Integrantes

Grupo 20  
Cesar Leonel Chamale Sican - 201700634  
Bryan Eduardo Méndez Quevedo - 201801528  
Oscar Daniel Oliva España – 201902663

Sistemas de bases de datos

Proyecto Fase 2

# Proceso de Carga

# Proceso para la creación de los archivos

Se crearon archivos con formato csv, en base a los resultados de las consultas hechas en SQL Server.

Se hizo una colección por cada una de las siguientes consultas:

**PLATFORMDETAIL**

SELECT DISTINCT

p.id AS PlatformID,

p.name AS PlatformName,

p.summary AS PlatformSummary,

p.alternative\_name as PlatformAlternativeName,

p.generation as PlatformGeneration,

pf.name as PlatformFamily,

pe.name as PlatformType,

(

select DISTINCT '[' + STRING\_AGG( '"'+ platform\_version.os+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformOS,

(

select DISTINCT '[' + STRING\_AGG( '"'+ platform\_version.memory+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformMemory,

(

select DISTINCT '[' + STRING\_AGG( '"'+ platform\_version.graphics+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as Platformgraphics,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.online+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as Platformonline,

(

select DISTINCT '[' + STRING\_AGG( '"'+ platform\_version.resolutions+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformRes,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.resolutions+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformRes,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.cpu+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformCpu,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.storage+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformStorage,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.sound+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformSound,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.Connectivity+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformConnectivity,

(

select DISTINCT '[' + STRING\_AGG( '"'+platform\_version.name+'"' , ', ') +']'

FROM PLATFORM

LEFT JOIN version ON PLATFORM.id = version.id\_platform

LEFT JOIN platform\_version ON version.id\_platform\_version = platform\_version.id

WHERE PLATFORM.id = p.id

)as PlatformVersions,

'[' + STRING\_AGG('"'+pvr.human+'"', ', ') +']' AS ReleaseDates,

'[' + STRING\_AGG(

CASE WHEN pvc.developer = 1 THEN pvc.company ELSE NULL END, ', '

)+']' AS Developers,

'[' +STRING\_AGG(

CASE WHEN pvc.manufacturer = 1 THEN pvc.company ELSE NULL END, ', '

)+']' AS Manufacturers,

(

SELECT DISTINCT

'[' + CAST(STRING\_AGG('"'+game.name+'"' , ', ') AS nvarchar(MAX)) +']'

FROM game

WHERE game.id\_game IN (

SELECT DISTINCT top 2800 id\_game

FROM platform\_game

WHERE platform\_game.id = p.id

)

)AS PlatformGames

FROM

platform AS p

LEFT JOIN platform\_enum as pe ON pe.id = p.platform\_enum

LEFT JOIN platform\_family as pf ON pf.id = p.platform\_family

LEFT JOIN version AS v ON p.id = v.id\_platform

LEFT JOIN platform\_version AS pv ON v.id\_platform\_version = pv.id

LEFT JOIN platform\_releases AS pr ON pv.id = pr.id\_Platform\_version

LEFT JOIN platform\_version\_releasedate AS pvr ON pr.id\_Platform\_version\_releasedate = pvr.id

LEFT JOIN Platform\_company as pc on pc.id\_Platform\_version = pv.id

LEFT JOIN platform\_version\_company AS pvc ON pc.id\_Platform\_version\_company = pvc.id

GROUP BY p.id, p.name, p.summary, p.alternative\_name,p.generation, pf.name, pe.name

;

**RAITINGGAME**

CREATE VIEW Top100Games AS

SELECT

g.id\_game as idgame,

g.name AS nombre,

(

SELECT ('['+STRING\_AGG('"'+p.name+'"', ', ')+']' )

FROM platform\_game AS pg

INNER JOIN platform AS p ON pg.id = p.id

WHERE pg.id\_game = g.id\_game

) AS plataformas,

g.rating AS rating,

g.aggregated\_rating AS valoracion,

(

SELECT '['+STRING\_AGG('"'+ge.name+'"', ', ')+']'

