SOFTWARE DESIGN AND CONSTRUCTION
Assignment 09 – Data Modeling

Lecturer: NGUYEN Thi Thu Trang, trangntt@soict.hust.edu.vn

1. SUBMISSION GUIDELINE

When you want to submit your individual work of in-class tasks for the Case Study, you have to push your work to your individual GitHub repository, complied with the naming convention "TeamName-StudentID.StudentName" (e.g. TKXDPM.KHMT.20231.20192012.HoangNghiaPhu or TKXDPM.VP.20231-20192122.LuongHongHai).

2. IN-CLASS ASSIGNMENT

In this section, we will get familiar with the software detailed design process and try ourselves with data modeling for the Case Study.

You are asked to work individually for this section, and then put all your file(s) and directories to a directory, namely "DetailedDesign/DataModeling". After that, push your commit to your individual repository before the announced deadline.

You may need free tools such as MySQL Workbench, moqups with template[1], and draw.io, or paid apps like Astah Pro, Navicat, and DataGrip in this lab for the purpose of data modeling.

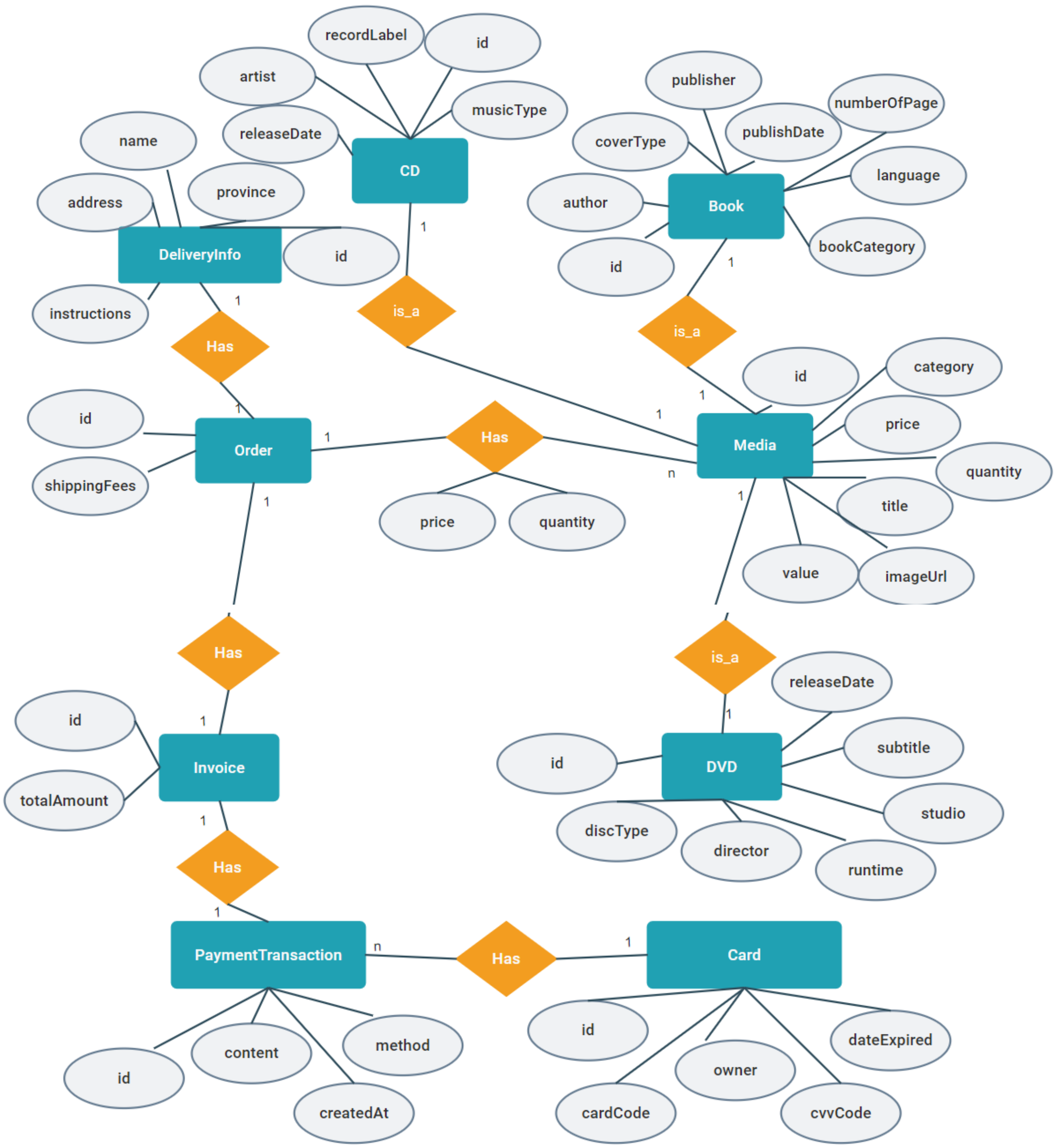
2.1. CONCEPTUAL DATA MODEL

Conceptual data model is a high-level data model that abstracts the natural expressions without any constraints imposed by database management system (DBMS) like PostgreSQL, SQLite, Microsoft Access, or MongoDB. A conceptual data model can be expressed by Entity-Relationship (ER) diagram,

To illustrate, we will create an ER diagram for AIMS.

[1] https://moqups.com/templates/diagrams-flowcharts/erd/

AIMS System ERD



2.2. DATABASE DESIGN

In this part, we need specify what is the decision of the Database Management System (DBMS) and describe the DBMS.

For example, we would use SQLite 3.7.2 as our DBMS of the Case Study. We choose SQLite because:

- SQLite is an open-source relational database management system.
- It is not only a small, fast, popular, self-contained, high-reliability, full-featured, SQL database engine but also stable, cross-platform, and backwards compatible with long-term support [2].
- It has bindings to Java. See https://github.com/xerial/sqlite-jdbc

On the other hand, SQLite cannot work as a server-side database like MySQL or PostgreSQL, which any ecommerce system would need, since it stores user data in the local device.

