Caso 1: Organismo Supervisor de las Contrataciones del Estado

Pregunta 1 (2 p.)
Mostrar la lista de entidades que no han presentados procesos para un determinado año. Para cada entidad mostrar el código y el RUC.

```
create function f_entidades_sin_procesos (@year int) returns table as return
select codigo, ruc
from entidades
where codigo not in
      (select codigo_entidad from procesos where year(fecha_publicacion) = @year)
go
```

Pregunta 2 (2 p.)
Mostrar la cantidad de ofertas presentadas por cada postor. Para cada postor mostrar el código, el RUC, el tipo de postor (persona natural o persona jurídica) y la cantidad de ofertas presentadas.

```
select p.codigo, p.ruc, 'Empresa' as tipo, count(*) as cantidad
from empresas as e
    inner join postores as p on e.codigo = p.codigo
    inner join participantes as pa on p.codigo = pa.codigo_postor
    inner join ofertas as o on pa.codigo = o.codigo_participa
group by p.codigo, p.ruc

union

select p.codigo, p.ruc, 'Persona' as tipo, count(*) as cantidad
from personas as pe
    inner join postores as p on pe.codigo = p.codigo
    inner join participantes as pa on p.codigo = pa.codigo_postor
    inner join ofertas as o on pa.codigo = o.codigo_participa
group by p.codigo, p.ruc
go
```

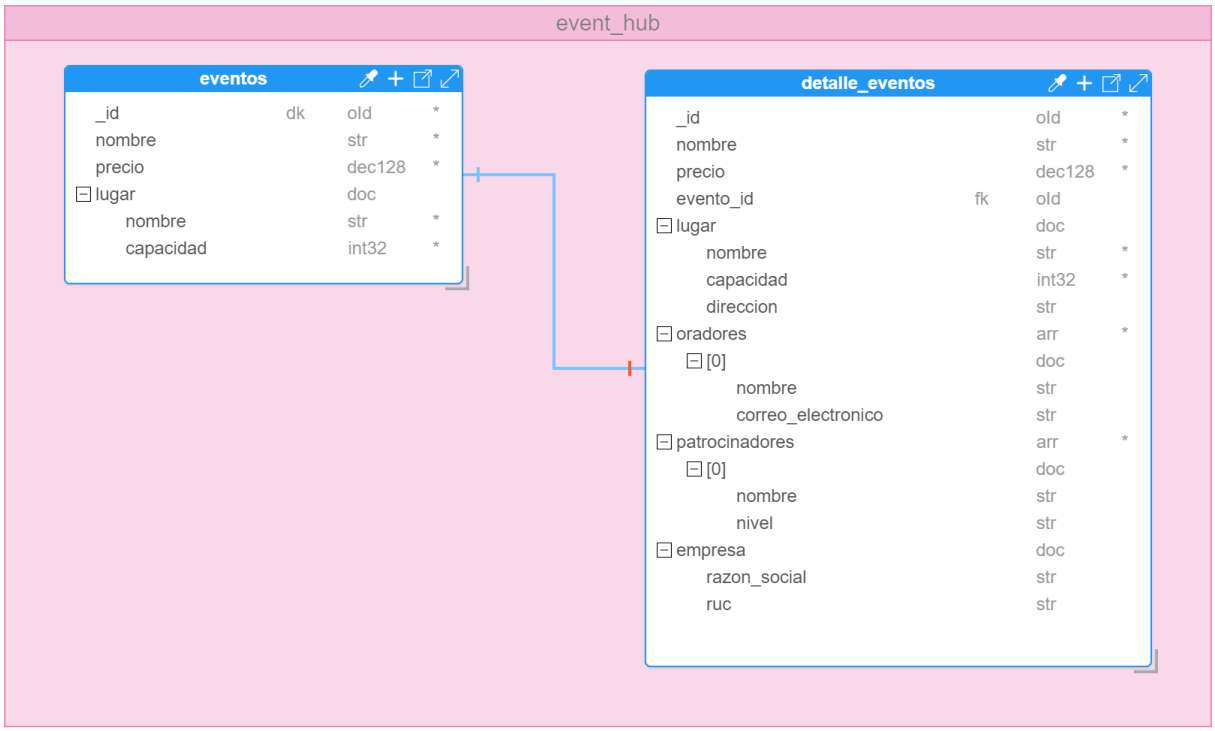
Pregunta 3 (2 p.)
Mostrar el proceso con la mayor cantidad de ofertas recibidas para un determinado año (considerar la fecha de presentación de la oferta). Para cada proceso mostrar el nombre y el valor referencial.

```
create view v_ofertas_por_proceso_por_anho as
select pro.nombre, pro.valor_referencial, year(fecha_oferta) as anho, count(*) as
cantidad
from procesos as pro
    inner join participantes as pa on pro.codigo = pa.codigo_proceso
    inner join ofertas as o on pa.codigo = o.codigo_participa
group by pro.nombre, pro.valor_referencial, year(fecha_oferta)
go
```

```
create function f_proceso_con_mas_ofertas(@year int) returns table
as
return
select nombre, valor_referencial
from v_ofertas_por_proceso_por_anho
where anho = @year and
    cantidad = (select max (cantidad) from v_ofertas_por_proceso_por_anho   where
anho = @year)
go
```

Caso 2: EventHub

Pregunta 4 (2 p.).
Crear un diagrama de documentos que considere al menos dos (2) colecciones para mostrar los datos más relevantes de los eventos a las personas interesadas en participar en ellos.



