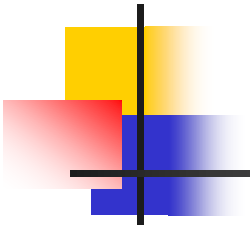
Bases de données et SGBD orientés Objet

Chapitre 3: Les SGBD relationnel/objet: **SQL3**

Z. Alimazighi, M. Azzouz



PLAN

1 . Rappels

1.1 Marché BD et standards

1.2 Modèle relationnel : Forces & Faiblesses

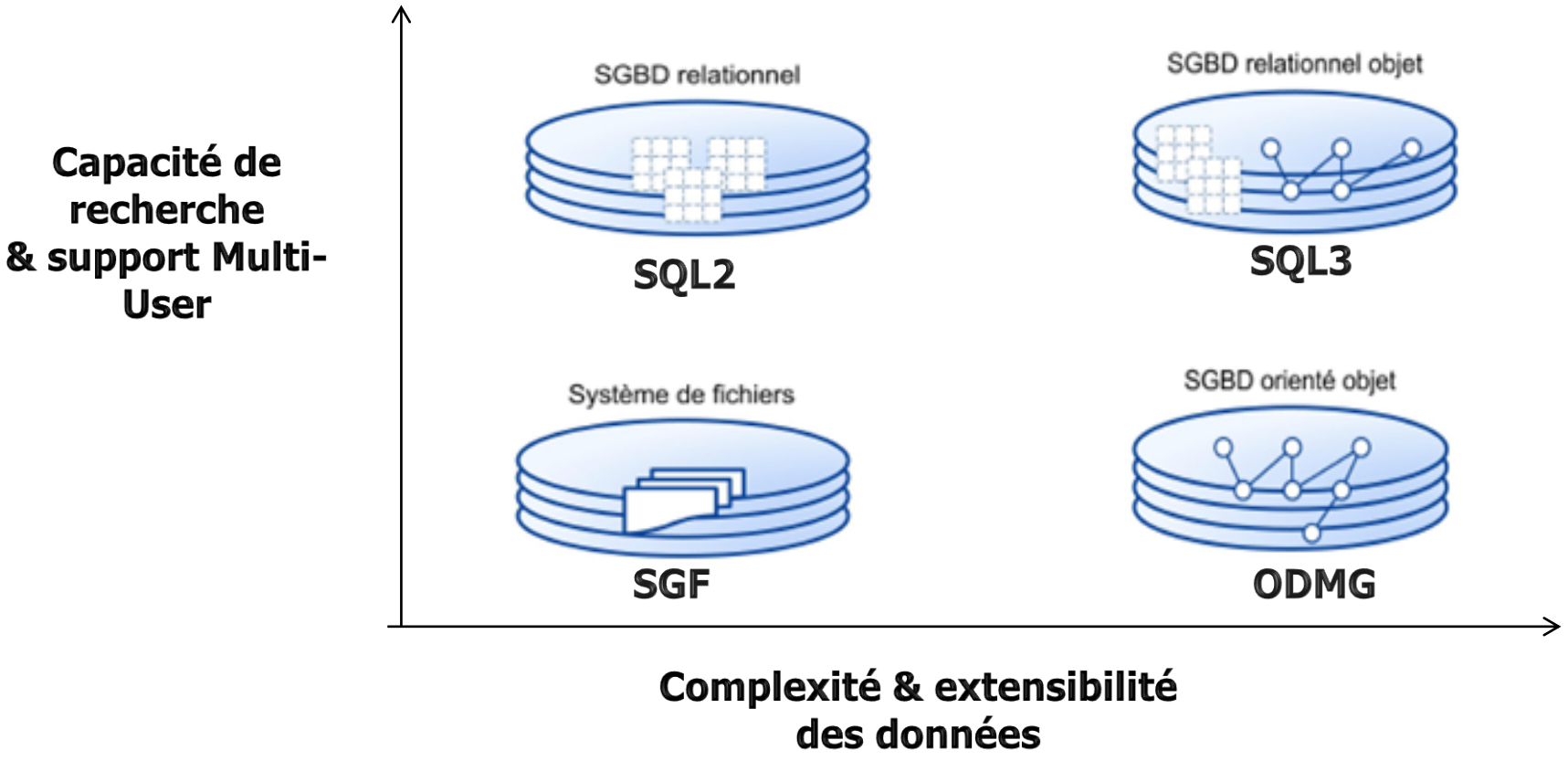
1.3 Modèle objet : Forces & Faiblesses

2. Modèle Relationnel objet et SQL3

3. Conclusion

1. Rappels

1.1 Marché BD et standards (Stonebraker 96 & Gartner)



1. Rappels

1.2 Modèle relationnel :

Forces :

