

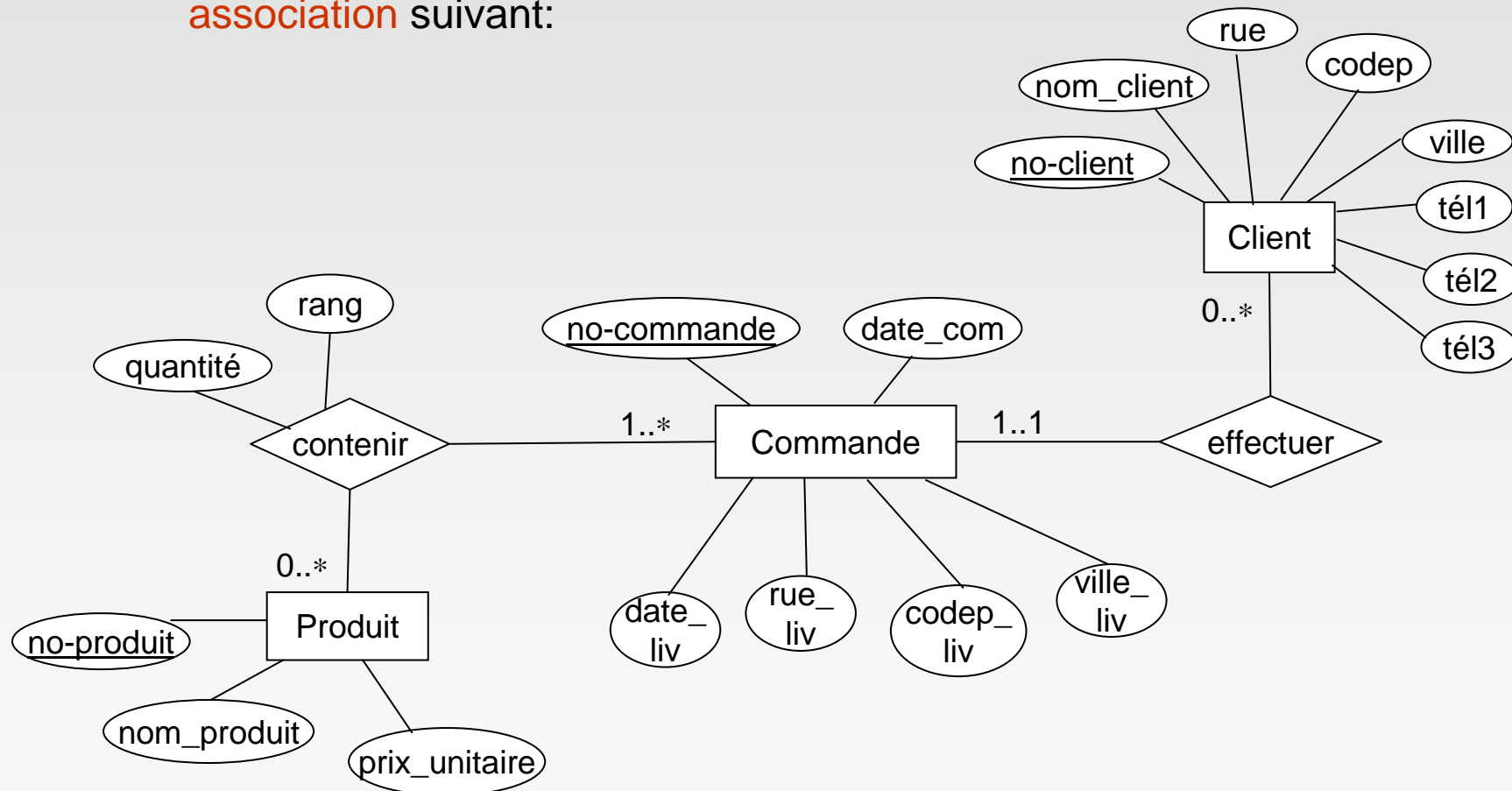
Chapitre 1b: Modèle relationnel-objet exemple

Contenu

- Modélisation EA
- Implémentation en relationnel: schéma, instance
- Implémentation en relationnel-objet: schéma, instance
- Requêtes en relationnel-objet et en relationnel

Modélisation EA

- On souhaite stocker des informations concernant des commandes de produits effectuées par des clients
- On modélise le champ d'application et on obtient le **schéma entité-association** suivant:



EA → relationnel

- On traduit le schéma EA en un **schéma relationnel**:
 - Client (no_client, nom_client, rue, codep, ville, tél1, tél2, tél3)
 - Produit (no_produit, nom_produit, prix_unitaire)
 - Commande (no_commande, date_com, date_liv, rue_liv, codep_liv, ville_liv, no_client)
 - Contenir (no_commande, no_produit, rang, quantité)
 - ▶ autre clé: {no_commande, rang}

