# Bases de Datos 2020
# Recuperatorio 2: NoSQL

Sergio Canchi, Cristian Cardellino,
Ramiro Demasi, Diego Piloni

## Contexto

La base de datos **sakila** contiene tres colecciones con información de alquileres de películas en tiendas físicas. Las colecciones que contiene son:

| **Colección** | **Descripción** |
|---|---|
| stores | Contiene información de las tiendas. |
| customers | Contiene información de los clientes. |
| films | Contiene información de las películas. |

Para cargar la base de datos **sakila** ejecutar los siguiente comandos:

```
$ tar -xvzf sakila.tgz
$ mongoimport --db sakila --collection stores --drop --file stores.json
$ mongoimport --db sakila --collection customers --drop --file customers.json
$ mongoimport --db sakila --collection films --drop --file films.json
```

## Consignas

**Operaciones CRUD**

1) Listar el nombre (*first name*), apellido (*last name*) y teléfono (*phone*) de los clientes de India, de distritos cuya primera letra sea “A” (también listar el distrito) y que hayan alquilado al menos una película en los primeros 15 días de Junio de 2005. (Hint: Aquellos campos cuyo nombre contengan espacios deberán ser referenciados siempre entre comillas, por ejemplo, “First Name” o “Rentals.Rental Date”).

2) Listar el título, rating y duración (*length*), del top 10 de películas con mayor duración, cuyo rating sea “G” o “PG”. (Hint: No es suficiente con proyectar el campo “Length” sino que se debe usar el operador $toInt para ordenar correctamente, dado que el tipo de “Length” es string).

3) Para aquellos clientes Argentinos que hayan realizado el pago del alquiler de al menos una película con precio mayor o igual a $8 (*Rentals.Payments.Amount*) entre el 10 y 20 de Junio de 2005 inclusive, actualizar el cliente agregando un campo “AvailableDiscount” con el valor 50. (Nota: Si bien *Rentals.Payments* es un array, se puede asumir que ese array es siempre de tamaño 1, es decir, todos los alquileres se realizaron en 1 solo pago).

**Pipeline de Agregación**

4) Crear la vista “most_expensive_movie_rented_per_customer” que liste el nombre y apellido del cliente y monto de la película más cara que haya alquilado cada cliente. Listar ordenado por monto de manera decreciente. (Hint: Recordar que operadores de agregación como $max y $first también pueden usarse en $project para operar sobre arrays).

5) Listar el id (*filmId*) de cada película alquilada junto con su título y la suma total de dinero gastado en alquileres de esa película por todos los clientes. Es decir, si una película con *id* 123 y *título* “Inception” se alquiló tres veces por $1, $1.2 y $1.5, entonces esta película debería aparecer en el resultado como:

```
...
 {
  filmId: 123,
  title: "Inception"
  total_spent: 3.7,
 }
...
```

6) Listar las categorías y la cantidad de alquileres del top 10 de categorías más alquiladas por clientes de China, ordenadas por cantidad de alquileres de manera decreciente.
El esquema de datos de salida debe ser el siguiente:

```
{
  Category: “string”,
  RentAmount: “int”
}
```

**Validación y Modelado de Datos**

7)

a) Especificar en la colección *stores* las siguientes reglas de validación: Los campos *Address y City* (*requeridos*) deben ser strings con un máximo de 50 caracteres, *Country (requerido)* debe ser alguno de los siguientes strings*: ["Argentina", "Australia", "Canada", "Uruguay"], Manager First Name y Manager Last Name* deben ser strings que comiencen con una letra mayúscula, *Phone* un string que contenga 0 o más números, con un máximo de 20 caracteres y *Staff* debe ser un array de objetos, con un campo *staffId* (*requerido*) de tipo entero.

b) Luego de especificar las reglas de validación intentar insertar 2 documentos válidos y 2 inválidos a stores (por distintas razones) y justificar los inválidos.

8) Evolucionar el modelo de datos de acuerdo a la siguiente necesidad: Se desea almacenar el inventario que posee cada tienda. El inventario debe permitir conocer qué películas tiene cada tienda, cuántas tiene en total y cuántas disponibles de cada película para alquilar.

a) Justificar cómo resolvería esta necesidad (reglas de modelado de relaciones/patrones de diseño/sentido común). Cargar al menos 2 películas al inventario de al menos 2 tiendas.

b) Definir una consulta que dado el título de una película permita saber qué tiendas la tienen disponible para alquilar.
El esquema de datos de salida debe ser el siguiente:

```
{
   FilmTitle: "string",
   AvailableStores: "array de stores"
}
```

Donde AvailableStores contiene toda la información de todos las tiendas (sin duplicados) que tengan la película disponible para alquilar.

# Puntos a tener en cuenta

- Resolver las consultas sin vistas salvo que se lo requiera explícitamente.
- Mostrar únicamente los campos pedidos en la consigna.
- Buscar hacer la consulta de la forma más sencilla y eficiente posible.
- Se evaluará el correcto formato de las soluciones:
    - El código entregado debe ser legible.

    - Utilizar indentación de espacios de manera uniforme.

# Entrega

- Se entregará un archivo comprimido `soluciones.js.gz` o `soluciones.zip` (con `soluciones.js` adentro) con todas las soluciones de todos los ejercicios. Separar las soluciones mediante comentarios de javascript.
- La entrega se hará mediante el **Aula Virtual** en el correspondiente apartado.
    - Tendrán hasta las 18:30 para que se considere una entrega completa. La recomendación es empezar a subir el archivo a las 18 hs.
    - Si se entrega después de esa hora, el límite serán las 19:00 y se descontará 1 punto por entrega tardía.
    - Después de las 19:00 se cerrará la entrega y el examen se considerará desaprobado.