1. Facilite l'optimisation des requêtes en appliquant des théorèmes de l'algèbre relationnelle.

2. Indépendance des données et de programmes de traitement $\Rightarrow$ partage d'information entre les applications.

3. Spécification logique des liens entre les tables.

4. Assurer l'intégrité de la BD par le concept de transaction et l'accès concurrent à la BD par le concept de verrou.

5. Le langage de requête : SQL qui permet l'interrogation, mise à jour des données.

6. …

1. Rappels

1.2 Modèle relationnel :

Faiblesses :

1. Dissociation des données et leurs comportements.
2. Multiplicité de nombre de tables par le respect des formes normales.
3. Nombre de type de base restreint et non extensible.
4. Les types CLOB, BLOB n'ont pas manipulables.
5. Faible capacité de modélisation (collection, graphes, hiérarchies).

1. Rappels

1.3 Modèle objet :

Forces :

1. Liaison dynamique : permet de spécifier des algorithmes de traitement appliqués à une famille de classe.

2. Un langage de programmation et un modèle de données unique.

3. La richesse des types de données et la possibilité de créer de nouveau types.

4. Grande capacité de modélisation.

5. Possibilité de concevoir des composants logiciels réutilisables permettant ainsi de construire des applications par construction.

1. Rappels

1.3 Modèle objet :

Faiblesses :

1. Manque de base théorique dans la systématisation de la conception d'un modèle objet.

2 . Problème de gestion de persistance des données.

3. Tenu de charge des moteurs de SGBDOO, lorsque le nombre des utilisateurs concurrents augmente d'une manière importante.

2. Modèle relationnel objet et SQL3

Constats :

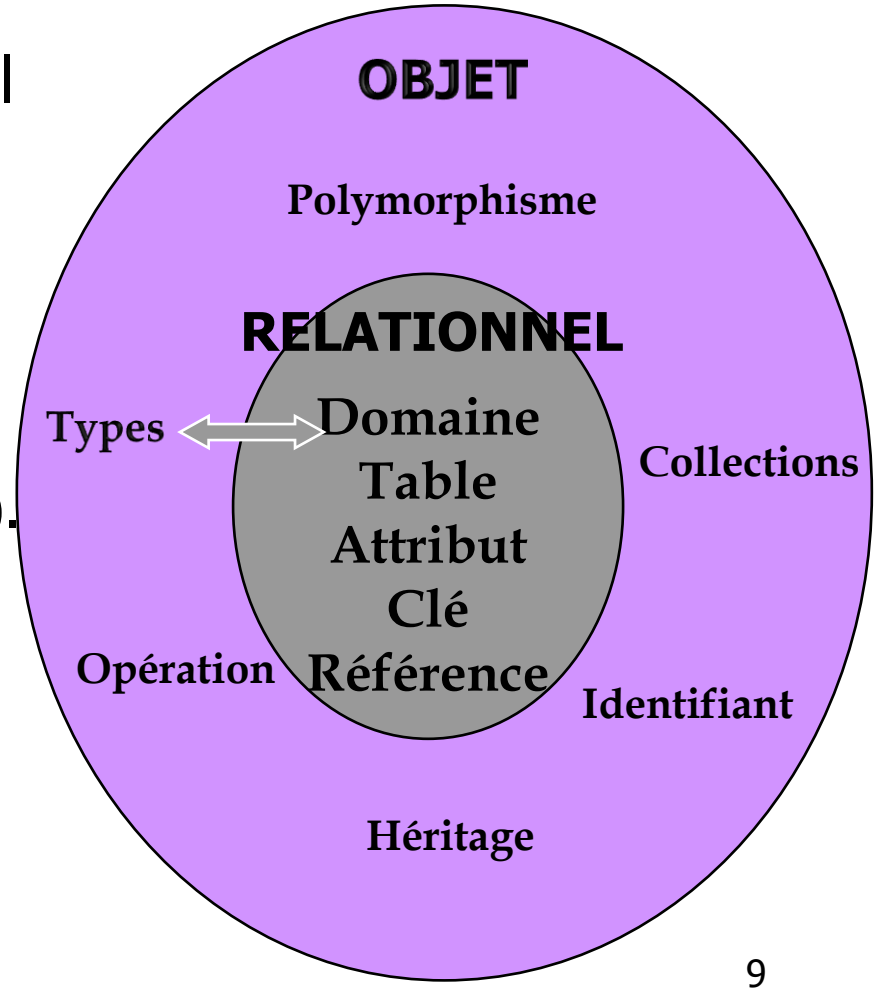
- Les bases de données *sont* relationnelles.
- Les grands constructeurs de SGBD-R sont riches et puissants.
- Le modèle relationnel est très limité car :
  - ✓ Il n'existe pas de constructeurs de types.
  - ✓ Le nombre de types prédéfinis est très faible.
- Les nouvelles applications (en particulier multimédia) exigent plus de souplesse.

