



RAPPORT FINAL

SYSTÈMES D'INFORMATION DÉCISIONNELS ET ENTREPOTS DE DONNEES

PROFESSEURS RÉFÉRENTS : M. BERIO – M. DUBOIS – MME. RAPHALEN

ÉTUDIANTS : BOURNISIE ALICE – KEHR MANON – TANDAMBA EDOUARD

DATE DE RENDU : 07 MAI 2020

MASTER 1 – DATA SCIENCE ET MODÉLISATION STATISTIQUE

TABLE DES MATIÈRES

SUJET	3
DESCRIPTION DES DONNÉES	4
1. SCHÉMAS CONCEPTUELS.....	5
2. MESURES DE LA QUALITÉ DES DONNÉES.....	7
3. SCHÉMA INTÉGRÉ LOGIQUE	11
4. SCHÉMA DIMENSIONNEL	12
5. SCHÉMA INTÉGRÉ SOUS SQL SERVER.....	13
CRÉATION DU SCHÉMA ET DES TABLES	13
6. ALIMENTATION DE LA BASE DE DONNÉE.....	14
CRÉATION DES SCHÉMAS ET DES TABLES	14
INTÉGRATION DES DONNÉES	15
EXEMPLE DE TABLE IMPORTÉE : CUSTOMER DE METROSTARLET.....	15
NETTOYAGE DES DONNÉES	16
ALIMENTATION DE LA BASE CORRESPONDANT AU SCHÉMA INTÉGRÉ.....	17
TABLE GADGET DE MOREMOVIES.....	17
TABLE LOCATIONS DE MOREMOVIES.....	17
TABLE CLIENT DE MOREMOVIES	18
TABLE ACHAT DE MOREMOVIES	19
7. SCHÉMA DIMENSIONNEL SOUS SQL SERVER.....	20
EXEMPLE DE DIMENSION CRÉÉE : CLIENT	20
SCHÉMA EN ÉTOILE	21
8. DÉFINITION DE REQUÊTES.....	22
CE QU'ON AURAIT AIMÉ FAIRE	24
CONCLUSION	24
ANNEXES.....	25

SUJET

Le projet qui a été proposé, par nos professeurs, consiste à mettre en pratique les nouvelles connaissances que nous avons acquises durant leurs heures de cours et de travaux pratiques. Le cas pratique porte sur la mise en œuvre d'intégration de données sur le logiciel SQL Server.

Concrètement :

La PME MoreMovies à acquis 3 magasins de ventes/locations de films et de produits dérivés :

- MovieMegaMart
- BuckBoaster
- MetroStarlet

Chacun était auparavant géré individuellement et par des propriétaires différents.

Chaque client de chaque magasin est identifié par un code personnel enregistré sur une carte magnétique. Ils sont les seuls à pouvoir acheter/louer les produits dans le magasin où ils sont enregistrés.

Chaque produit est identifié électroniquement.

Chacun des 3 magasins à son propre système d'information. Les données sont donc enregistrées différemment en fonction du magasin. Il est donc impossible de pouvoir faire facilement des analyses statistiques fiables.

FINALITE DU PROJET : intégrer les données provenant des 3 magasins dans un système unique, c'est-à-dire créer un entrepôt de données. Cela permettra par la suite d'effectuer simplement des études des ventes/locations suivants les produits et les clients. De plus, cela permettra de réaliser des prévisions fiables : extension de la vente de gadgets à l'ensemble des magasins, nouvelle proposition de ventes.

BUT DU PROJET : réaliser les étapes initiales pour la construction d'un entrepôt de données pour le nouveau magasin MoreMovies.

MATERIEL : Les 3 sources de données, qui sont des bases de données relationnelles, sont disponibles. Ce sont des bases ACCESS et aucun document explicatif de ces dernières n'est à notre disposition.

LIVRAISON : Deux livraisons du projet sont demandées : une livraison intermédiaire à rendre pour fin mars et une livraison finale pour début mai.

Pour nous guider lors des différentes étapes de réalisation de ce projet, plusieurs questions nous ont été posées. Cependant, la première question nécessite, a priori, une bonne connaissance des données (composition des tables, variables disponibles, type de variable...). Pour cela, une étape de description des données a été essentielle.

DESCRIPTION DES DONNÉES

Avant de commencer à répondre aux questions proposées, nous avons procédé à une inspection des données minutieuse pour pouvoir se familiariser avec ses dernières. Pour cela, nous avons réalisé un fichier Excel contenant plusieurs pages. La première feuille de ce document, disponible en **ANNEXE 1**, donne, pour chaque fichier de données (1/magasin) :

- La dénomination des tables que le fichier contient
- Les différentes variables qui sont présentes dans les tables du fichier
- L'intitulé de la variable, c'est-à-dire « que représente la variable ? », ainsi que son type de données : quantitatives (nombre entier, à virgules, ou binaire), qualitatives (nombre de caractère), date (format de la date), etc.

Grâce à cette première étape, nous avons réussi à s'accoutumer des données et nous avons pu noter plusieurs remarques, telles que :

- Certaines modalités de certaines variables contiennent un « T » inutile au début de la modalité (ex : TCÉSAR, TDARRELL...).
- Les identifiants des clients et des films ne sont, comme prévu, pas les mêmes d'un magasin à l'autre (ex : CC100000, CC10000, CC100002 et 0, 1, 2).
- Certaines tables ne sont pas nécessaires à la réalisation de l'entrepôt de données, comme la table « MSysCompactError » du magasin BuckBoaster.
- Deux magasins ont une table « client » ou une table « film » et pour l'autre magasin, il n'y a pas de table concernant ces derniers, ces informations sont enregistrées dans une table qui regroupe les ventes ou locations.
- Certains prix sont « entiers » et d'autre « à virgules » (ex : 10 et 10.53)
- Certaines variables contiennent des modalités « vides ».
- La variable concernant le sexe n'est pas renseignée de la même manière entre les magasins (ex : M/F et -1/0).
- La table « moviecopy » du magasin MovieMegamart est complètement vide.
- Certaines variables ne sont pas disponibles d'un magasin à l'autre, comme « disposed » ou « is_new ».
- Deux magasins font de la vente et de la location et un autre ne fait que de la vente.
- Deux magasins vendent, en plus des films, des gadgets et un autre ne fait pas de vente de produits dérivés.
- Les dates ne sont pas toutes au même format, soit AAAA-MM-JJ ou JJ-MM-AAAA.

Une fois cette description réalisée, nous avons pu commencer à répondre aux questions qui nous ont été posées. Les différentes réponses, aux questions correspondantes, sont disponibles dans la suite de ce rapport.

1. SCHÉMAS CONCEPTUELS

La première partie de la question consiste à produire pour chaque source un schéma conceptuel, qui utilise un diagramme de classes UML. Nous avons réalisé ces derniers sous le logiciel VisualParadigm, nous avons ajouté un préfixe aux tables pour distinguer correctement les tables des différents magasins :