Pregunta 5 (2 p.).
Indicar los patrones de modelado de datos utilizados en el diagrama de documentos.

Embedded document pattern

- eventos
  El campo **lugar** es un documento embebido que contiene los datos del lugar donde se realiza el evento.

- detalle eventos
  El campo **lugar** es un documento embebido que contiene los datos del lugar donde se realiza el evento.
  El campo **oradores** un arreglo de documentos embebidos que contiene los datos de los oradores que participan en el evento.
  El campo **patrocinadores** un arreglo de documentos embebidos que contiene los datos de los patrocinadores que auspician el evento.
  El campo **empresa** es un documento embebido que contiene los datos de la empresa que organiza el evento.

Subset pattern (documento embebido)

- La colección **eventos** es un subconjunto de la colección **detalle_eventos**. La colección muestra los datos más relevantes de cada evento, mientras que la colección **detalle_eventos** muestra el detalle de cada evento.

Reference pattern

- Por cada documento de la colección **evento** existe un documento en la colección **detalle_eventos**, dado que se trata del mismo evento. Para ello se requiere que en cada documento de la colección **detalle_eventos** haya una referencia al documento correspondiente en la colección **eventos**.

Pregunta 6 (2 p.).

- Establecer una regla de validación utilizando JSON Schema para una de las colecciones mostradas en el diagrama de documentos.

```
db.createCollection("eventos", {
    "validator": {
        "$jsonSchema": {
            "bsonType": "object",
            "properties": {
                "_id": {
                    "bsonType": "objectId"
                },
                "nombre": {
                    "bsonType": "string"
                },
                "precio": {
                    "bsonType": "decimal"
                },
                "lugar": {
                    "bsonType": "object",
                    "properties": {
                        "nombre": {
                            "bsonType": "string"
                        },
                        "capacidad": {
                            "bsonType": "int"
                        }
                    },
                    "required": ["nombre", "capacidad"]
                }
            },
            "required": ["_id", "nombre", "precio"]
        }
    },
});
```

Caso 3: Booking.com

Pregunta 7 (2 p.).

Establecer una regla de validación utilizando JSON Schema para una colección de documentos que representen el detalle por hotel de acuerdo con las pantallas mostradas.

```
db.createCollection("detalle_hoteles", {
    "validator": {
        "$jsonSchema": {
            "bsonType": "object",
            "properties": {
                "_id": {
                    "bsonType": "objectId"
                },
                "nombre": {
                    "bsonType": "string"
                },
                "ubicacion": {
                    "bsonType": "object",
                    "properties": {
                        "ciudad": {
                            "bsonType": "string"
                        },
                        "paris": {
                            "bsonType": "string"
                        },
                        "latitud": {
                            "bsonType": "decimal"
                        },
                        "longitud": {
                            "bsonType": "decimal"
                        }
                    }
                },
                "precio": {
                    "bsonType": "decimal"
                },
                "calificacion": {
                    "bsonType": "decimal"
                },
                "opiniones": {
                    "bsonType": "array",
                    "items": {
                        "bsonType": "object",
                        "properties": {
                            "autor": {
                                "bsonType": "string"
                            },
                            "titulo": {
                                "bsonType": "string"
                            },
                            "calificacion": {
                                "bsonType": "int"
                            },
                            "comentario": {
```

```
                            "bsonType": "string"
                        }
                    }
                }
            }
        },
        "required": ["_id", "nombre", "ubicacion", "precio"]
    }
  }
})
```

Pregunta 8 (2 p.).
Indicar los patrones de modelado de datos utilizados para el documento que representa el detalle por hotel.

Embedded document pattern
- El campo **ubicacion** es un documento embebido que contiene los datos de la ubicación del hotel.
- El campo **opiniones** es un arreglo de documentos embebidos que contiene los datos de las opiniones de los usuarios.

Subset pattern (documento embebido)
- El campo **opiniones** muestra solo un conjunto de todas las opiniones que se han realizado al hotel. Mostrando solo las más relevantes.

Caso 4: Inspecciones

Pregunta 9 (2 p.).
Escribir una consulta que permita mostrar la cantidad de inspecciones que no tuvieron un resultado favorable.

```
db.inspecciones.countDocuments({ result: "Fail" })
```

Pregunta 10 (2 p.).
Escribir una consulta que permita mostrar la cantidad de inspecciones realizadas por cada ciudad.

```
db.inspecciones.aggregate([
    {
        $group: {
            _id: "$address.city",
            cantidad: { $count: {} }
        }
    }
])
```