=> Réaction des constructeurs de SGBD-R :
les SGBD relationnels-objets (ou universels).

2. Modèle relationnel objet et SQL3

Extension du modèle relationnel

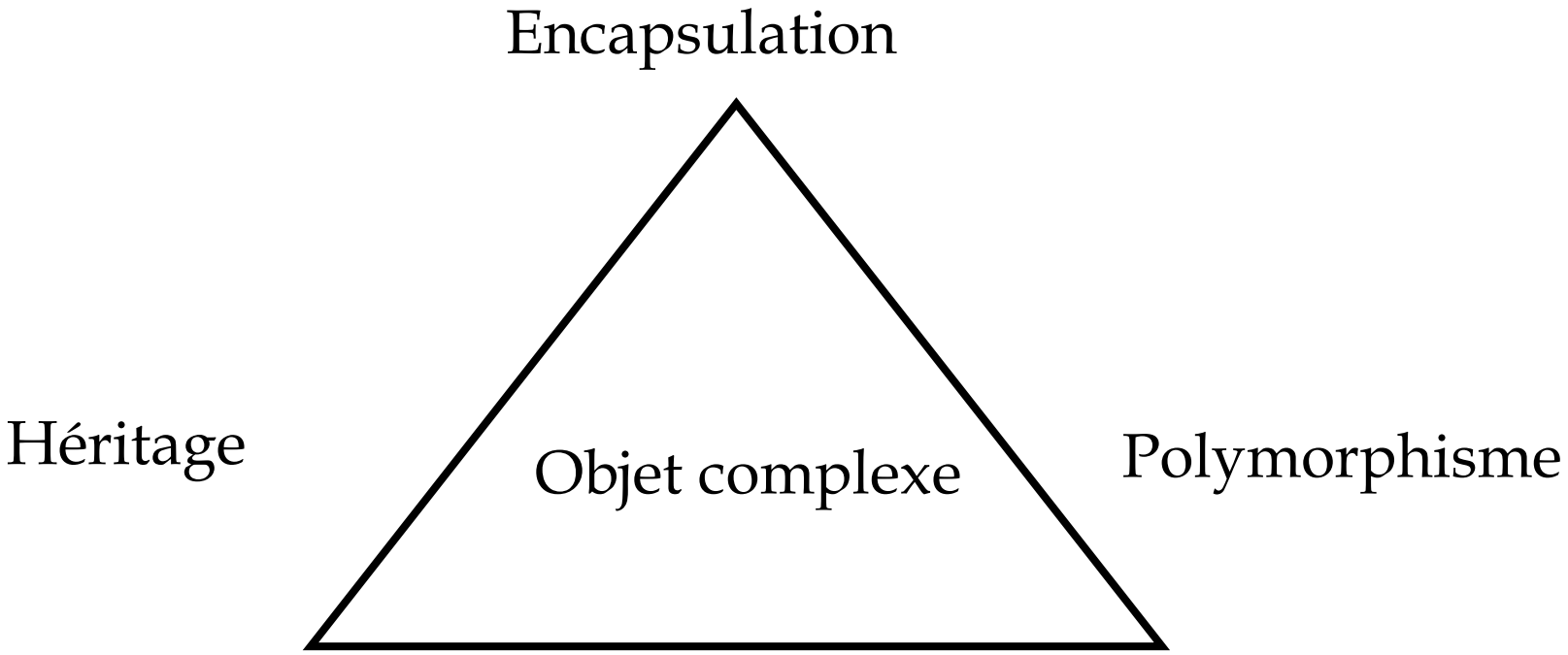
- Attributs multivalués : structure, liste, tableau, ensemble.
- Héritage sur tables et types
- Domaine type abstrait de Données (structure cachée + méthodes).
- Identité d'objets.