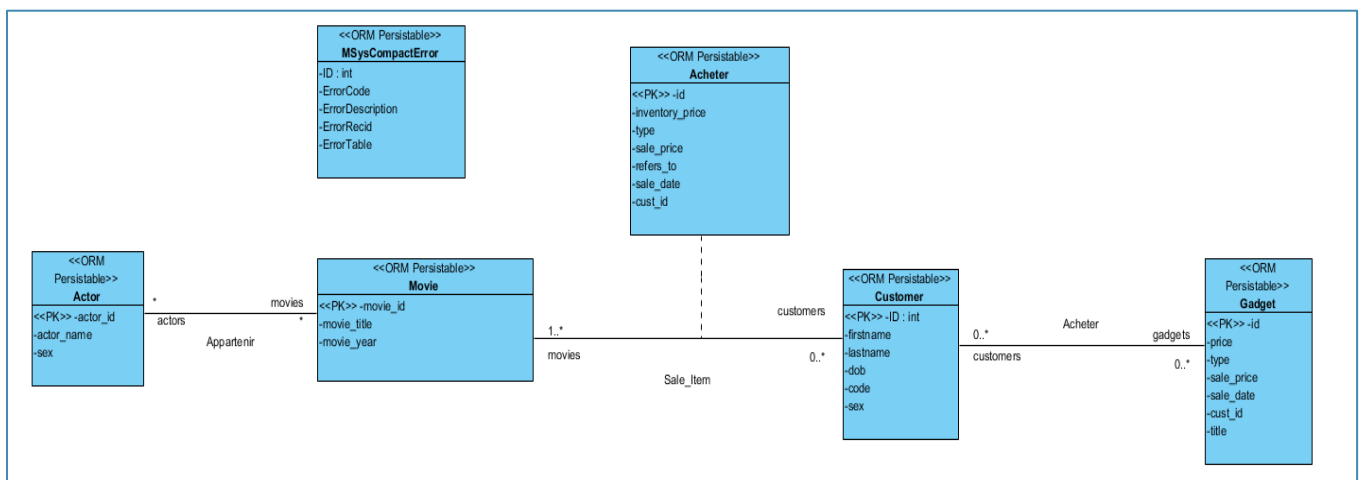


Figure 1 - MCD de la source BuckBooster

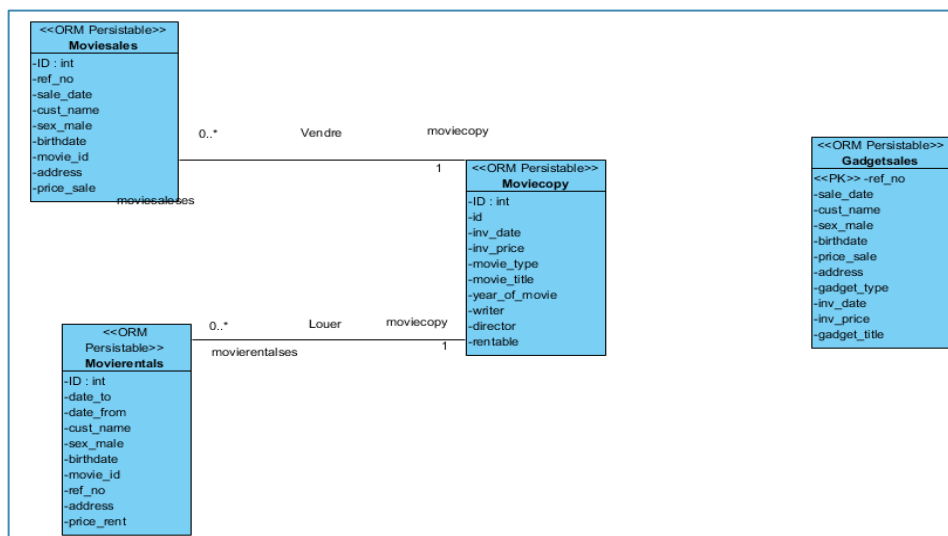


Figure 2 - MCD de la source MovieMegamart

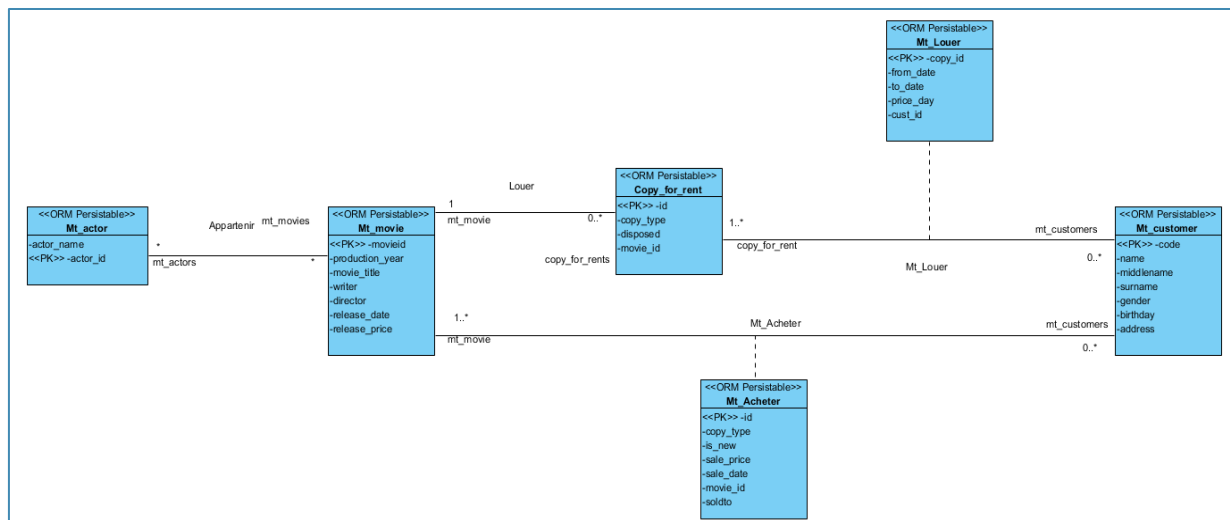


Figure 3 - MCD de la source MetroStarlet

Une fois que les trois schémas ont été conçus, un schéma conceptuel intégré a pu être réalisé, nous avons ajouté un suffixe « -i » pour distinguer les tables du schéma conceptuel intégré des autres schémas conceptuels.

Avant la réalisation graphique, nous avons créé un fichier Excel pour se mettre d'accord sur les tables essentielles au schéma intégré, ainsi qu'aux variables qui seront présentes dans chacune d'entre elle. En **ANNEXE 2**, se trouve un aperçu de ce fichier. Avec l'aide des correspondances inter-schéma, nous avons donc créé de nouvelles tables, ou bien rassemblé plusieurs tables. Nous avons au final 7 tables : customer, gadget, movie, sales, rentals et MySysCompactError (table qui ne servira pas par la suite). Nous avons donc pris différentes décisions telles que :

- Création de tables qui distinguent : les films (*movie*), les gadgets (*gadget*), les clients (*customer*), les ventes (*sales*), les locations (*rentals*) et les acteurs (*actors*).
- Rassemblement des variables identiques entre chaque tables (ex : price_sale/sale_price/price devient une unique variable : sale_price ou encore movie_id/id/movieid devient une unique variable : movie_id).
- Création d'un codage unique pour les identifiants
- Création de variables (ex : sale_type : cette variable de la table « sales » informera si c'est un gadget ou un film qui a été vendu).
- Suppression de certaines variables non nécessaires (middle_name, is_new, disposed, etc.)

Une fois que ces changements et créations ont été faits, le schéma conceptuel intégré a pu être réalisé :

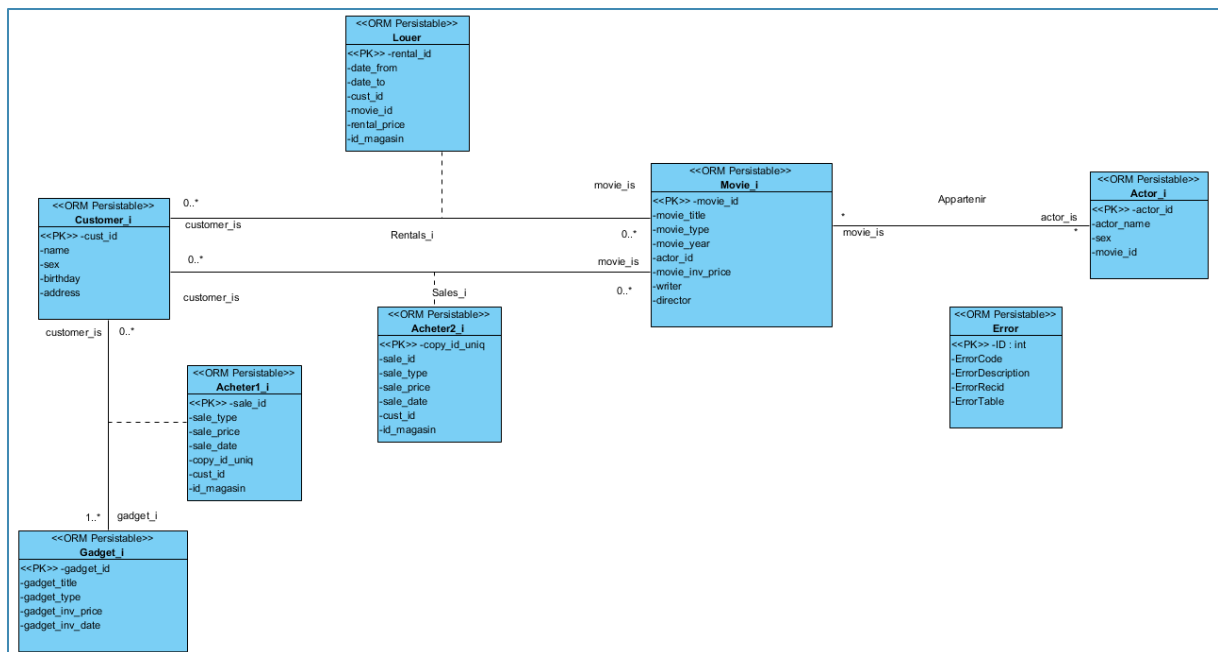


Figure 4 - MCD de la fusion des sources

2. MESURES DE LA QUALITÉ DES DONNÉES

La seconde question porte sur la qualité des données. Il est essentiel de réaliser cette étape pour prendre conscience des modifications ou solutions qui seront à apporter pour une intégration dans un entrepôt de données conforme. Les mesures que nous avons souhaité calculer sont les résultats des sorties du logiciel SAS. Nous avons programmé afin de trouver les informations que nous avons jugé nécessaires. Nous avons utilisé en particulier la PROC SQL qui est disponible sous SAS. Nous avons également utilisé le logiciel Excel pour confirmer nos résultats. Un aperçu du programme que nous avons rédigé est disponible en **ANNEXE 3**.

- **Les valeurs manquantes** : ce sont les valeurs manquantes pour chaque variable.
- **Les doublons** : ce sont le nombre de lignes ou d'observations qui se répètent au moins deux fois. ATTENTION : Dans quelques tables, les doublons n'ont pas été recensés par le logiciel, car ils ont un identifiant différent, mais il s'agit bel et bien de la même observation. Cela sera spécifié en dessous de la table.
- **Le totale des observations** : c'est le nombre total des enregistrements ou de ligne de chaque table.
- Nous n'avons pas trouvé d'inconsistances, c'est-à-dire aucune incohérence dans les données

BUCKBOASTER

TABLE ACTOR	actor_name	actor_id	sex
Valeurs manquantes	0	0	1000 100%
Doublons	0		
Total observations	1000		

Dans la table « actor », aucun doublon n'a été repéré par le logiciel SAS, mais en réalité il y a 14 doublons, c'est-à-dire au moins 14 acteurs qui apparaissent 2 fois ou plus. Cependant, nous ne pouvons pas affirmer qu'il s'agit de véritables doublons, il s'agit peut-être d'homonymes. Nous l'avons trouvé grâce à la mise en forme conditionnelle sur le logiciel Excel. Nous avons sélectionné toutes les lignes apparaissant en double, et cela, sans prendre en compte l'identifiant.

TABLE MOVIE	movie_title	movie_id	movie_year
Valeurs manquantes	510 33,77%	510 33,77%	510 33,77%
Doublons	510		
Total observations	1510		

Les doublons qui apparaissent dans ce tableau sont en réalité 510 enregistrements « vides », le logiciel les considère comme des enregistrements en plusieurs fois, mais nous savons que ce n'est pas réellement des doublons mais plutôt des valeurs manquantes. Grâce à Excel, nous avons repéré que 2 films apparaissent plusieurs fois dans la table, ils n'ont pas le même identifiant, mais exactement les mêmes valeurs pour les différentes variables, il s'agit donc sûrement de doublons. De plus, 8 films ont le même titre, mais pas les mêmes valeurs pour les différentes variables, il ne s'agit donc peut-être pas de véritables doublons.

TABLE CUSTOMER	firstname	lastname	dob	code	sex
Valeurs manquantes	0	0	0	0	0
Doublons	0				
Total observations	3756				

Dans la table « customer », aucun doublon n'a été repéré par le logiciel, mais en réalité il y a 379 doublons. Nous les avons trouvés grâce à la mise en forme conditionnelle sur le logiciel Excel. Nous avons sélectionné toutes les lignes apparaissant en double, et cela, sans prendre en compte l'identifiant.