Implémentation en relationnel

■ Création de tables relationnelles:

- CREATE TABLE **Client** (
 no_client INTEGER PRIMARY KEY,
 nom_client VARCHAR2(200) NOT NULL,
 rue VARCHAR2(200) NOT NULL,
 codep VARCHAR2(20) NOT NULL,
 ville VARCHAR2(200) NOT NULL,
 tel1 VARCHAR2(20),
 tel2 VARCHAR2(20),
 tel3 VARCHAR2(20));
- CREATE TABLE **Produit** (
 no_produit INTEGER PRIMARY KEY,
 nom_produit VARCHAR2(200) NOT NULL,
 prix_unitaire NUMBER(6,2));

Implémentation en relationnel (suite)

- CREATE TABLE **Commande** (
 no_commande INTEGER PRIMARY KEY,
 date_com DATE,
 date_liv DATE, /* date de livraison */
 /* adresse de livraison */
 rue_liv VARCHAR2(200) NOT NULL,
 codep_liv VARCHAR2(20) NOT NULL,
 ville_liv VARCHAR2(200) NOT NULL,
 no_client INTEGER references Client);
- CREATE TABLE **Contenir** (
 no_commande INTEGER references Commande,
 no_produit INTEGER references Produit,
 rang INTEGER,
 quantité INTEGER,
 PRIMARY KEY (no_commande, no_produit));

Implémentation en relationnel (suite)

■ Insertion de données dans les tables relationnelles:

- ```
INSERT INTO Produit VALUES (1004, 'grande table', 67.50);
INSERT INTO Produit VALUES (1011, 'chaise', 45.00);
INSERT INTO Produit VALUES (1534, 'petite table', 22.35);
INSERT INTO Produit VALUES (1535, 'tabouret', 34.55);
```
- ```
INSERT INTO Client VALUES (1, 'Jean Dupont', '24 Dufour',
                             '1204', 'Geneve', '022-1234567', NULL, NULL);
INSERT INTO Client VALUES (2, 'Michel Durand', '20 Rhone',
                             '1204', 'Geneve', '022-2345678', '079-5678901',
                             NULL);
```
- ```
INSERT INTO Commande VALUES (1001, SYSDATE, '10-MAY-2007',
 NULL, NULL, NULL, 1);
INSERT INTO Commande VALUES (2001, SYSDATE, '20-MAY-2007',
 '10 Bel-Air', '1204', 'Geneve', 2);
```
- ```
INSERT INTO Contenir VALUES (1001, 1534, 1, 12);
INSERT INTO Contenir VALUES (1001, 1535, 2, 10);
INSERT INTO Contenir VALUES (2001, 1004, 1, 1);
INSERT INTO Contenir VALUES (2001, 1011, 2, 6);
```

Implémentation en relationnnel-objet

- créer un type objet T_Produit, puis définir une table d'objets Produit pour pouvoir référencer des produits

- ```
CREATE TYPE T_Produit AS OBJECT (
 no_produit INTEGER,
 nom_produit VARCHAR2(200),
 prix_unitaire NUMBER(6,2))

/
```

- ```
CREATE TABLE Produit OF T_Produit  
    (no_produit PRIMARY KEY)  
    OBJECT IDENTIFIER IS PRIMARY KEY;
```

- ▶ on peut spécifier que l'oid des objets de la table Produit est généré par le système (OBJECT IDENTIFIER IS SYSTEM GENERATED, par défaut, 16 bytes) ou corresponde à la clé primaire de la ligne

Implémentation en relationnnel-objet (suite)

- créer un type objet T_Adresse, puis l'utiliser pour l'adresse d'un client et l'adresse de livraison d'une commande

- ```
CREATE TYPE T_Adresse AS OBJECT (
 rue VARCHAR2(200) ,
 codep VARCHAR2(20) ,
 ville VARCHAR2(200))

/
```

- créer un type collection T\_liste\_tel, puis l'utiliser pour les téléphones d'un client

- ```
CREATE TYPE T_liste_tel AS VARRAY(3) OF VARCHAR2(20)  
  
/
```

Implémentation en relationnnel-objet (suite)

- créer un type objet T_client, puis définir une table d'objets Client pour pouvoir référencer des clients

- ```
CREATE TYPE T_Client AS OBJECT (
 no_client INTEGER,
 nom_client VARCHAR2(200),
 adresse T_Adresse,
 telephones T_liste_tel) NOT FINAL
/
```

- ▶ NOT FINAL permet de créer plus tard des sous-types de Client. Par défaut les types sont créés comme FINAL.

- ```
CREATE TABLE Client OF T_Client  
    (no_client PRIMARY KEY)  
    OBJECT IDENTIFIER IS PRIMARY KEY;
```

Implémentation en relationnnel-objet (suite)

- but: faire que les produits commandés fassent partie de la commande (composition).