2. Modèle relationnel objet et SQL3

- ✓ SQL3 est une extension du SQL2.

Les nouveaux concepts orienté objet ajoutés sont :



SQL3 (Suite) :

Dans cet ordre d'idée nous examinons quelques caractéristiques qui sont :

- ✓ Constructeurs de types.
- ✓ Types définis par les utilisateurs.
- ✓ Procédures, fonctions, méthodes et opérateurs définis par l'utilisateur.
- ✓ Prise en compte des vastes objets (BLOB, CLOB).

SQL3 (Suite) :

2.1 Type « ligne » :

type ligne (de données) est une séquence de paires de nom de champ et de type de donnée, qui forme un type de donnée représentant les types des lignes.

❑ Un type de ligne permet de déposer dans une colonne d'une table des valeurs de lignes.

⇒

- ✓ Emmagasiner des lignes complètes dans des variables.
- ✓ Passer de telles variables en argument de routines.
- ✓ Retourner des valeurs de ce genre à partir d'appels de fonctions.

SQL3 (Suite) :

2.1 Type « ligne »: EXEMPLE

```
CREATE TABLE Filiale ( numFiliale CHAR(4),
 adresse ROW (rue VARCHAR(25), ville VARCHAR(15),
 codePostal ROW (identificateurVille VARCHAR(4),
quartieVARCHAR(4)  )));


INSERT INTO Filiale
VALUES ('F005', ROW('22 Bd de l'Impératrice', 'Paris', ROW('750',
'015')));
```

SQL3 (Suite) :

2.2 Type « Collection » :

- Les **collections** sont des constructeurs de types qui permettent de définir des collections d'autres types.
- Les collections présentent l'intérêt essentiel de stocker des valeurs multiples dans une seule colonne d'une table.
- Les types collection ont les significations suivantes :
  - ✓ ARRAY
  - ✓ MULTISET
  - ✓ LIST
  - ✓ SET

SQL3 (Suite) :

2.2 Type « collection » : EXEMPLE (ARRAY)

NumTél VARCHAR(13) ARRAY[3]

```
SELECT NumTél[1]
FROM Filiale
WHERE numFiliale = 'F003';
```

SQL3 (Suite) :

2.3 TDU :

- *TDU* : Types Définis par Utilisateur.
- Les TDU se subdivisent en deux catégories :
  - Types distincts.
  - Types structurés.
- Ils s'utilisent de la même manière que les types prédéfinis.

Types distincts :

Permet une différenciation entre les mêmes types de base sous-jacents

```
CREATE TYPE TypeNuméroPropriétaire AS VARCHAR(5) FINAL;
CREATE TYPE TypeNuméroPersonnel AS VARCHAR(5) FINAL;
```

Si nous tentons ensuite de traiter une instance d'un type comme une instance de l'autre type, alors le système génère une erreur.

SQL3 (Suite) :

2.3 TDU : (Suite)

Types structurés :

- Constituer d'un ou plusieurs attributs (type prédéfini, TDU).
- Tous les aspects du comportement sont fournis par des méthodes, fonctions, procédures définies par l'utilisateur.
- « Get » et « Set » sont des fonction prédéfinies permettant l'accès aux valeurs des attributs.
- La comparaison entre les valeurs se fait seulement par des fonctions définies par l'utilisateur.
- La valeur d'un attribut est accessible par l'entremise de la notation point (.).
- Extension de type : la méthode de construction New.