TABLE ACTSIN	movieid	actorid
Valeurs manquantes	0	0
Doublons	6	
Total observations	1000	

TABLE GADGET	price	type	sale_price	id	sale_date	cust_id	title
Valeurs manquantes	0	0	0	0	0	0	0
Doublons	0						
Total observations	5656						

3. SCHÉMA INTÉGRÉ LOGIQUE

À partir du schéma conceptuel intégré conçu à la question 1, un schéma intégré logique a été produit.

Le MCD¹ permet de :

- Modéliser la **sémantique** des informations de la future base de données
- Utilise le graphisme Entité-Relation
- **Ne permet pas** d'implémenter la BDD dans un SGBD

Tandis que le MLD² permet de

- Modéliser la **structure** selon laquelle les données seront stockées dans la future base de données
- Utilise le formalisme graphique Merise
- **Permet** l'implémentation de la BDD dans un SGBD donné

Ce modèle permet par la suite la création des tables de la base de données en langage SQL. Le code SQL est fourni automatiquement par le logiciel VisualParadigm.

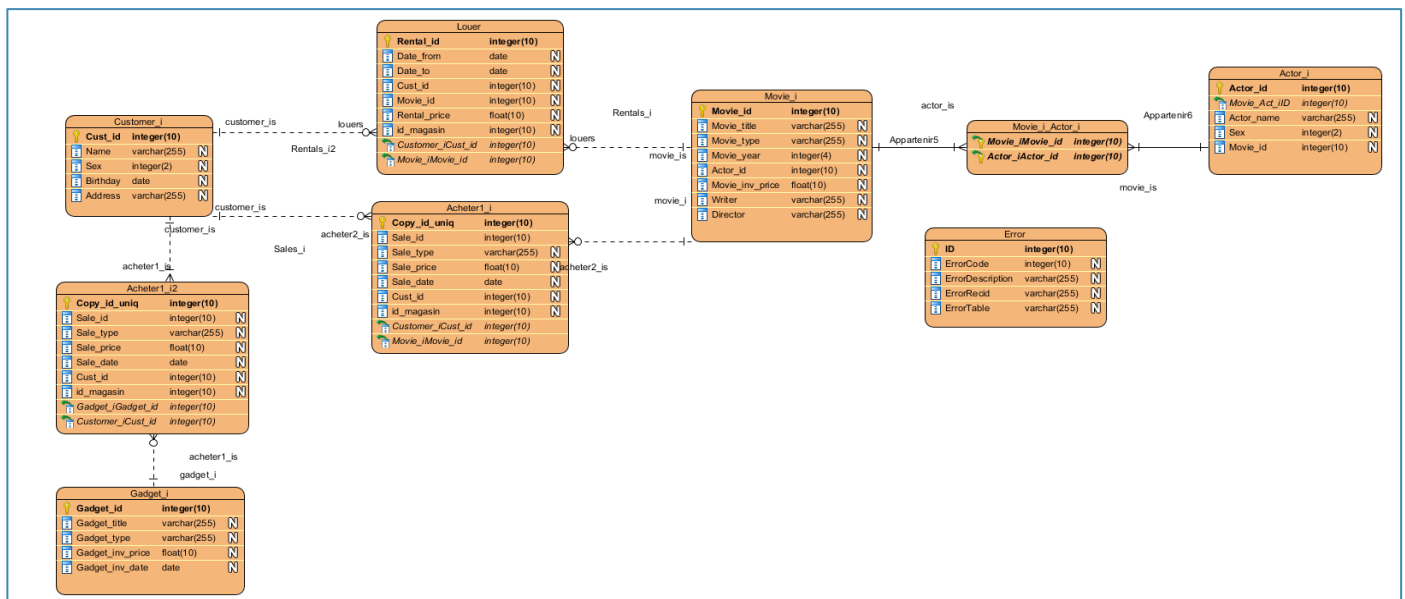


Figure 5 - MLD de la fusion des sources

¹ Modèle Conceptuel des Données

² Modèle Logique des Données

4. SCHÉMA DIMENSIONNEL

La quatrième question consiste à proposer un schéma dimensionnel constitué de dimensions et d'une table des faits.

Les dimensions décrivent les entités d'entreprise en intégrant la notion de temps. Ici, ce sont :

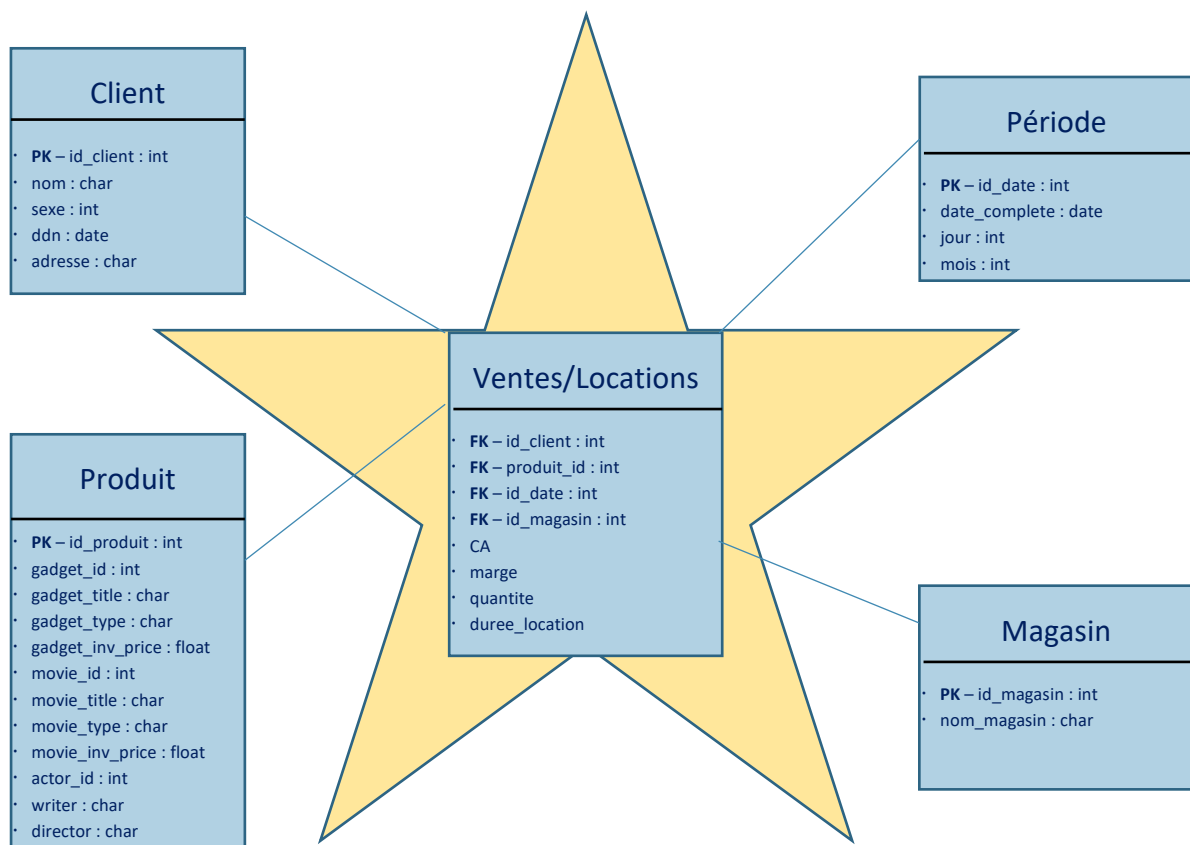
- Produit
- Magasin
- Période
- Client

La table de faits, qui est décrite par les dimensions précédentes, stockent des observations/événements et les mesures numériques suivantes :

- Chiffre d'affaires
- Marge
- Quantité vendue
- Durée location

Les attributs de la table de faits correspondent aux clés des différentes dimensions : la date pour la dimension période, idClient pour la dimension client, id_produit pour la table produit, id_magasin pour la dimension magasin.

Il s'agit d'un schéma en étoile : chaque table de faits représente un processus métier, et chacune de ces tables est reliée à des dimensions,



5. SCHÉMA INTÉGRÉ SOUS SQL SERVER















CRÉATION DU SCHÉMA ET DES TABLES

Cette question a pour but de produire le schéma intégré logique, cette fois-ci sous SQL Server. Le code nécessaire à cette création est disponible ci-dessous. Tout d'abord, il faut créer le schéma :

```
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'MoreMovies')
BEGIN
EXEC ('CREATE SCHEMA MoreMovies;');
END;
```

Ensuite, on affecte à ce schéma les différentes tables que nous avons définies dans la première partie de ce projet. Nous avons apporté quelques modifications, notamment par rapport au type de données, ou alors l'ajout de variable qui nous a semblé nécessaire, comme un identifiant de magasin ou encore le type de l'article (si c'est un film ou un gadget). On a donc le code ci-dessous, ainsi que l'apparition des tables dans SQL Server :

```
-- Actor
CREATE TABLE MoreMovies.Acteurs(
actor_id INT IDENTITY (1, 1),
actor_name varchar(255));
-- Movie
CREATE TABLE MoreMovies.Films(
movie_id INT IDENTITY (1, 1),
movie_title varchar(255),
movie_type varchar(255),
movie_year INT,
actor_id INT,
movie_inv_price FLOAT,
writer varchar(255),
director varchar(255));
-- Location
CREATE TABLE MoreMovies.Locations(
rental_id INT IDENTITY (1, 1),
date_from DATE,
date_to DATE,
cust_id INT,
movie_id INT,
rental_price FLOAT,
id_magasin VARCHAR(255));
-- Achat
CREATE TABLE MoreMovies.Achats(
copy_id_uniq INT IDENTITY (1, 1),
sales_type varchar(255),
sales_price FLOAT,
sales_date DATE,
cust_id varchar(255),
id_magasin varchar(255),
type_article varchar(255));
-- Customer
CREATE TABLE MoreMovies.Clients(
cust_id INT IDENTITY (1, 1),
name varchar(255),
sex varchar(255),
age INT,
adress varchar(255));
-- Gadget
CREATE TABLE MoreMovies.Gadget(
gadget_id INT IDENTITY (1, 1),
gadget_tile varchar(255),
gadget_type varchar(255),
gadget_sale_price FLOAT,
gadget_inv_price FLOAT,
gadget_inv_date DATE);
```