Créer un type objet T_Contenir, puis créer une table imbriquée de ce type et l'utiliser dans T_Commande

- ```
CREATE TYPE T_Contenir AS OBJECT (
 ref_produit REF T_Produit,
 rang INTEGER,
 quantité INTEGER)

/

CREATE TYPE T_ens_Contenir AS TABLE OF T_Contenir

/
```

mieux que VARRAY:

- ▶ possibilité d'interroger et de manipuler l'ensemble des produits d'une commande
- ▶ ordre des produits dans l'attribut rang
- ▶ pas de limite concernant le nombre de produits d'une commande

# Implémentation en relationnnel-objet (suite)

- CREATE TYPE T\_Commande AUTHID CURRENT\_USER AS OBJECT (  
    no\_commande INTEGER,  
    date\_com DATE,  
    date\_liv DATE,  
    adresse\_liv T\_Adresse,  
    tab\_produits T\_ens\_Contenir,  
    ref\_client REF T\_Client  
  
    MEMBER FUNCTION montant RETURN NUMBER )  
/  
  
● CREATE TABLE Commande OF T\_Commande (  
    PRIMARY KEY (no\_commande),  
    FOREIGN KEY (ref\_client) REFERENCES Client)  
    OBJECT IDENTIFIER IS PRIMARY KEY  
    NESTED TABLE tab\_produits STORE AS produits\_com (  
        (PRIMARY KEY (NESTED\_TABLE\_ID, rang)) );

# Implémentation en relationnnel-objet (suite)

- Dans la table imbriquée *produits\_com*, il y a une référence à un produit
  - ▶ pas possible de définir une contrainte « foreign key » dans une table imbriquée, donc définir une contrainte « scope »
  - ▶ `ALTER TABLE produits_com  
ADD (SCOPE FOR (ref_produit) IS Produit);`
  - ▶ différence avec FK: pas de dépendance entre une référence et un objet référencé
    - si un produit est dans une commande, et que ce produit est supprimé de la table Produit, la référence à ce produit devient DANGLING REF

# Implémentation en relationnnel-objet (suite)

## ■ Méthode *montant*:

- CREATE TYPE BODY T\_Commande AS

```
MEMBER FUNCTION montant RETURN NUMBER IS
 i INTEGER;
 un_produit T_Produit;
 total NUMBER := 0;
BEGIN
 FOR i IN 1..SELF.tab_produits.COUNT LOOP
 UTL_REF.SELECT_OBJECT(
 tab_produits(i).ref_produit, un_produit);
 total := total + SELF.tab_produits(i).quantite
 * un_produit.prix_unitaire;
 END LOOP;
 RETURN total;
END;

END;
/
```

# Implémentation en relationnnel-objet (suite)

- **self**: mot-clé, pour référencer l'objet
- **count**: mot-clé, donne le nombre d'éléments dans une table PL/SQL ou un tableau (array)
- Méthode **select\_object** du package **utl\_ref**: pour déréférencer une référence d'objet (REF), car ce pas implicite dans PL/SQL. Ici pour obtenir l'objet produit correspondant à la référence `ref_produit`
- **authid current\_user**: les méthodes du type seront exécutées avec les droits de l'utilisateur courant, et pas les droits de l'utilisateur qui a défini le type

# Implémentation en relationnnel-objet (suite)

## ■ Insertion de données dans les tables objet:

- ```
INSERT INTO Produit VALUES (1004, 'gde table', 67.50);
INSERT INTO Produit VALUES (1011, 'chaise', 45.00);
INSERT INTO Produit VALUES (1534, 'pt table', 22.35);
INSERT INTO Produit VALUES (1535, 'tabouret', 34.55);
```
- ```
INSERT INTO Client VALUES (1, 'Jean Dupont',
 T_Adresse('24 Dufour', '1204', 'Geneve'),
 T_liste_tel('022-1234567'));

INSERT INTO Client VALUES (2, 'Michel Durand',
 T_Adresse('20 Rhone', '1204', 'Geneve'),
 T_liste_tel('022-2345678', '079-5678901'));
```

# Implémentation en relationnnel-objet (suite)

- INSERT INTO Commande  
SELECT 1001, SYSDATE, '10-MAY-2007', NULL,  
tab\_produits(), REF(C)  
FROM Client C  
WHERE C.no\_client = 1;  
  