SQL3 (Suite) :

2.3 TDU : (Exemple)

```
CREATE TYPE TypePersonne AS ( dateNaissance DATE, prénom
VARCHAR(15), nom VARCHAR(15), genre CHAR)

INSTANTIABLE
NOT FINAL
REF IS SYSTEM GENERATED

INSTANCE METHOD age() RETURNS INTEGER,
INSTANCE METHOD age (DDN DATE) RETURNS TypePersonne;
CREATE INSTANCE METHOD age() RETURNS INTEGER
FOR TypePersonne
BEGIN
RETURN /* Âge calculé à partir de SELF.dateNaissance. */;
END;
```

SQL3 (Suite) :

2.4 Types de référence et identité d'objet :

- Type de référence offre des fonctionnalités semblables à celles de l'identificateur d'objet (IDO) d'un SGBD orienté objet.
- La valeur d'un type de référence est stockée dans une table.
- Les types référence permettent de :
  - Définir des associations entre types ligne et d'identifier de façon unique une ligne d'une table.
  - Partager une ligne parmi plusieurs tables et, aux utilisateurs.
  - Remplacer des définitions de jointures complexes dans des requêtes par des expressions de chemin plus simple.
  - Donner à l'optimiseur une alternative pour naviguer parmi les données à la place de jointures en fonction des valeurs

SQL3 (Suite) :

2.4 Types de référence et identité d'objet : (Exemple)

```
CREATE TABLE PropriétéALouer( numPropriété NuméroPropriété
NOT NULL, rue Rue NOT NULL, ville Ville NOT NULL,
codePostal CodePostal, type TypePropriété NOT NULL DEFAULT 'A',
pièces PiècesPropriété NOT NULL DEFAULT 4,
location LocationPropriété NOT NULL DEFAULT 600,

 IDPersonnel  REF(TypePersonnel) SCOPE Personnel
REFERENCES ARE CHECKED ON DELETE CASCADE,

PRIMARY KEY (numPropriété));
```

SQL3 (Suite) :

2.5 Sous-types et super-types :

- TDU à participer à une hiérarchie de sous-types et de supertypes à l'aide de la clause UNDER.
- L'héritage multiple n'est pas autorisé.
- Un sous-type hérite de tous les attributs et du comportement (les méthodes) de son super-type.
- Un sous-type peut définir des attributs et des fonctions supplémentaires comme tout autre TDU.
- Un sous-type peut écraser des fonctions héritées (substitution).

SQL3 (Suite) :

2.5 Sous-types et super-types : (Exemple)

```
CREATE TYPE TypePersonnel UNDER TypePersonne AS (
numPersonnel VARCHAR(5), fonction VARCHAR(10) DEFAULT 'Assistant', salaire
DECIMAL(7, 2), numFiliale CHAR(4))
INSTANTIABLE
NOT FINAL
INSTANCE METHOD estGérant () RETURNS BOOLEAN;
CREATE INSTANCE METHOD estGérant (e TypePersonnel)
RETURNS BOOLEAN
FOR TypePersonnel
BEGIN
IF SELF.fonction = 'Gérant' THEN
RETURN TRUE;
ELSE
RETURN FALSE;
END IF
END)
```

SQL3 (Suite) :

2.6 Routines définies par l'utilisateur (RDUs) :

- Les routines définies par l'utilisateur (RDU) définissent des méthodes de manipulation des données.
- Elle est définie :
  - ✓ En tant que parties d'un TDU.

ou

  - ✓ Séparément de tout TDU, en tant que partie d'un schéma.
- Elle est soit une procédure, soit une fonction, soit une méthode.
- Elle peut être soit fournie en externe dans un langage de programmation évolué, soit définie totalement en SQL.
- Elle est invoquée à l'aide de l'instruction SQL CALL.