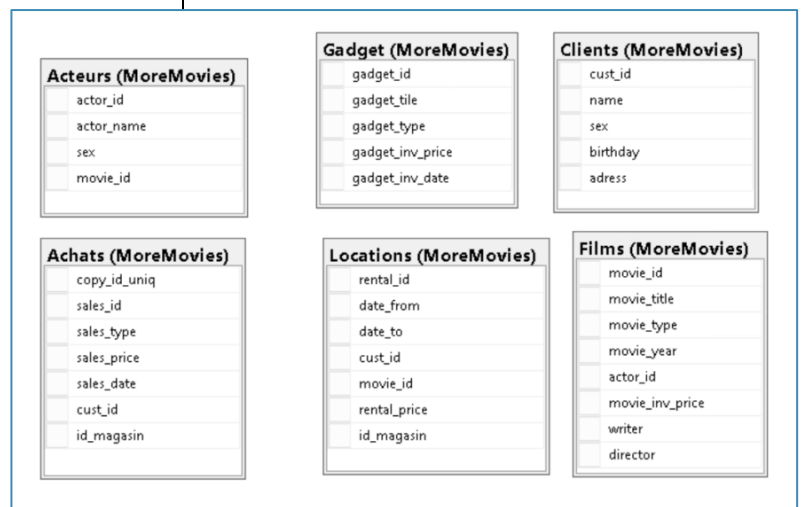
		MoreMovies.Achats
		MoreMovies.Acteurs
		MoreMovies.Actor
		MoreMovies.Clients
		MoreMovies.Films
		MoreMovies.Gadget
		MoreMovies.Locations

6. ALIMENTATION DE LA BASE DE DONNÉE

CRÉATION DES SCHÉMAS ET DES TABLES

Tout d'abord, avant d'alimenter la base de données correspondant au schéma intégré, il faut créer les différents schémas, c'est-à-dire un par magasins, c'est-à-dire 3 schémas distincts. Ces schémas contiennent les différentes tables que nous avons à notre disposition. Le code pour la création des trois schémas, et les tables qui les composent est disponible ci-dessous. Seule la source MovieMegamart est présentée ici pour ne pas alourdir la présentation. Le reste du code est disponible en **ANNEXE 4**.

```
-- la source Moviemegamart a son schéma qui hebergera tables, vues,...
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'Moviemegamart')
BEGIN
EXEC ('CREATE SCHEMA Moviemegamart;');
END;
--- Création des tables pour MovieMegamart:
CREATE TABLE Moviemegamart.gadgetsales(
ref_no INT ,
sale_date DATE, -- format 27/12/2010
cust_name varchar(255), -- a enlever le T
sex_male INT,
birthdate DATE, -- format 27/12/2010
price_sale FLOAT,
address varchar(255), -- a enlever le T
gadget_type varchar(255), -- a enlever le T
inv_date DATE, -- format 27/12/2010
inv_price FLOAT,
gadget_title varchar(255) -- a enlever le T
);
CREATE TABLE Moviemegamart.moviecopy(
id INT, --vide
inv_date DATE, --vide
inv_price INT, --vide
movie_type VARCHAR(255), --vide
movie_title varchar(255), --vide
year_of_movie DATE, --vide JUSTE ANNEE
writer varchar(255), --vide
director varchar(255), --vide
rentable varchar(255) --vide
);
CREATE TABLE Moviemegamart.movierentals(
date_to DATE, --format 26/12/2010
date_from DATE, --format 26/12/2010
cust_name varchar(255), -- a enlever le T
sex_male INT,
birthdate DATE, --format 26/12/2010
movie_id INT,
ref_no INT ,
address varchar(255), -- a enlever le T
price_rent FLOAT
);
CREATE TABLE Moviemegamart.moviesales(
sale_date DATE, --format 26/12/2010
cust_name varchar(255), -- a enlever le T
sex_male INT,
birthdate DATE, --format 26/12/2010
movie_id INT,
ref_no INT,
address varchar(255), -- a enlever le T
price_sale FLOAT);
```



INTÉGRATION DES DONNÉES

Une fois les schémas et tables créés, nous avons pu passer, à partir des données mémorisées dans les sources, aux différentes intégrations de données, en procédant de la même manière pour chaque source et chaque table :



Le mapping est très important pour que l'importation des données se passe correctement. Cela consiste à définir la correspondance entre les deux modèles de données, à la fois au niveau du type de données (une variable de type caractère ne doit pas correspondre à une variable de type numérique, et inversement), le nombre de variables doit coïncider entre les différentes tables, et les noms doivent également être les mêmes pour faciliter cette étape.

EXEMPLE DE TABLE IMPORTÉE : CUSTOMER DE METROSTARLET

	name	middlename	surname	gender	code	birthday	address
1	TAARON	TA.	TLYCHWALA	M	745	T1946-01-28	TVan Road 271 New York
2	TAARON	TA.	TRIETH	M	3121	T1980-12-28	TNewkirk Square 453 New York
3	TAARON	TY.	TRODE	M	3691	T1944-04-17	TLynch Square 456 New York
4	TABE	TA.	TDOUGHTRY	M	1137	T1977-01-26	TMetropolitan Alley 311 New York
5	TABE	TP.	TTRUCKS	M	1139	T1939-03-18	TFerry Lane 156 New York
6	TABEL	TM.	TGRANADOS	M	148	T1965-08-30	TChurch Lane 30 New York
7	TABEL	TN.	TKIRTON	M	119	T1963-08-25	TKansas Square 460 New York
8	TABEL	TP.	TBARTER	M	1646	T1961-04-21	TDwight Avenue 82 New York
9	TABIGAIL	TN.	THAGENY	F	3308	T1969-02-12	TMorgan Alley 494 New York
10	TABRAHAM	TP.	TBLAKSTAD	M	227	T1937-04-06	TStockton Lane 277 New York

NETTOYAGE DES DONNÉES

Une fois les données intégrées, le nettoyage des tables est nécessaire et inévitable. Le code qui nous a permis de réaliser cette étape est disponible ci-dessous. Un aperçu avant après est disponible à la suite de ce code. Pour le nettoyage, voici quelques modifications et/ou améliorations que nous avons apportées aux données. Comme précédemment, seul un des magasins est présenté, ici MetroStarlet, pour ne pas alourdir la présentation. Le reste du code est disponible en **ANNEXE 5**.

- Suppression, s'il y a, des lignes vides.
- Suppression du caractère « T » en début de modalité de certaines variables.
- Création de variables, comme l'âge, quand il n'était pas disponible.
- Mise en forme des dates.
- Suppression des doublons.

```
DELETE FROM Metrostarlet.actor WHERE actor_name is null and actor_id is null ;
SELECT SUBSTRING(actor_name, 2, 255) AS actor_name, actor_id INTO metrostarlet.actor2 FROM metrostarlet.actor
--table acts_in
DELETE FROM Metrostarlet.acts_in WHERE movie_id is null and actor_id is null ;
--table copy_for_rent -> table copy_for_rent 2
DELETE FROM Metrostarlet.copy_for_rent WHERE movie_id is null and id is null and disposed is null and copy_type is null ;
SELECT SUBSTRING(copy_type, 2, 255) AS copy_type, disposed, id, movie_id INTO Metrostarlet.copy_for_rent2 FROM
Metrostarlet.copy_for_rent
--table copy_for_sale -> table copy_for_sale 2
DELETE FROM Metrostarlet.copy_for_sale WHERE copy_type is null and is_new is null and sale_price is null and id is null and sale_date
is null and movie_id is null and soldto is null;
SELECT SUBSTRING(copy_type, 2, 255) AS copy_type, is_new, sale_price, id, sale_date, movie_id, soldto INTO
metrostarlet.copy_for_sale2 FROM metrostarlet.copy_for_sale
--table copy_rented_to
DELETE FROM Metrostarlet.copy_rented_to WHERE from_date is null and to_date is null and price_day is null and copy_id is null and
cust_id is null ;
--table customer -> table customer 3
DELETE FROM Metrostarlet.customer WHERE name is null and middlename is null and surname is null and gender is null and code is
null and birthday is null and address is null ;
SELECT SUBSTRING(name, 2, 255) AS name, SUBSTRING(middlename, 2, 255) AS middlename, SUBSTRING(surname, 2, 255) AS surname,
gender, code, SUBSTRING(birthday, 2, 255) AS birthday, SUBSTRING(address, 2, 255) AS address INTO metrostarlet.customer2 FROM
metrostarlet.customer
SELECT *, DATEDIFF(YEAR, birthday, GETDATE()) AS age INTO metrostarlet.customer3 --création de l'âge
FROM metrostarlet.customer2;
--table movie -> table movie 2
DELETE FROM Metrostarlet.movie WHERE production_year is null and movie_title is null and movieid is null and writer is null and
director is null and release_date is null and release_price is null ;
SELECT production_year, SUBSTRING(movie_title, 2, 255) AS movie_title, movieid, SUBSTRING(writer, 2, 255) AS writer,
SUBSTRING(director, 2, 255) AS director, release_date, release_price INTO metrostarlet.movie2 FROM metrostarlet.movie
```

AVANT

	name	middlename	surname	gender	code	birthday	address
1	TAARON	TA.	TLYCHWALA	M	745	T1946-01-28	TVan Road 271 New York
2	TAARON	TA.	TRIETH	M	3121	T1980-12-28	TNewkirk Square 453 New York
3	TAARON	TY.	TRODE	M	3691	T1944-04-17	TLynch Square 456 New York
4	TABE	TA.	TDOUGHTRY	M	1137	T1977-01-26	TMetropolitan Alley 311 New York
5	TABE	TP.	TTRUCKS	M	1139	T1939-03-18	TFerry Lane 156 New York
6	TABEL	TM.	TGRANADOS	M	148	T1965-08-30	TChurch Lane 30 New York
7	TABEL	TN.	TKIRTON	M	119	T1963-08-25	TKansas Square 460 New York
8	TABEL	TP.	TBARTER	M	1646	T1961-04-21	TDwight Avenue 82 New York
9	TABIGAIL	TN.	THAGENY	F	3308	T1969-02-12	TMorgan Alley 494 New York
10	TABRAHAM	TP.	TBLAKSTAD	M	227	T1937-04-06	TStockton Lane 277 New York