FROM genres\_game AS gg

INNER JOIN genre AS ge ON gg.id = ge.id

WHERE gg.id\_game = g.id\_game

) AS generos

FROM game AS g

ORDER BY g.rating DESC, g.aggregated\_rating DESC;

**LanguageSupportGame**

WITH GameLanguageDetails AS (

SELECT

ls.id\_game,

l.name AS LanguageName,

ls.language\_support\_type,

CASE

WHEN ls.language\_support\_type = 1 THEN 'Audio'

WHEN ls.language\_support\_type = 2 THEN 'Subtitles'

WHEN ls.language\_support\_type = 3 THEN 'Interface'

END AS SupportType

FROM

language\_support ls

JOIN language l ON ls.id\_language = l.id

),

ConsolidatedLanguageDetails AS (

SELECT

gld.id\_game,

gld.LanguageName,

-- Preparando el string en un formato que puede ser convertido en JSON

CONCAT('[',

STRING\_AGG(

CONCAT(

'{"supportType":"',

gld.SupportType,

'"}'

),

','

) WITHIN GROUP (ORDER BY gld.SupportType),

']') AS LanguageSupportJSON

FROM

GameLanguageDetails gld

GROUP BY

gld.id\_game,

gld.LanguageName

),

FinalLanguageDetails AS (

SELECT

id\_game,

-- Concatenando todos los objetos JSON de idiomas en un array JSON

CONCAT(

'[',

STRING\_AGG(

CONCAT(

'{"languageName":"',

LanguageName,

'", "supports":',

LanguageSupportJSON,

'}'

),

','

) WITHIN GROUP (ORDER BY LanguageName),

']'

) AS LanguageSupportDetails

FROM

ConsolidatedLanguageDetails

GROUP BY

id\_game

),

GameRatings AS (

SELECT

g.id\_game,

g.name,

g.rating,

ROW\_NUMBER() OVER(ORDER BY g.rating DESC, g.name) as rn -- Agregamos numeración basada en la calificación

FROM

game g

)

-- Consulta final para obtener los resultados deseados

SELECT

gr.id\_game,

gr.name,

gr.rating,

fld.LanguageSupportDetails

FROM

GameRatings gr

JOIN FinalLanguageDetails fld ON gr.id\_game = fld.id\_game

WHERE

gr.rn <= 300000 -- Filtramos para mantener solo el top 10

ORDER BY

gr.rating DESC, gr.name;

**GenresViewPlatform**

WITH GenreData AS (

SELECT

g.id\_game,

g.name AS game\_name,

'[' + STRING\_AGG(

CONCAT(

'{"id\_genre": ',

CAST(gen.id AS NVARCHAR(MAX)),

', "genre": "',

gen.name,

'"}'

),

', '

) + ']' AS aggregated\_genres

FROM

game g

JOIN

genres\_game gg ON g.id\_game = gg.id\_game

JOIN

genre gen ON gg.id = gen.id

GROUP BY

g.id\_game, g.name

),

-- CTE para los datos de plataforma

PlatformData AS (

SELECT

g.id\_game,

g.name AS game\_name,

'[' + STRING\_AGG(

CONCAT(

'{"id\_platform": ',

CAST(pl.id AS NVARCHAR(MAX)), -- Convertir a texto para la concatenación

', "platform": "',

pl.name,

'"}'

),

', '

) + ']' AS aggregated\_platforms

FROM

game g

JOIN

platform\_game plg ON g.id\_game = plg.id\_game

JOIN

platform pl ON plg.id = pl.id

GROUP BY

g.id\_game, g.name

),

-- CTE para los datos de calificación máxima

MaxRatingData AS (

SELECT

g.id\_game,

MAX(g.rating) AS max\_rating

FROM

game g

GROUP BY

g.id\_game

)

-- Consulta principal para combinar resultados

SELECT

gd.id\_game AS GameID,

gd.game\_name AS GameName,

gd.aggregated\_genres AS Genres,

pd.aggregated\_platforms AS Platforms,

mrd.max\_rating AS MaxRating

FROM

GenreData gd

JOIN

PlatformData pd ON gd.id\_game = pd.id\_game

JOIN

MaxRatingData mrd ON gd.id\_game = mrd.id\_game

ORDER BY

mrd.max\_rating DESC;

