Base de Datos 2020
Lab X: MongoDB - Modelado de datos

Sergio Canchi, Cristian Cardellino,
Ramiro Demasi, Diego Piloni

Parte I

Continuando con la base de datos mflix:

Agregar las siguientes reglas de validación usando JSON Schema. Luego de cada especificación testear que efectivamente las reglas de validación funcionen, intentando insertar 5 documentos válidos y 5 inválidos (por distintos motivos).

1. Especificar en la colección *users* las siguientes reglas de validación: El campo *name (requerido)* debe ser un string con un máximo de 30 caracteres, *email (requerido)* debe ser un string que matchee con la expresión regular: `"^(.*)@(.*)\\.(.{2,4})$"` , *password* (requerido) debe ser un string con al menos 50 caracteres.
2. Obtener metadata de la colección *users* que garantice que las reglas de validación fueron correctamente aplicadas.
3. Especificar en la colección *theaters* las siguientes reglas de validación: El campo *theaterId (requerido)* debe ser un int y *location (requerido)* debe ser un object con:
    a. un campo *address (requerido) que sea un object con campos street1, city, state y zipcode* todos de tipo string y requeridos
    b. un campo *geo (no requerido) que sea un object* con un campo *type*, con valores posibles *"Point"* o *null* y *coordinates que debe ser una lista de 2 doubles*

    Por último, estas reglas de validación no deben prohibir la inserción o actualización de documentos que no las cumplan sino que solamente deben advertir.
4. Especificar en la colección *movies* las siguientes reglas de validación*:* El campo title (requerido) es de tipo string, year (requerido) int con mínimo en 1900 y máximo en 3000, y que tanto *cast*, *directors*, *countries*, como *genres* sean arrays de strings sin duplicados.
5. Crear una colección user*Profiles* con las siguientes reglas de validación: Tenga un campo *user_id* (requerido) de tipo "objectId", un campo *language* (requerido) con alguno de los siguientes valores ["English", "Spanish", "Portuguese"] y un campo *favorite_genres* (no requerido) que sea un array de strings sin duplicados.

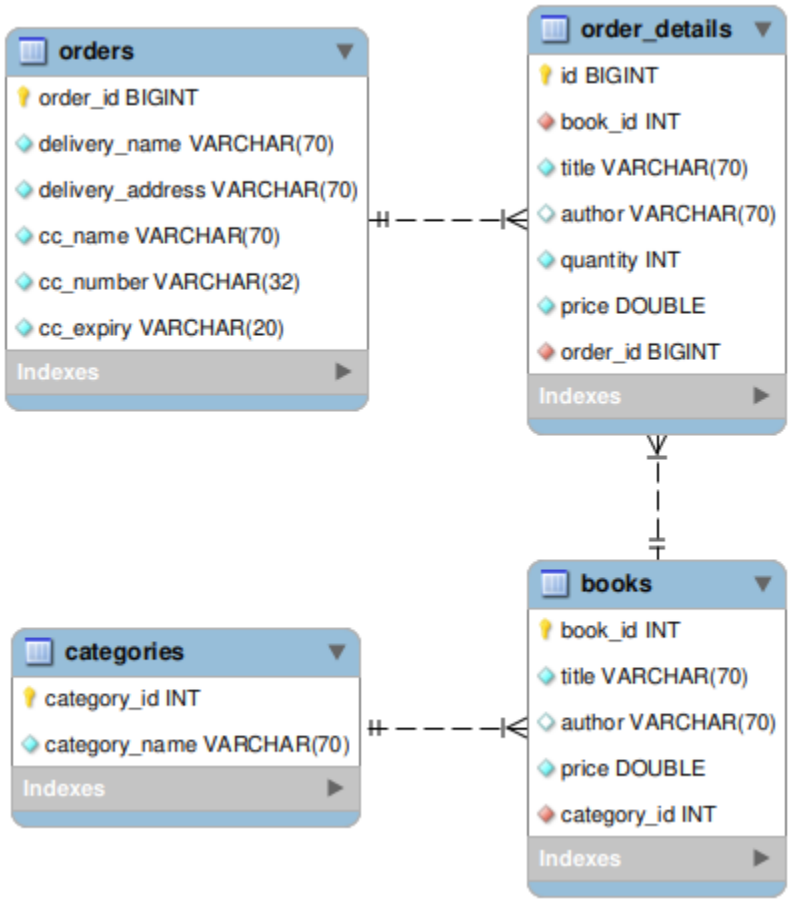
Modelo de relaciones en MongoDB

6. Identificar los distintos tipos de relaciones (One-To-One, One-To-Many) en las colecciones *movies* y *comments.* Determinar si se usó embedding o referenciación en cada relación y justificar la razón.

Parte II

Contexto

La siguiente figura muestra el modelo de datos de la base de datos **shop.** El archivo shop.tar.gz contiene los scripts ddl y dml (no tiene datos para todas las tablas).



Tareas

Resolver los ejercicios usando las reglas de modelado de relaciones y/o patrones de diseño, justificando las decisiones tomadas.

1. Crear el modelo de datos en mongodb para shop.
2. Crear el dataset para las colecciones de shop para esta versión inicial.
3. Actualizar el modelo datos según las siguientes necesidades:
    a. Modelar información de los usuarios (username, password, email, name, address, contacts).
    b. Modelar información de reviews de libros (title, content, rating) realizados por los usuarios.

c. Se necesita acceder a la cantidad y promedio de rating de reviews por libro. Dicha información debe poder consultarse de manera rápida cada vez que se accede a la información de un libro, y se va a acceder de manera frecuente.

4. Crear el dataset para las colecciones de shop para esta nueva versión.

Parte III

Contexto

La base de datos **analytics** contiene información de servicios financieros. A continuación se describen las colecciones que contiene.

Colección	**Descripción**
accounts	Contiene información de las cuentas de los clientes.
customers	Contiene información de los clientes.
transactions	Contiene información de las transacciones de los clientes.

Descargar la base de datos analytics.tar.gz y restaurar localmente

$ tar xzvf analytics.tar.gz
$ mongorestore --host <host> --drop --gzip --db analytics analytics/

Tareas

Resolver los ejercicios usando las reglas de modelado de relaciones y/o patrones de diseño, justificando las decisiones tomadas.

1. Actualizar el modelo de datos de la base de datos analytics para que pueda contemplar las siguientes nuevas necesidades:
   (i) Poder almacenar los reportes diarios, semanales y anuales de las transacciones. Cada tipo de reporte (diario, semanal, o anual) debe almacenar el monto total y cantidad de transacciones por código de transacción (transaction_code).
   (ii) La información de los reportes debe ser almacenada en una sola colección.
   (iii) Definir la regla de validación de esquemas de datos de los reportes.

   (Hint: Puede considerar los siguientes documentos como guía. Donde se puede observar que cada reporte tiene campos en común y otros campos que son propios de cada tipo de reporte)

{
summary_type: "daily"
transaction_code: "sell"
total: <>
count: <>
date: <>
}
	{
	summary_type: "weekly"
	transaction_code: "buy"
	total: <>
	count: <>
	summary_start_date: <>
	summary_end_date: <>
	}
		{
		summary_type: "annually"
		transaction_code: "buy"
		total: <>
		count: <>
		year: <>
		}

2. Crear las consultas para calcular los reportes del punto anterior.
3. (Opcional) Crear un script (se recomienda javascript o python) que permita calcular y almacenar reportes (es decir, usando lo definido en los puntos 2 y 1 respectivamente)