APRÈS

	name	middlename	surname	gender	code	birthday	address	age
1	AARON	A.	LYCHWALA	M	745	1946-01-28	Van Road 271 New York	74
2	AARON	A.	RIETH	M	3121	1980-12-28	Newkirk Square 453 New York	40
3	AARON	Y.	RODE	M	3691	1944-04-17	Lynch Square 456 New York	76
4	ABE	A.	DOUGHTRY	M	1137	1977-01-26	Metropolitan Alley 311 New York	43
5	ABE	P.	TRUCKS	M	1139	1939-03-18	Ferry Lane 156 New York	81
6	ABEL	M.	GRANADOS	M	148	1965-08-30	Church Lane 30 New York	55
7	ABEL	N.	KIRTON	M	119	1963-08-25	Kansas Square 460 New York	57
8	ABEL	P.	BARTER	M	1646	1961-04-21	Dwight Avenue 82 New York	59
9	ABIGAIL	N.	HAGENY	F	3308	1969-02-12	Morgan Alley 494 New York	51
10	ABRA...	P.	BLAKSTAD	M	227	1937-04-06	Stockton Lane 277 New York	83

ALIMENTATION DE LA BASE CORRESPONDANT AU SCHÉMA INTÉGRÉ

Les données étant intégrées dans leur schéma et tables correspondantes, ainsi que nettoyées, nous pouvons les intégrer dans le schéma et donc les tables du Magasin MoreMovies. Pour cela, nous avons écrit et générer le code ci-dessous (seules les tables gadget, achat, locations et client sont présentées ici, **ANNEXE 6**) :

TABLE GADGET DE MOREMOVIES

```
-- Table Gadget
-- Integration
INSERT INTO MoreMovies.Gadget (gadget_inv_date, gadget_inv_price, gadget_sale_price,
gadget_tile, gadget_type)
SELECT sale_date, price, sale_price, title, type
FROM Buckboaster.gadget2
UNION
SELECT sale_date, inv_price, price_sale, gadget_title, gadget_type
FROM Moviemegamart.gadgetsales2
ORDER BY 1;
```

MoreMovies.Gadget	
Columns	
gadget_id	(int, not null)
gadget_tile	(varchar(255), null)
gadget_type	(varchar(255), null)
gadget_sale_price	(float, null)
gadget_inv_price	(float, null)
gadget_inv_date	(date, null)

	gadget_id	gadget_tile	gadget_type	gadget_sale_price	gadget_inv_price	gadget_inv_date
1	1	Alarm Clock	Clock	176,27785	123,39449	2004-01-06
2	2	Baseball Cap	Hat	30,316418	21,221493	2004-01-08
3	3	Baseball Cap	Hat	129	90	2004-01-12
4	4	Alarm Clock	Clock	248	173	2004-01-13
5	5	Black TShirt	TShirt	135,94113	95,15879	2004-01-19
6	6	Medium Mug	Mug	142,41714	99,692	2004-01-19
7	7	LED Alarm Clock	Clock	102	71	2004-01-20
8	8	Pink TShirt	TShirt	249	174	2004-01-20
9	9	LED Alarm Clock	Clock	101	71	2004-01-21
10	10	Ice Cap	Hat	128,68436	90,079056	2004-01-21

TABLE LOCATIONS DE MOREMOVIES

```
-- table Location
-- Création d'un identifiant magasin
ALTER TABLE Metrostarlet.copy_for_rent2
ADD nom_magasin varchar(255);
UPDATE Metrostarlet.copy_for_rent2
SET nom_magasin = 'Metrostarlet';
ALTER TABLE Metrostarlet.copy_rented_to
ADD nom_magasin varchar(255);
UPDATE Metrostarlet.copy_rented_to
SET nom_magasin = 'Metrostarlet';
ALTER TABLE Moviemegamart.movierentals2
ADD nom_magasin varchar(255);
UPDATE Moviemegamart.movierentals2
SET nom_magasin = 'Moviemegamart';
-- Integration
INSERT INTO MoreMovies.Locations(date_from, date_to, cust_id, movie_id, rental_price, id_magasin)
SELECT date_from, date_to, NULL, movie_id, price_rent, nom_magasin
FROM Moviemegamart.movierentals2
UNION ALL
SELECT from_date, to_date, cust_id, NULL, price_day, nom_magasin
FROM Metrostarlet;
```

	rental_id	date_from	date_to	movie_id	rental_price	id_magasin
1	1	2005-01-01	2005-01-05	60581	17,35	Moviemegamart
2	2	2005-01-02	2005-01-05	35222	15,31	Moviemegamart
3	3	2005-01-02	2005-01-05	60581	17,35	Moviemegamart
4	4	2005-01-03	2005-01-06	35222	15,31	Moviemegamart
5	5	2005-01-03	2005-01-05	60581	17,35	Moviemegamart
6	6	2005-01-03	2005-01-05	70522	16,33	Moviemegamart
7	7	2005-01-03	2005-01-06	35222	15,31	Moviemegamart
8	8	2005-01-04	2005-01-06	38134	20,41	Moviemegamart
9	9	2005-01-04	2005-01-07	35222	15,31	Moviemegamart

TABLE CLIENT DE MOREMOVIES

Pour cette table, plusieurs améliorations ont été apportées comme :

- Concaténation quand il le fallait du nom et du prénom du client
- Uniformisation du sexe en « M » et « F »
- Mise en majuscule des noms pour uniformiser le visuel
- Ajout d'une colonne « AGE »

```
-- Table Customer
-- Création d'une nouvelle colonne pour avoir un format unique en caractère
ALTER TABLE Moviemegamart.gadgetsales2 ADD sex varchar(255);
ALTER TABLE Moviemegamart.movierentals2 ADD sex varchar(255);
ALTER TABLE Moviemegamart.moviesales2 ADD sex varchar(255);
-- Transformation en caractere
UPDATE Moviemegamart.gadgetsales2 SET sex = CASE WHEN sex_male = -1
THEN 'M' ELSE 'F' END FROM Moviemegamart.gadgetsales2;
UPDATE Moviemegamart.movierentals2 SET sex = CASE WHEN sex_male = -1
THEN 'M' ELSE 'F' END FROM Moviemegamart.movierentals2;
UPDATE Moviemegamart.moviesales2 SET sex = CASE WHEN sex_male = -1
THEN 'M' ELSE 'F' END FROM Moviemegamart.moviesales2;
-- Integration
INSERT INTO MoreMovies.Clients(name, sex, age, adress)
SELECT CONCAT(firstname, ' ', lastname), sex, age, NULL
FROM Buckboaster.customer2
UNION
SELECT CONCAT(surname, ' ', name), gender, DATEDIFF(YEAR, birthday, GETDATE()), address
FROM Metrostarlet.customer2
UNION
SELECT cust_name, sex, age, address
FROM Moviemegamart.gadgetsales2
UNION
SELECT cust_name, sex, age, address
FROM Moviemegamart.moviesales2
UNION
SELECT cust_name, sex, age, address
FROM Moviemegamart.movierentals2
ORDER BY 1;
-- On met tous les sexes au même format
UPDATE MoreMovies.Clients SET sex = CASE WHEN sex = 'Female' THEN 'F'
WHEN sex = 'F' THEN 'F'
WHEN sex = 'M' THEN 'M'
WHEN sex = 'Male' THEN 'M'
END FROM MoreMovies.Clients;
-- on met tout en MAJ pour uniformiser
UPDATE MoreMovies.Clients SET name = UPPER(name) FROM MoreMovies.Clients;
```

	cust_id	name	sex	age	adress
1	1	AAKRE MONICA	F	31	Union Street 339 New York
2	2	AALDERINK CAMMY	F	42	Woodhull Lane 265 New York
3	3	AARON CURETON	M	70	NULL
4	4	AARON LYSHWALA	M	74	NULL
5	5	AASE BOBBIE	M	56	Bay Square 158 New York
6	6	ABBEY N. TREMEL	F	47	College Alley 139 New York
7	7	ABBS ELIZEBETH	F	50	Paerdegat Lane 341 New York
8	8	ABDUL ARDOIN	M	39	NULL
9	9	ABDUL GAVE	M	82	NULL
10	10	ABDUL N. GLADFELTER	M	82	Albemarle Road 374 New York

TABLE ACHAT DE MOREMOVIES

Pour cette table, plusieurs améliorations ont été apportées comme :

- Ajout d'un identifiant magasin qui permet de savoir qui a réalisé la vente
- Ajout d'une variable qui indique si c'est un gadget ou un film

```
-- table Achat
-- Création d'un identifiant magasin
ALTER TABLE Buckbooster.sale_item2 ADD nom_magasin varchar(255);
UPDATE Buckbooster.sale_item2 SET nom_magasin = 'Buckbooster';
ALTER TABLE Buckbooster.gadget2 ADD nom_magasin varchar(255);
UPDATE Buckbooster.gadget2 SET nom_magasin = 'Buckbooster';
ALTER TABLE Metrostarlet.copy_for_sale2 ADD nom_magasin varchar(255);
UPDATE Metrostarlet.copy_for_sale2 SET nom_magasin = 'Metrostarlet';
ALTER TABLE Moviemegamart.gadgetsales2 ADD nom_magasin varchar(255);
UPDATE Moviemegamart.gadgetsales2 SET nom_magasin = 'Moviemegamart';
ALTER TABLE Moviemegamart.moviesales2 ADD nom_magasin varchar(255);
UPDATE Moviemegamart.moviesales2 SET nom_magasin = 'Moviemegamart';
-- Création d'une variable qui indique si c'est un film ou un gadget
ALTER TABLE Buckbooster.sale_item2 ADD type_article varchar(255);
UPDATE Buckbooster.sale_item2 SET type_article = 'FILM';
ALTER TABLE Buckbooster.gadget2 ADD type_article varchar(255);
UPDATE Buckbooster.gadget2 SET type_article = 'GADGET';
ALTER TABLE Metrostarlet.copy_for_sale2 ADD type_article varchar(255);
UPDATE Metrostarlet.copy_for_sale2 SET type_article = 'FILM';
ALTER TABLE Moviemegamart.gadgetsales2 ADD type_article varchar(255);
UPDATE Moviemegamart.gadgetsales2 SET type_article = 'GADGET';
ALTER TABLE Moviemegamart.moviesales2 ADD type_article varchar(255);
UPDATE Moviemegamart.moviesales2 SET type_article = 'FILM';
-- Integration
INSERT INTO MoreMovies.Achats(sales_type, sales_price, sales_date, cust_id, id_magasin, type_article)
SELECT type, sale_price, sale_date, cust_id, nom_magasin, type_article
FROM Buckbooster.sale_item2
UNION ALL
SELECT type, sale_price, sale_date, cust_id, nom_magasin, type_article
FROM Buckbooster.gadget2
UNION ALL
SELECT copy_type, sale_price, sale_date, NULL, nom_magasin, type_article
FROM Metrostarlet.copy_for_sale2
UNION ALL
SELECT gadget_type, price_sale, sale_date, NULL, nom_magasin, type_article
FROM Moviemegamart.gadgetsales2
UNION ALL
SELECT NULL, price_sale, sale_date, NULL, nom_magasin, type_article
FROM Moviemegamart.moviesales2
ORDER BY 1;
```

