela `stock_tab`

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

- cada linha da tabela `STOCK_TAB` é um objeto do tipo `STOCK_INFO_T`

Criação da tabela `purchase_tab`

```
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 ;
```

Criação da tabela `purchase_tab`

- SCOPE FOR (`custref`) IS `customer_tab`
 - `custref` é do tipo REF `customer_info_t`
 - Limita as referencias somente a objetos que estejam na tabela `customer_tab`
- NESTED TABLE `line_item_list`
STORE AS `po_line_tab`
 - Cria a tabela `po_line_tab` para armazenar as colunas do atributo `line_item_list` de todas as linhas de `purchase_tab`
 - É uma tabela separada
 - Relacionamento feito através do atributo NESTED_TABLE_ID
 - Podem ser usadas para evitar `joins`

Inserção de valores em `stock_tab`

```
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);
```

Inserção de valores em `customer_tab`

```
INSERT INTO customer_tab VALUES (
    1, 'Jean Nance',
    address_t('2 Avocet Drive', 'Red 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')
);
```

Inserção de valores em `purchase_tab`

```
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 ;
```

- Construção de um objeto `PURCHASE_ORDER_T` com os seguintes valores:
 - `pono` 1001
 - `custref` REF to customer number 1
 - `orderdate` SYSDATE
 - `shipdate` 10-MAY-1997
 - `line_item_list` an empty `line_item_list_t`
 - `shiptoaddr` NULL
- A instrução usa uma consulta para construir uma referência (REF) ao objeto que está na tabela de objetos `CUSTOMER_TAB` que tem o `CUSTNO` com o valor 1

Inserção de valores em `line_item_list`

```
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;
```

- A instrução usa uma subconsulta, sinalizada pela palavra `THE`, para identificar o destino da inserção
 - a nested table `LINE_ITEM_LIST` do objeto armazenado dentro na tabela `PURCHASE_TAB` com `PONO` de valor 1001
- O item inserido contém uma referência (REF) ao objeto, da tabela `STOCK_TAB`, que tem o `STOCKNO` com o valor 1534

Inserção de valores em purchase_tab e line_item_list

```
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 = 2001
)
SELECT 02, REF(S), 10, 10
FROM stock_tab S
WHERE S.stockno = 1535;
```

Inserção de valores em purchase_tab e line_item_list

```
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 ;
```

Consulta em purchase_tab

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

- Invoca o método RET_VALUE de PURCHASE_ORDER_T

Consulta em purchase_tab (1)

- **Cliente e Item da Compra para a Ordem de Compra 1001**

```
SELECT Deref(p.custref),p.shiptoaddr,
p.pono,p.orderdate, line_item_list
FROM purchase_tab p
WHERE p.pono = 1001 ;
```

Consulta em purchase_tab (2)

- **Total de cada Ordem de Compra**

```
SELECT p.pono, p.total_value()
FROM purchase_tab p ;
```

Consulta em purchase_tab (3)

- **Ordem de Compra e Item de Compra relacionados com o produto 1004**

```
SELECT po.pono, po.custref.custno,  
CURSOR (  
  SELECT *  
  FROM TABLE (po.line_item_list) L  
  WHERE L.stockref.stockno = 1004  
)  
FROM purchase_tab po ;
```

Remoção em purchase_tab

- **Deletar a Ordem de Compra 1001**

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

- Neste caso, todas as informações daquela ordem de compra são removidas