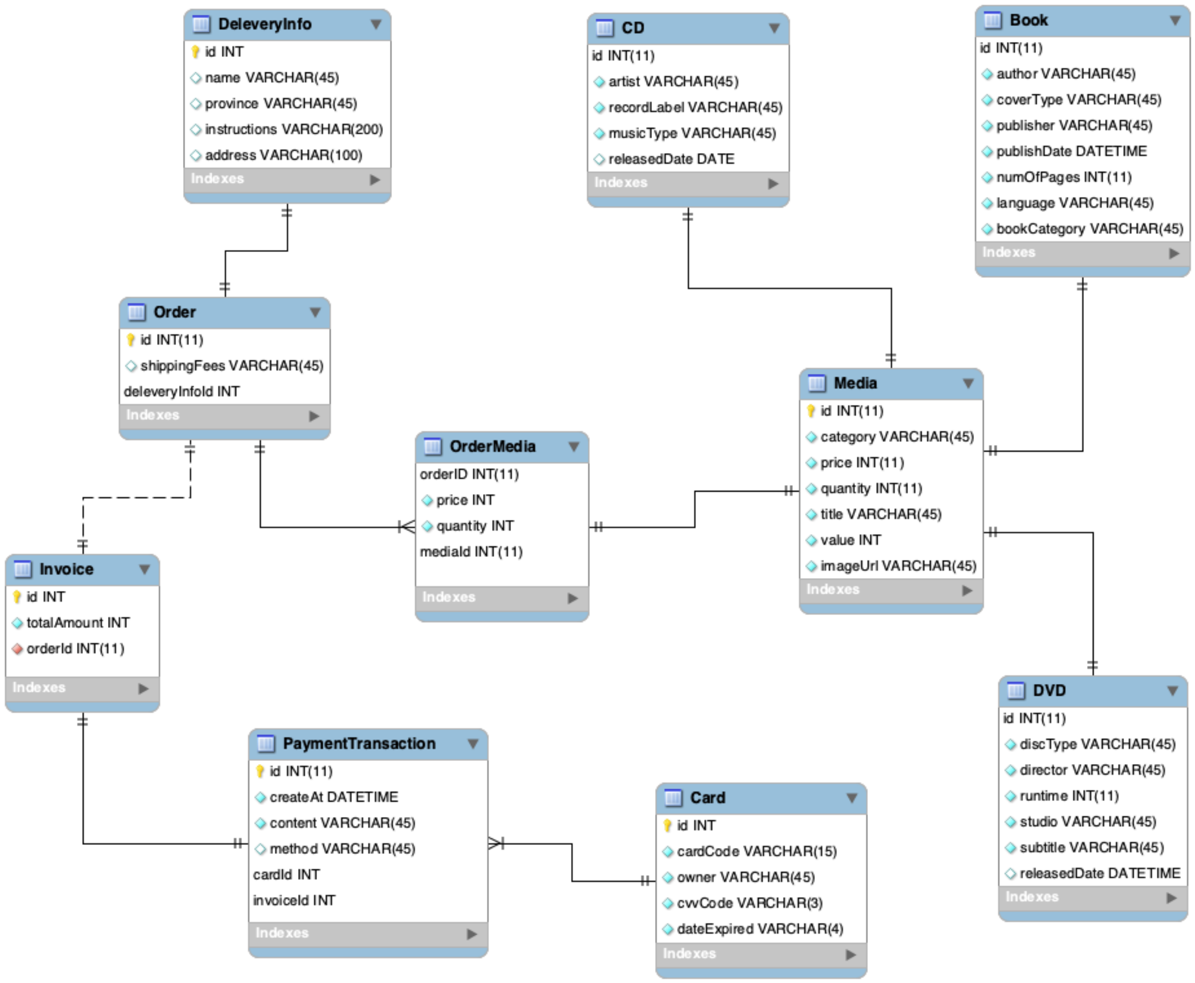
In this course, we, however, use SQLite since the sample project for the Case Study only runs locally, and it is still able to give sufficient illustration to the learners. Note that SQLite is used for students to practice with DB, but not suitable for such real-life e-commerce system.

2.2.1. Logical Data Model

From the conceptual data model (i.e., ER diagram) in the previous section, we can achieve the logical data model with respect to the chosen DBMS (i.e., SQLite).

Here is a logical data model regarding our ER diagram.

[2] https://www.sqlite.org/index.html



2.2.2. Physical Data Model

In this part, we need to give a detail design of each element in the DB diagram. For instance, in a Relational DBMS, we give a detail design for each Table and their constraints, illustrated in below table (PK: Primary Key, FK: Foreign Key).

- **Media**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	x		id	Integer	Yes	ID, auto increment
2.			category	VARCHAR(45)	Yes	Media type, e.g., CD, DVD
3.			price	Integer	Yes	Current price
4.			quantity	Integer	Yes	Number of products

#	PK	FK	Column Name	Data type	Mandatory	Description
5.			title	VARCHAR(45)	Yes	Product name
6.			value	Integer	Yes	Value of the product
7.			imageUrl	VARCHAR(45)	Yes	Product image path

- CD

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		x	id	Integer	Yes	ID, same as ID of Media of which type is CD
2.			artist	VARCHAR(45)	Yes	Artist's name
3.			recordLabel	VARCHAR(45)	Yes	Record label
4.			musicType	VARCHAR(45)	Yes	Music genres
5.			releasedDate	DATE	No	Release date

- Book

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		x	id	Integer	Yes	ID, same as ID of Media of which type is Book
2.			author	VARCHAR(45)	Yes	Author
3.			coverType	VARCHAR(45)	Yes	Cover type
4.			Publisher	VARCHAR(45)	Yes	Publishing house
5.			publishDate	DATETIME	Yes	Date of publishing

#	PK	FK	Column Name	Data type	Mandatory	Description
6.			numOfPages	Integer	Yes	Page number
7.			language	VARCHAR(45)	Yes	Language
8.			bookCategory	VARCHAR(45)	Yes	Book category