copy_id_uniq	sales_type	sales_price	sales_date	cust_id	id_magasin	type_article
60061	TShirt	138	2009-06-09	CC10...	Buckbooster	GADGET
60062	TShirt	138,06248	2009-12-07	NULL	Moviemegamart	GADGET
60063	TShirt	138,5195	2009-09-11	NULL	Moviemegamart	GADGET
60064	TShirt	138,53369	2010-09-03	NULL	Moviemegamart	GADGET
60065	TShirt	138,54001	2008-10-01	NULL	Moviemegamart	GADGET
60066	TShirt	138,9035	2009-01-16	NULL	Moviemegamart	GADGET
60067	TShirt	139	2005-07-27	CC10...	Buckbooster	GADGET
60068	TShirt	139	2008-01-29	CC10...	Buckbooster	GADGET
60069	TShirt	139	2008-05-15	CC10...	Buckbooster	GADGET

7. SCHÉMA DIMENSIONNEL SOUS SQL SERVER

Dans la première partie, nous avons déjà réalisé un schéma dimensionnel sous Visual Paradigm. Cette fois-ci, nous allons le générer sous SQL Server. Pour cela, nous avons d'abord créé un schéma, puis nous avons créé les tables de dimensions et la table de fait. Ensuite, nous avons indiqué les dépendances et interactions entre les dimensions et la table de faits. Enfin, nous avons généré le schéma dimensionnel correspondant. Le code pour la création des composants du schéma dimensionnel est disponible ci-dessous, tandis que la démarche pour générer ce dernier est disponible juste après.

SHEMA	DIMENSIONS	FAITS	RELATIONS
<pre> ----- création du schéma pour les dimensions IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'DIM') BEGIN EXEC ('CREATE SCHEMA DIM;'); END; GO </pre>	<pre> ----- Création des tables dimensions --Produit Create table DIM.Produit(id_produit int primary key identity, gadget_title varchar(50), gadget_type varchar(50), gadget_inv_price varchar(20), movie_title varchar(50), movie_type varchar(50), movie_inv_price varchar(50), actor_id int not null, writer varchar(50), director varchar(50)) go --Magasin Create table DIM.Magasin(id_magasin int primary key identity, nom_magasin varchar(50))go --Client Create table DIM.Client(id_client int primary key identity, nom varchar(50), sexe varchar(10),ddn date, adresse varchar(50))go --Periode-- Create table DIM.Periode(id_date int primary key identity, date_complete date, jour int, mois int)go </pre>	<pre> -- Création de la table de fait Ventes/Locations Create Table DIM.Ventes_Locations(Ventes_Locations_id int primary key identity, id_client int not null, id_produit int not null, id_magasin int not null, id_date int not null, CA float, marge float, quantite float, duree_location float) Go </pre>	<pre> ----- Création des relations entre les dimensions et la table de fait ALTER TABLE DIM.Ventes_Locations ADD CONSTRAINT FK_id_client FOREIGN KEY (id_client)REFERENCES DIM.Client(id_client); ALTER TABLE DIM.Ventes_Locations ADD CONSTRAINT FK_id_produit FOREIGN KEY (id_produit)REFERENCES DIM.Produit(id_produit); ALTER TABLE DIM.Ventes_Locations ADD CONSTRAINT FK_id_magasin FOREIGN KEY (id_magasin)REFERENCES DIM.Magasin(id_magasin); ALTER TABLE DIM.Ventes_Locations ADD CONSTRAINT FK_id_periode FOREIGN KEY (id_date)REFERENCES DIM.Periode(id_date); Go </pre>

EXEMPLE DE DIMENSION CRÉÉE : CLIENT

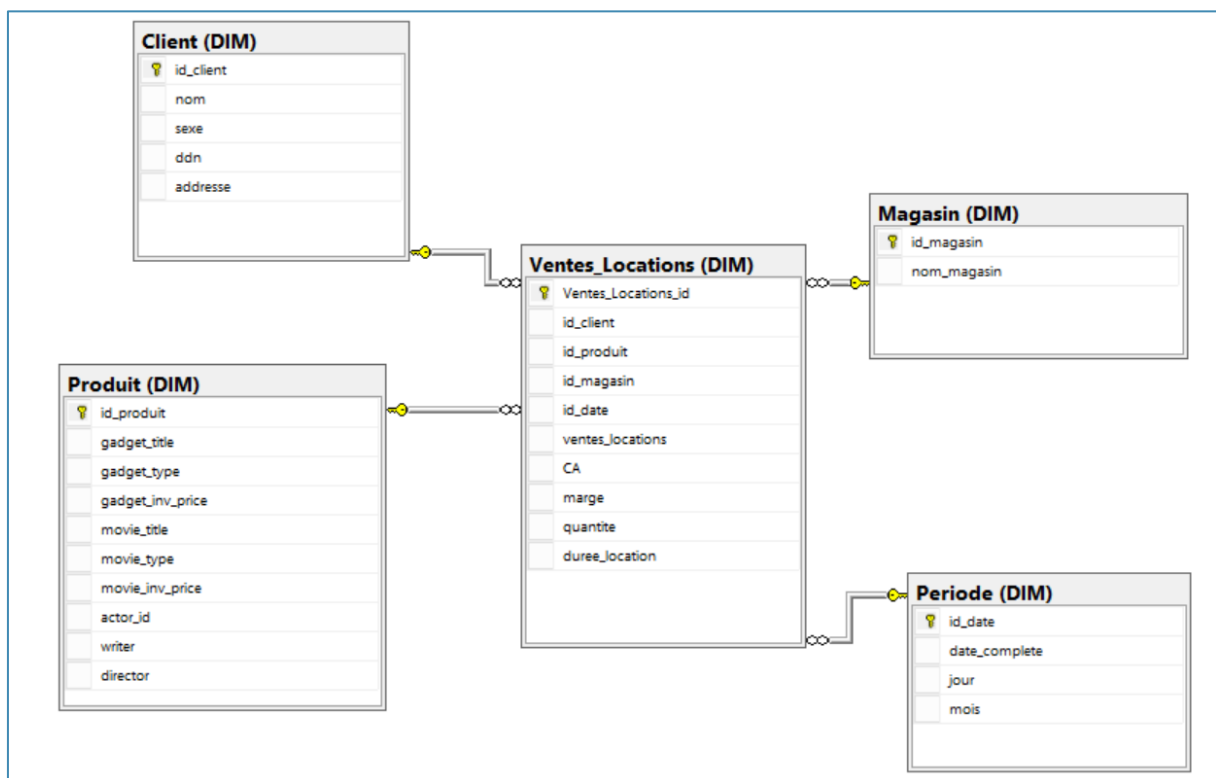
DIM.Client	
Columns	
id_client	(PK, int, not null)
nom	(varchar(50), null)
sexe	(varchar(10), null)
ddn	(date, null)
adresse	(varchar(50), null)

Grâce à une vidéo³ explicative nous avons réussi à savoir comment générer, à partir de nos schémas, ainsi que de nos interactions préalablement faites, le schéma en étoile. Le résultat est visible ci-dessous.



Nous avons ajouté, par rapport au schéma que nous avons proposé précédemment, une variable « ventes_locations » qui indiquera s'il s'agit d'une « vente » ou d'une « location ». Cela nous servira dans la suite du projet pour les différentes requêtes que nous proposerons.

SHÉMA EN ÉTOILE



³ Source : https://www.youtube.com/watch?v=Kr4tx4X_N0o

8. DÉFINITION DE REQUÊTES

Cette partie est consacrée à l'écriture de requêtes permettant de répondre à des questions précises à partir du schéma dimensionnel. Les questions, ainsi que les réponses sont présentées ci-dessous.

Quels sont les 5 films représentant les plus forts montants de ventes mensuelles ?

```
SELECT TOP 5 mois, movie_title, SUM (CA) AS CAproduit
FROM produit, ventes/locations, periode
GROUP BY movie_title, mois
ORDER BY CAproduit DESC;
```

Quel est, par produit, le montant mensuel des ventes ?

```
SELECT produit_id, mois, sum(sale_price) AS CA
FROM ventes/locations, periode
WHERE ventes_locations = 1 -- cette ligne signifie qu'on ne garde que les ventes. On ne prend pas les locations = 2.
Il existe une colonne ventes_locations qui indique si c'est une vente ou une location.
GROUP BY movie_id, mois
ORDER BY movie_id, mois;
```

Quel est l'âge moyen des clients (femmes, hommes) qui louent des films ?

```
ALTER TABLE client
age = DATEDIFF( NOW() , ddn); -- ajout d'une colonne age

SELECT sexe, MEAN(age)
FROM client
WHERE ventes_locations = 2 -- cette ligne signifie qu'on ne garde que les locations. On ne prend pas les ventes = 1.
GROUP BY sexe;
```

Quels sont, par magasin, les films les plus loués ?

Nous avons décidé d'afficher les 5 films les plus loués par magasin.

```
SELECT TOP 5 id_magasin, movie_title, COUNT(movie_title)
FROM ventes/locations, produit
WHERE ventes_locations = 2 -- cette ligne signifie qu'on ne garde que les locations. On ne prend pas les ventes = 1.
GROUP BY id_magasin, movie_title;
```

En termes de nombre de film loués par mois, quels sont les mois pour lesquels une baisse de ce nombre de plus de 15% par rapport au mois précédent est constatée ?