▶ insère une commande no 1001, avec la référence au client no 1, et aucun produit (table imbriquée vide)

```
INSERT INTO TABLE(SELECT tab_produits
 FROM Commande
 WHERE no_commande = 1001)
SELECT REF(P), 1, 12
FROM Produit P
WHERE P.no_produit = 1534;
```

- ▶ ajoute un produit dans la commande no 1001 (une ligne dans la table imbriquée): la référence au produit no 1534, rang 1, quantité 12

# Implémentation en relationnnel-objet (suite)

```
INSERT INTO TABLE(SELECT tab_produits
 FROM Commande
 WHERE no_commande = 1001)
SELECT REF(P), 2, 10
FROM Produit P
WHERE P.no_produit = 1535;
```

- ▶ ajoute le deuxième produit à la commande no 1001

# Implémentation en relationnnel-objet (suite)

- ```
INSERT INTO Commande
      SELECT 2001, SYSDATE, '20-MAY-2007',
             T_Adresse('10 Bel-Air', '1204', 'Geneve'),
             tab_produits(), REF(C)
FROM      Client C
WHERE     C.no_client = 2;
```

```
INSERT INTO TABLE(SELECT tab_produits
                     FROM      Commande
                     WHERE     no_commande = 2001)
SELECT  REF(P), 1, 1
FROM      Produit P
WHERE     P.no_produit = 1004;
```

```
INSERT INTO TABLE(SELECT tab_produits
                     FROM      Commande
                     WHERE     no_commande = 2001)
SELECT  REF(P), 2, 6
FROM      Produit P
WHERE     P.no_produit = 1011;
```

Implémentation en relationnnel-objet (suite)

Commande

no_ com	date_ com	date_ liv	adresse_liv			tab_produits	ref_ client									
			rue_liv	codep_liv	ville_liv											
2001	20-MAR-07	20-MAY-07	10 Bel-Air	1204	Geneve	<table><tr><th>ref_ produit</th><th>rang</th><th>quan tite</th></tr><tr><td></td><td>1</td><td>1</td></tr><tr><td></td><td>2</td><td>6</td></tr></table>	ref_ produit	rang	quan tite		1	1		2	6	
ref_ produit	rang	quan tite														
	1	1														
	2	6														

Produit

no_produit	nom_produit	prix_unitaire
1004	gde table	67.50
1011	chaise	45.00

Client

no_client	nom_client	adresse			telephones
		rue	codep	ville	
2	Michel Durand	20 Rhone	1204	Geneve	022-2345678
					079-5678901

Requêtes

- donner le client, l'adresse de livraison, la date de commande, et les produits de la commande no 1001

- ```
SELECT Deref(ref_client), adresse_liv, date_com,
 tab_produits
FROM Commande
WHERE no_commande = 1001;
```

- relationnel:

```
SELECT cl.*, co.rue_liv, co.codep_liv, co.ville_liv,
 co.date_com, c.*
FROM Commande co, Client cl, Conteneur c
WHERE co.no_commande = 1001 AND
 cl.no_client = co.no_client AND
 c.no_commande = co.no_commande
```

# Requêtes (suite)

## ■ donner le montant de chaque commande

- ```
SELECT no_commande, montant()  
FROM   Commande
```

- relationnel:

```
SELECT co.no_commande,  
       SUM(p.prix_unitaire * ct.quantite)  
FROM   Commande co, Produit p, Contenir ct  
WHERE  ct.no_commande = co.no_commande AND  
       p.no_produit = ct.no_produit  
GROUP BY co.no_commande
```

Requêtes (suite)

- donner le no de commande, le rang et la quantité du produit no 1004 pour chaque commande où il apparaît

- ```
SELECT c.no_commande, p.rang, p.quantite
FROM Commande c, TABLE(c.tab_produits) p
WHERE Deref(p.ref_produit).no_produit = 1004;
```

- relationnel:

```
SELECT no_commande, rang, quantite
FROM Contenir
WHERE no_produit = 1004;
```

# Requêtes (suite)

- modifier la quantité du produit no 1534 dans la commande no 1001 à 20

- ```
UPDATE TABLE(SELECT tab_produits
                  FROM    Commande
                  WHERE   no_commande = 1001)
SET    quantite = 20
WHERE  Deref(ref_produit).no_produit = 1534;
```

- relationnel:

```
UPDATE Contenir
SET    quantite = 20
WHERE  no_commande = 1001 AND no_produit = 1534;
```

Requêtes (suite)

■ supprimer la commande no 1001

- ```
DELETE FROM Commande
WHERE no_commande = 1001;
```

- relationnel: 2 ordres de suppression

```
DELETE FROM Contenir
WHERE no_commande = 1001;
```

```
DELETE FROM Commande
WHERE no_commande = 1001;
```