- **DVD**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		x	id	Integer	Yes	ID, same as ID of Media of which type is DVD
2.			discType	VARCHAR(45)	Yes	Disc type
3.			director	VARCHAR(45)	Yes	Director
4.			runtime	Integer	Yes	Duration
5.			studio	VARCHAR(45)	Yes	Manufacturer
6.			subtitle	VARCHAR(45)	Yes	Subtitles
7.			releasedDate	DATETIME	Yes	Release date
8.			filmType	VARCHAR(45)	Yes	Genres

- **Card**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	x		id	Integer	Yes	ID, auto increment
2.			cardCode	VARCHAR(45)	Yes	Card code

#	PK	FK	Column Name	Data type	Mandatory	Description
3.			owner	VARCHAR(45)	Yes	Cardholders
4.			cvvCode	VARCHAR(3)	Yes	CVV code
5.			dateExpired	VARCHAR(4)	Yes	Expiration date

- **DeliveryInfo**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	X		id	Integer	Yes	ID, auto increment
2.			name	VARCHAR(45)	Yes	Receiver name
3.			province	VARCHAR(45)	Yes	Provinces
4.			instructions	VARCHAR(200)	No	Delivery instructions
5.			address	VARCHAR(100)	Yes	Delivery address

- **Order**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	X		id	Integer	Yes	ID
2.			shippingFees	VARCHAR(45)	Yes	Shipping fee
3.		X	deliveryInfoId	Integer	Yes	Delivery Info ID

- **OrderMedia**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		X	mediaID	Integer	Yes	Media ID
2.		X	orderID	Integer	Yes	Order ID
3.			price	Integer	Yes	Selling price
4.			quantity	Integer	Yes	Number

- **Invoice**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	x		id	Integer	Yes	ID
2.			totalAmount	Integer	Yes	Total
3.		x	orderId	Integer	Yes	Order ID

- **PaymentTransaction**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	x		id	Integer	Yes	ID
2.			createAt	DATETIME	Yes	Date of creation
3.			content	VARCHAR(45)	Yes	Transaction contents
4.			method	VARCHAR(45)	Yes	Payment methods
5.		x	cardId	Integer	Yes	ID of used card
6.		x	invoiceId	Integer	Yes	Invoice ID