Ici, nous avons décidé de créer une nouvelle table contenant le nombre de film loués par mois. Une fois cette table remplie, nous effectuons des sélections permettant de dire, si oui ou non, il y a eu une baisse de 15 %.

```
CREATE TABLE table_mois (
Janv = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Janvier' and ventes_locations = 2
Fev = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Fevrier' and ventes_locations = 2
Mars = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Mars' and ventes_locations = 2
Avril = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Avril' and ventes_locations = 2
Mai = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Mai' and ventes_locations = 2
Juin = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Juin' and ventes_locations = 2
Juillet = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Juillet' and ventes_locations = 2
Aout = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Aout' and ventes_locations = 2
Sept = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Septembre' and ventes_locations = 2
Oct = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Octobre' and ventes_locations = 2
Nov = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Novembre' and ventes_locations = 2
Dec = SELECT COUNT(movie_title) as somme FROM produit, periode WHERE mois = 'Decembre'); and ventes_locations = 2);
```

<pre>SELECT CASE WHEN (Janv-Fev)/Janv > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimFev FROM table_mois, CASE WHEN (Fev-Mars)/Fev > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimMars FROM table_mois, CASE WHEN (Mars-Avril)/ Avril > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimAvril FROM table_mois, CASE WHEN (Avril-Mai)/ Avril > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimMai FROM table_mois, CASE WHEN (Mai -Juin)/ Mai > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimJuin FROM table_mois, CASE WHEN (Juin -Juillet)/ Juin > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimJuillet FROM table_mois, CASE WHEN (Juillet -Aout)/ Juillet > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimAout FROM table_mois,</pre>	<pre> CASE WHEN (Aout -Sept)/ Aout > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimSept FROM table_mois, CASE WHEN (Sept -Oct)/ Sept > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimOct FROM table_mois, CASE WHEN (Oct -Nov)/ Oct > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimNov FROM table_mois, CASE WHEN (Nov-Dec)/ Nov > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimDec FROM table_mois, CASE WHEN (Dec-Janvier)/ Dec > 0.15 THEN 'oui, il y a eu une baisse de 15 % ' ELSE 'non, pas de baisse de 15 %' END AS DimJanv FROM table_mois;</pre>
---	---

CE QU'ON AURAIT AIMÉ FAIRE

Ce projet a été réalisé lors d'une phase sanitaire assez délicate et donc une situation étudiante assez particulière. Nous avons découvert le logiciel et ses fonctions par nos propres moyens. Cela nous a pris beaucoup de temps. Nous avons énormément expérimenté avant d'en avoir une connaissance suffisante pour continuer sur la réalisation du projet. De plus, la distance entre les membres de notre groupe de travail n'a pas été simple, mais nous avons réussi à trouver une organisation satisfaisante pour continuer le projet dans les meilleures conditions.

Par manque de temps, nous aurions voulu améliorer notre intégration des données. C'est-à-dire qu'on aurait voulu créer des tables de correspondances afin d'être sûrs que le regroupement de tables soit correct. Il s'agit de jointures fiables et donc qui garantissent une unicité des différentes variables une fois qu'elles sont regroupées.

CONCLUSION

Le projet qui nous a été proposé a été, sur beaucoup de points, très enrichissant. Grâce à ce dernier, nous avons pu bénéficier d'une approche concrète des systèmes d'information décisionnels et surtout des entrepôts de données.

Nous avons expérimenté la création d'un système d'information sur la base de schéma, la création de base de données, ainsi que l'intégration de données.

Par ailleurs, cela nous a apporté de nouvelles compétences, mais a aussi enrichi notre prise d'initiative, la découverte ou l'amélioration de l'auto-formation ainsi que l'optimisation du travail en équipe à distance. Ces savoirs faire seront, sans doute, essentiels pour notre futur métier.

ANNEXES

ANNEXE 1

APERÇU DE LA FEUILLE 1 DU FICHIER DE DESCRIPTION DES DONNEES.

	A	B	C	D
1				
2	Fichier	Tables	Variables	Description des variables
3	Buckboaster	actor	actor_name	caractère / !!! Il y a un 'T' en trop au debut de la modalité
4			actor_id	numérique / de la forme T100...
5			sex	Vide
6		actsin	movieid	numérique / de la forme TMN + 5 chiffres !!! Il y a un 'T' en trop au debut de la modalité
7			actorid	numérique / T + différents chiffres !!! Il y a un 'T' en trop au debut de la modalité
8		customer	firstname	caractère / !!! Il y a un 'T' en trop au debut de la modalité
9			lastname	caractère / !!! Il y a un 'T' en trop au debut de la modalité
10			dob	date / date de naissance
11			code	caractère / identifiant unique de la forme TCC1..... !!! Il y a un 'T' en trop au debut de la modalité
12			sex	!!! Il y a un 'T' en trop au debut de la modalité
13		gadget	price	numérique
14			type	caractère / !!! Il y a un 'T' en trop au debut de la modalité
15			sale_price	numérique
16			id	numérique/nombre (de 1 à ...)
17			sale_date	date
18			cust_id	caractère / identifiant de la forme TCC10.... !!! Il y a un 'T' en trop au debut de la modalité
19			title	caractère / !!! Il y a un 'T' en trop au debut de la modalité
20		movie	movie_title	caractère / !!! Il y a un 'T' en trop au debut de la modalité
21			movie_id	caractère / identifiant unique de la forme TMN..... : !!! Il y a un 'T' en trop au debut de la modalité
22			movie_year	nombre / année / !!! Il y a un 'T' en trop au debut de la modalité
23		MSvsCompactError	ErrorCode	"-1053"
	tableau1	rassemblement des tables		

ANNEXE 2

APERÇU DU RASSEMBLEMENT DES DIFFERENTES SOURCES

	A	B	C	D	E
43	actor	actor_name		customer	last_name = cust_name = name
44		actor_id			middleware => pas utile
45		sex => vide pour l'autre table			firstname = cust_name = surname
46		movieid = movie_id => vide pour moviemegamart			sex = sex_male = gender
47					code (pas les mêmes dans les deux tables) => créer une nouvelle colonne => vide pour moviemegamart
48	gadget	price = inv_price => vide pour métrostarlet			birthday = dob = birthday
49		type = gadget_type => vide pour métrostarlet			address x3
50		id = ref_no (pas les mêmes dans les deux tables) => vide pour métrostarlet? => on met de la forme G+id			cust_id => a créer à partir de "code"
51		cust_id			
52		inv_date => vide pour buckboaster et métrostarlet ?		sales	sale_type => crée par nous qui dit soit gadget ou soit movie
53		title = gadget_title => vide pour métrostarlet?			sale_price = price_sale = sales_price
54					id = ref_no = id
55	MSysCompactError	ErrorCode			refers_to = movie_id = movie_id
56		ErrorDescription			sale_date x3
57		ErrorRecid			cust_id = soldto
58		ErrorTable			copy_id_unique => on recupère soit movie_id soit gadget_id
59					
60	movie	movie_id = id = movieid => on met de la forme M+id		rentals	date_to = to_date => vide pour bickboaster
61		inv_date => vide pour buckboaster et métrostarlet			date_from = from_date => vide pour bickboaster
62		inventory_price = inv_price = release_price			cust_id => vide pour bickboaster et moviemegamart
63		movie_type => vide pour buckboaster et métrostarlet			movie_id x2 => vide pour bickboaster
64		movie_title			ref_no = id & copy_id => vide pour bickboaster
65		movie_year = year_of_movie = production_year			price_rent = price_day => vide pour bickboaster
66		writer => vide pour buckboaster			copy_type => vide pour bickboaster et moviemegamart
67		director => vide pour buckboaster			
68		rentable => pas utile			
69		release_date => pas utile			
70		actorid = actor_id => vide pour moviemegamart			
71		type = copy_type => on se sert de cette colonne pour différencier les movies des gadgets (modalité)			

ANNEXE 3

EXEMPLE D'ANALYSE POUR LA TABLE « ACTOR » DU FICHIER BUCKBOASTER

```
/*Une librairie par fichier */
LIBNAME bb BASE
'C:\Users\EDWARD\Documents\données\données_text\ buckboaster';

/* ----- AUTOMATISATION DE L'IMPORTATION DES FICHIERS ----- */
%MACRO importTables(librairie, nomTable, cheminFichier);
PROC IMPORT OUT = &librairie..&nomTable.
    datafile =
'C:\Users\EDWARD\Documents\données\données_text\&cheminFichier.'
    DBMS = DLM REPLACE;
    DELIMITER = ";";
    GETNAMES = YES;
RUN;

%MEND importTables;
/* ----- IMPORTATION POUR BUCKBOASTER ----- */
%importTables(bb, actor, buckboaster\actor.txt);
%importTables(bb, actsin, buckboaster\actsin.txt);
%importTables(bb, customer, buckboaster\customer.txt);
%importTables(bb, gadget, buckboaster\gadget.txt);
%importTables(bb, movie, buckboaster\movie.txt);
%importTables(bb, erreur, buckboaster\MSysCompactError.txt);
%importTables(bb, sale_item, buckboaster\sale_item.txt);

/* ----- ANALYSE DE LA TABLE ACTOR DU FICHIER BUCKBOASTER ----- */

/* --- RECHERCHE DE DOUBLONS --- */
PROC SQL;
SELECT COUNT(*) AS nb_doublon, actor_id, actor_name, sex
FROM BB.ACTOR
GROUP BY actor_id, actor_name, sex
HAVING COUNT(*) > 1;
QUIT;

/* --- STATISTIQUES DESCRIPTIVES --- */
PROC PRINT DATA=BB.ACTOR;
RUN;
PROC CONTENTS DATA=BB.ACTOR;
RUN;
PROC FREQ DATA=BB.ACTOR;
RUN;

/* --- RECHERCHE DES VALEURS MANQUANTES --- */
PROC SQL ;
    SELECT nmiss(actor_name) AS actor_name_NA,
           nmiss(actor_id) as actor_id_NA
           nmiss(sex) as sex_NA
    FROM BB.ACTOR ;
QUIT ;

/* --- NOMBRE TOTAL D'OBSERVATIONS --- */
PROC SQL PRINT;
    SELECT COUNT(*) INTO :nbrobs FROM bb.actor;
QUIT;
```