SQL3 (Suite) :

2.6 Routines définies par l'utilisateur (RDUs) : (Suite)

- Elle prend zéro paramètre ou plus et les paramètres peuvent être individuellement d'entrée (IN), de sortie (OUT) ou d'entrée-sortie (INOUT).
- Si elle est complètement définie en SQL, elle possède un corps

Exemple :

```
CREATE FUNCTION médaillon(IN monImage ImageType) RETURNS
BOOLEAN
EXTERNAL NAME médaillon
LANGUAGE C
PARAMETER STYLE SQL
DETERMINISTIC
NO SQL;
```

SQL3 (Suite) :

2.7 Polymorphisme :

- Des routines différentes peuvent porter le même nom.
- Deux fonctions du même schéma ne peuvent pas porter la même signature.
- Deux fonctions du même schéma peuvent porter le même nom et posséder le même nombre de paramètres.
- SQL3 emprunte un modèle objet généralisé, de sorte que les types de tous les arguments d'une routine sont pris en considération lorsqu'il s'agit de déterminer la routine à invoquer, et de gauche à droite.

SQL3 (Suite) :

2.8 Modules de stockage persistant :

Un certain nombre de types d'instructions ont été ajoutés à SQL pour rendre le langage complet au niveau du calcul.

- Le comportement d'objet (les méthodes) puisse être stocké et exécuté au sein de la base de données sous forme d'instructions SQL.
- Regrouper les instructions dans une instruction composite (ou bloc), avec ses propres variables locales.

SQL3 (Suite) :

2.8 Modules de stockage persistant : (Suite)

Voici quelques-unes des instructions supplémentaires :

- Une instruction IF . . . THEN . . . ELSE . . . END IF permet le traitement conditionnel.
- Une instruction CASE autorise la sélection d'une voie d'exécution parmi une série d'alternatives.
- Un ensemble d'instructions permettent l'exécution répétitive d'un bloc d'instructions SQL. Les instructions itératives sont FOR, WHILE et REPEAT.
- L'instruction CALL permet d'appeler des procédures et une instruction RETURN permet de renvoyer le résultat.

SQL3 (Suite) :

2.9 Déclencheurs (triggers) :

Un déclencheur est une instruction (composite) en SQL qui s'exécute automatiquement au sein du SGBD par effet de bord.

- Les déclencheurs servent à un certain nombre de fins, dont :
  - ✓ La validation des données d'entrée et la conservation de contraintes d'intégrité complexes qui, sans eux, serait complexe, voire impossible, par le biais de contraintes de table;
  - ✓ La prise en compte d'alertes (par exemple par courrier électronique) concernant des actions à entreprendre lorsqu'une table subit une mise à jour d'une certaine forme;
  - ✓ L'entretien des informations d'audit, en enregistrant les changements effectués et par qui ils ont été apportés;
  - ✓ Le soutien à la duplication.

SQL3 (Suite) :

2.9 Déclencheurs (triggers) : (Suite)

Principal avantage :
Stocker des fonctions standard au sein de la base de données et les imposer de manière cohérente à chaque mise à jour.

```
CREATE TRIGGER NomDéclencheur
BEFORE | AFTER <événementDéclencheur>
ON <TableName>
[REFERENCING
<listeAliasAnciennesOuNouvellesValeurs>]
[FOR EACH {ROW | STATEMENT}]
[WHEN (conditionDéclenchement)]
<corpsDéclencheur>
```

Inconvénients :
- La complexité
- Le masquage des fonctionnalités.
- Les contraintes sur les performances.

SQL3 (Suite) :

2.10 Grands objets :

grand objet est un type de donnée capable de contenir une très grande quantité de données, par exemple un fichier de texte ou un graphisme.

- SQL3 définit trois types différents de grands objets :
  - ✓ Le grand objet binaire (BLOB, *Binary Large Object*), une chaîne binaire qui ne possède pas de jeu de caractères ni d'association de collationnement.
  - ✓ Le grand objet de caractères (CLOB, *Character Large Object*), le grand objet de caractères nationaux (NCLOB, *National Character Large Object*), tous deux des chaînes de caractères.

Exemple :

```
ALTER TABLE Personnel
ADD COLUMN résumé CLOB(50K);
ALTER TABLE Personnel
ADD COLUMN photo BLOB(12M);
```

3.Conclusion

Comparaison du partage des données des SGBDRO et SGBDOO.

Caractéristique	SGBDRO	SGBDOO
Transactions ACID	Support fort	Supportées
Récupération	Support fort	Supportée mais le niveau de soutien dépend du produit
Modèles de transactions évolués	Aucun support	Supportés mais le niveau de soutien dépend du produit
Sécurité, intégrité et vues	Support fort	Support réduit