Finally, we need a database script. With specialized database development tools and plugins, we can generate a database script directly from logical data model.

```
BEGIN;
CREATE TABLE "aims"."Media"(
  "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  "category" VARCHAR(45) NOT NULL,
  "price" INTEGER NOT NULL,
  "quantity" INTEGER NOT NULL,
  "title" VARCHAR(45) NOT NULL,
  "value" INTEGER NOT NULL,
  "imageUrl" VARCHAR(45) NOT NULL
);
CREATE TABLE "aims"."CD"(
  "id" INTEGER PRIMARY KEY NOT NULL,
  "artist" VARCHAR(45) NOT NULL,
  "recordLabel" VARCHAR(45) NOT NULL,
  "musicType" VARCHAR(45) NOT NULL,
  "releasedDate" DATE,
  CONSTRAINT "fk_CD_Media1"
    FOREIGN KEY("id")
    REFERENCES "Media"("id")
);
CREATE TABLE "aims"."Book"(
  "id" INTEGER PRIMARY KEY NOT NULL,
  "author" VARCHAR(45) NOT NULL,
  "coverType" VARCHAR(45) NOT NULL,
  "publisher" VARCHAR(45) NOT NULL,
  "publishDate" DATETIME NOT NULL,
  "numOfPages" INTEGER NOT NULL,
  "language" VARCHAR(45) NOT NULL,
  "bookCategory" VARCHAR(45) NOT NULL,
  CONSTRAINT "fk_Book_Media1"
    FOREIGN KEY("id")
    REFERENCES "Media"("id")
);
CREATE TABLE "aims"."DeleveryInfo"(
  "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  "name" VARCHAR(45),
  "province" VARCHAR(45),
  "instructions" VARCHAR(200),
  "address" VARCHAR(100)
);
CREATE TABLE "aims"."Card"(
  "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  "cardCode" VARCHAR(15) NOT NULL,
  "owner" VARCHAR(45) NOT NULL,
  "cvvCode" VARCHAR(3) NOT NULL,
  "dateExpired" VARCHAR(4) NOT NULL
);
CREATE TABLE "aims"."DVD"(
  "id" INTEGER PRIMARY KEY NOT NULL,
  "discType" VARCHAR(45) NOT NULL,
  "director" VARCHAR(45) NOT NULL,
  "runtime" INTEGER NOT NULL,
  "studio" VARCHAR(45) NOT NULL,
  "subtitle" VARCHAR(45) NOT NULL,
  "releasedDate" DATETIME,
  CONSTRAINT "fk_DVD_Media1"
    FOREIGN KEY("id")
    REFERENCES "Media"("id")
);
CREATE TABLE "aims"."Order"(
  "id" INTEGER NOT NULL,
  "shippingFees" VARCHAR(45),
  "deleveryInfoId" INTEGER NOT NULL,
  PRIMARY KEY("id","deleveryInfoId"),
  CONSTRAINT "fk_Order_DeleveryInfo1"
    FOREIGN KEY("deleveryInfoId")
    REFERENCES "DeleveryInfo"("id")
);
CREATE INDEX "aims"."Order.fk_Order_DeleveryInfo1_idx" ON "Order"
("deleveryInfoId");
CREATE TABLE "aims"."OrderMedia"(
  "orderID" INTEGER NOT NULL,
  "price" INTEGER NOT NULL,
  "quantity" INTEGER NOT NULL,
  "mediaId" INTEGER NOT NULL,
  PRIMARY KEY("orderID","mediaId"),
  CONSTRAINT "fk_ordermedia_order"
    FOREIGN KEY("orderID")
    REFERENCES "Order"("id"),
  CONSTRAINT "fk_OrderMedia_Media1"
    FOREIGN KEY("mediaId")
    REFERENCES "Media"("id")
);
CREATE INDEX "aims"."OrderMedia.fk_ordermedia_order_idx" ON "OrderMedia"
("orderID");
CREATE INDEX "aims"."OrderMedia.fk_OrderMedia_Media1_idx" ON "OrderMedia"
("mediaId");
CREATE TABLE "aims"."Invoice"(
  "id" INTEGER PRIMARY KEY NOT NULL,
  "totalAmount" INTEGER NOT NULL,
  "orderId" INTEGER NOT NULL,
  CONSTRAINT "fk_Invoice_Order1"
    FOREIGN KEY("orderId")
    REFERENCES "Order"("id")
);
CREATE INDEX "aims"."Invoice.fk_Invoice_Order1_idx" ON "Invoice" ("orderId");
CREATE TABLE "aims"."PaymentTransaction"(
  "id" INTEGER NOT NULL,
  "createAt" DATETIME NOT NULL,
  "content" VARCHAR(45) NOT NULL,
  "method" VARCHAR(45),
  "cardId" INTEGER NOT NULL,
  "invoiceId" INTEGER NOT NULL,
  PRIMARY KEY("id","cardId","invoiceId"),
  CONSTRAINT "fk_PaymentTransaction_Card1"
    FOREIGN KEY("cardId")
    REFERENCES "Card"("id"),
  CONSTRAINT "fk_PaymentTransaction_Invoice1"
    FOREIGN KEY("invoiceId")
    REFERENCES "Invoice"("id")
);
CREATE INDEX "aims"."PaymentTransaction.fk_PaymentTransaction_Card1_idx" ON "PaymentTransaction" ("cardId");
CREATE INDEX "aims"."PaymentTransaction.fk_PaymentTransaction_Invoice1_idx" ON "PaymentTransaction" ("invoiceId");
COMMIT;
```

2.3. DATA MODELING FOR UC “PLACE RUSH ORDER”

You are asked to update the data models with use case “Place Rush Order.”

When you finish all the tasks, please export your work into a PDF file, and push everything to your individual repository before the announced deadline.