ANNEXE 4

SUITE DU CODE POUR LA CREATION DES SCHEMAS ET DES TABLES

```
-- Schéma pour la source Buckboaster
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name =
'Buckboaster')
BEGIN
EXEC ('CREATE SCHEMA Buckboaster;');
END;
-- Schéma pour la source Metrostarlet
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name =
'Metrostarlet')
BEGIN
EXEC ('CREATE SCHEMA Metrostarlet;');
END;
---- Creation des tables des différentes sources pour l'importation
des donnees
---- Création des tables pour BuckBoaster:
CREATE TABLE BuckBoaster.actor(
actor_name VARCHAR(255), -- a enlever le T
actor_id VARCHAR(255), -- a changer en int après integration et
modifs
sex VARCHAR(255)-- vide);
CREATE TABLE BuckBoaster.actsin(
movieid VARCHAR(255), -- a enlever le T
actorid VARCHAR(255)-- a changer en int après integration et
modifs);
CREATE TABLE BuckBoaster.customer(
firstname VARCHAR(255), -- a enlever le T
lastname VARCHAR(255), -- a enlever le T
dob DATE, -- format 22/01/1960
code VARCHAR(255), -- a enlever le T
sex VARCHAR(255)-- a enlever le T);
CREATE TABLE BuckBoaster.gadget(
price INT,
type VARCHAR(255), -- a enlever le T
sale_price INT,
id INT,
sale_date DATE, -- format 22/01/1960
cust_id VARCHAR(255), -- a enlever le T
title VARCHAR(255) -- a enlever le T);
CREATE TABLE BuckBoaster.movie(
movie_title VARCHAR(255), -- a enlever le T
movie_id VARCHAR(255), -- a enlever le T
movie_year VARCHAR(255) -- a enlever le T !! JUSTE ANNEE);
CREATE TABLE BuckBoaster.MSysCompactError(
ErrorCode INT,
ErrorDescription VARCHAR(255),
ErrorRecid VARCHAR(255),
ErrorTable VARCHAR(255));
CREATE TABLE BuckBoaster.sale_item(
inventory_price INT,
type VARCHAR(255), -- a enlever le T
sale_price INT,
id INT,
refers_to VARCHAR(255), -- a enlever le T
sale_date DATE, -- format 27/12/2010
cust_id VARCHAR(255) -- a enlever le T);

-- Création des tables pour MetroStarlet:
CREATE TABLE MetroStarlet.actor(
actor_name varchar(255), -- a enlever le T + faire certains ont
des nombres
actor_id INT
);
CREATE TABLE MetroStarlet.acts_in(
actor_id INT,
movie_id INT
);
CREATE TABLE MetroStarlet.copy_for_rent(
copy_type varchar(255), -- a enlever le T
disposed INT,
id INT,
movie_id INT
);
CREATE TABLE MetroStarlet.copy_for_sale(
copy_type varchar(255), -- a enlever le T
is_new INT,
sale_price INT,
id INT,
sale_date DATE, --format 13/12/2010
movie_id INT,
soldto INT
);
CREATE TABLE MetroStarlet.copy_rented_to(
from_date DATE, --format 13/12/2010
to_date DATE, --format 13/12/2010
price_day INT,
copy_id INT,
cust_id INT
);
CREATE TABLE MetroStarlet.customer(
name varchar(255), -- a enlever le T
middlename varchar(255), -- a enlever le T
surname varchar(255), -- a enlever le T
gender varchar(255),
code INT,
birthday varchar(255), -- a enlever le T + format 2010/12/13
address varchar(255) -- a enlever le T
);
CREATE TABLE MetroStarlet.movie(
production_year INT, -- a enlever le T !! JUSTE ANNEE
movie_title varchar(255), -- a enlever le T
movieid INT,
writer varchar(255), -- a enlever le T
director varchar(255), -- a enlever le T
release_date DATE,
release_price FLOAT -- format 13/12/2005
);
```

ANNEXE 5

SUITE DU CODE POUR L'AMÉLIORATION DES DONNÉES

```
-- Pour Buckboaster :
-- table actor -> table actor 2
DELETE FROM Buckboaster.actor WHERE actor_name is null and actor_id is null and sex is null; -- On enlève les lignes vides
SELECT actor_id, SUBSTRING(actor_name, 2, 255) AS actor_name, sex INTO Buckboaster.actor2 -- On enlève le "T" au début de chaque
modalité de chaque colonne
FROM Buckboaster.actor;
-- table actsin -> table actsin 2
DELETE FROM Buckboaster.actsin WHERE movieid is null and actorid is null ;
SELECT SUBSTRING(movieid, 2, 255) AS movieid, SUBSTRING(actorid, 2, 255) AS actorid INTO Buckboaster.actsin2
FROM Buckboaster.actsin;
-- table customer -> table customer 2
DELETE FROM Buckboaster.customer WHERE firstname is null and lastname is null and dob is null and code is null and sex is null;
SELECT SUBSTRING(firstname, 2, 255) AS firstname, SUBSTRING(lastname, 2, 255) AS lastname, dob, SUBSTRING(code, 2, 255) AS code,
SUBSTRING(sex, 2, 255) AS sex, DATEDIFF(YEAR, dob, GETDATE()) AS age INTO Buckboaster.customer2
FROM Buckboaster.customer;
--table gadget -> table gadget 2
DELETE FROM Buckboaster.gadget WHERE price is null and type is null and sale_date is null and cust_id is null and title is null;
SELECT price, SUBSTRING(type, 2, 255) AS type, sale_price, id, sale_date, SUBSTRING(cust_id, 2, 255) AS cust_id, SUBSTRING(title, 2,
255) AS title INTO Buckboaster.gadget2
FROM Buckboaster.gadget;
--table movie -> table movie 2
DELETE FROM Buckboaster.movie WHERE movie_title is null and movie_id is null and movie_year is null ;
SELECT SUBSTRING(movie_title, 2, 255) AS movie_title, SUBSTRING(movie_id, 2, 255) AS movie_id, SUBSTRING(movie_year, 2, 255) AS
movie_year INTO Buckboaster.movie2
FROM Buckboaster.movie
--table MSysCompactError
DELETE FROM Buckboaster.MSysCompactError WHERE ErrorCode is null and ErrorDescription is null and ErrorTable is null ;
--table sale_item -> table sale_item 2
DELETE FROM Buckboaster.sale_item WHERE inventory_price is null and type is null and sale_price is null and id is null and refers_to is
null and sale_date is null and cust_id is null;
SELECT inventory_price, SUBSTRING(type, 2, 255) AS type, sale_price, id,
SUBSTRING(refers_to, 2, 255) AS refers_to, sale_date, SUBSTRING(cust_id, 2, 255) AS cust_id
INTO Buckboaster.sale_item2 FROM Buckboaster.sale_item;
-- Pourmoviemegamart
--table gadgetsales -> table gadgetsales 2
DELETE FROM Moviemegamart.gadgetsales WHERE ref_no is null and sale_date is null and cust_name is null and sex_male is null and
birthdate is null and price_sale is null and address is null and gadget_type is null and inv_date is null and inv_price is null and
gadget_title is null;
SELECT ref_no, sale_date, SUBSTRING(cust_name, 2, 255) AS cust_name, sex_male, birthdate, price_sale, SUBSTRING(address, 2, 255)
AS address, SUBSTRING(gadget_type, 2, 255) AS gadget_type, inv_date, inv_price, SUBSTRING(gadget_title, 2, 255) AS gadget_title,
DATEDIFF(YEAR, birthdate, GETDATE()) AS age INTO moviemegamart.gadgetsales2
FROM moviemegamart.gadgetsales
--table movierentals -> table movierentals 2
DELETE FROM Moviemegamart.movierentals WHERE date_to is null and date_from is null and cust_name is null and sex_male is null
and birthdate is null and movie_id is null and address is null and ref_no is null and price_rent is null ;
SELECT date_to, date_from, SUBSTRING(cust_name, 2, 255) AS cust_name, sex_male, birthdate, movie_id, ref_no, SUBSTRING(address,
2, 255) AS address, price_rent, DATEDIFF(YEAR, birthdate, GETDATE()) AS age INTO moviemegamart.movierentals2
FROM moviemegamart.movierentals
--table moviesales -> table moviesales 2
DELETE FROM Moviemegamart.moviesales WHERE sale_date is null and cust_name is null and sex_male is null and birthdate is null and
movie_id is null and address is null and ref_no is null and price_sale is null ;
SELECT sale_date, SUBSTRING(cust_name, 2, 255) AS cust_name, sex_male, birthdate, movie_id, ref_no, SUBSTRING(address, 2, 255)
AS address, price_sale, DATEDIFF(YEAR, birthdate, GETDATE()) AS age INTO moviemegamart.moviesales2
FROM moviemegamart.moviesales
```

ANNEXE 6

SUITE DU CODE POUR L'INTÉGRATION DES DONNÉES

```
-- Table Acteurs
-- Integration
INSERT INTO MoreMovies.Acteurs (actor_name)
SELECT actor_name
FROM Buckboaster.actor2
UNION
SELECT actor_name
FROM Metrostarlet.actor2
ORDER BY 1 ;

-- Table Gadget
-- Integration
INSERT INTO MoreMovies.Gadget (gadget_inv_date, gadget_inv_price, gadget_sale_price,
gadget_tile, gadget_type)
SELECT sale_date, price, sale_price, title, type
FROM Buckboaster.gadget2
UNION
SELECT sale_date, inv_price, price_sale, gadget_title, gadget_type
FROM Moviemegamart.gadgetsales2
ORDER BY 1 ;

-- Table Films
-- Integration
INSERT INTO MoreMovies.Films(movie_title, movie_year, actor_id, movie_inv_price, writer,
director)
SELECT movie_title, movie_year, NULL, NULL, NULL, NULL
FROM Buckboaster.movie2
UNION
SELECT movie_title, production_year, NULL, release_price, writer, director
FROM Metrostarlet.movie2
ORDER BY 1 ;
```