**ULTIMA**

CREATE PROCEDURE AMONGO2

@Inicio INT,

@Fin INT

AS

BEGIN

WITH Subquery1 AS (

SELECT DISTINCT

g.name AS JUEGO,

g.first\_release\_date AS Primera\_Fecha\_Lanzamiento,

g.summary AS Descripcion,

g.rating AS Puntaje,

g.aggregated\_rating AS Puntaje\_de\_Critica,

g.local\_rating AS Puntaje\_Total,

g.storyline AS Sinopsis,

g.id\_game AS ID\_Juego

FROM game AS g

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY

g.name, g.first\_release\_date, g.summary, g.rating, g.aggregated\_rating, g.local\_rating, g.storyline, g.id\_game

),

Subquery2 AS (

SELECT DISTINCT

g.id\_game,

'['+ STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(genre.id AS NVARCHAR(MAX))+',',

' "Name": "'+genre.name+'",',

' "Url": "'+genre.url+'"}'

)

, ',')+']' AS Genero

FROM game AS g

LEFT JOIN genres\_game AS gg ON gg.id\_game = g.id\_game

LEFT JOIN genre ON gg.id = genre.id

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery3 AS (

SELECT DISTINCT

g.id\_game,

'['+ STRING\_AGG('{'+

CONCAT(

'"Id":'+CAST(p.id AS NVARCHAR(MAX))+',',

' "Name": "'+p.name+'",',

' "Abbreviation": "'+p.abbreviation+'",',

' "Alternative\_Name": "'+p.alternative\_name+'",',

' "Generation": "'+p.generation+'",',

' "Summary": "'+p.summary+'",',

' "Url": "'+p.url+'"}'

)

, ',') +']' AS Plataforma

FROM game AS g

LEFT JOIN platform\_game AS pg ON pg.id\_game = g.id\_game

LEFT JOIN platform AS p ON p.id = pg.id

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery4 AS (

SELECT

g.id\_game,

'['+STRING\_AGG(

ISNULL(

CONCAT(

ISNULL('{"Plataforma":"'+p.name+'",','{'),

ISNULL('"Fecha":"'+rd.human+'",',''),

ISNULL('"Region":'+

CASE

WHEN rd.region= 1 THEN '"europe"}'

WHEN rd.region = 2 THEN '"north\_america"}'

WHEN rd.region = 3 THEN '"australia"}'

WHEN rd.region = 4 THEN '"new\_zealand"}'

WHEN rd.region = 5 THEN '"japan"}'

WHEN rd.region = 6 THEN '"china"}'

WHEN rd.region = 7 THEN '"asia"}'

WHEN rd.region = 8 THEN '"worldwide"}'

WHEN rd.region = 9 THEN '"korea"}'

WHEN rd.region = 10 THEN '"brazil"}'

END

,'}')

)

,'')

,',')+']' AS Release

FROM game AS g

LEFT JOIN release\_date AS rd ON rd.id\_game = g.id\_game

LEFT JOIN platform AS p ON p.id = rd.platform

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery5 AS (

SELECT DISTINCT

g.id\_game,

(

SELECT

'['+ STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(c.id AS NVARCHAR(MAX))+',',

' "Name": "'+c.name+'",',

' "Country": "'+c.country+'",',

' "Summary": "'+c.description+'",',

' "Start\_Date": "'+CAST(c.start\_date AS NVARCHAR(MAX))+'"}'

)

, ', ') + ']' AS Developer

FROM involved\_company AS ic

LEFT JOIN company AS c ON c.id = ic.company

WHERE g.id\_game = ic.game

AND ic.developer = 'true'

) AS Developer,

(

SELECT '['+ STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(c.id AS NVARCHAR(MAX))+',',

' "Name": "'+c.name+'",',

' "Country": "'+c.country+'",',

' "Summary": "'+c.description+'",',

' "Start\_Date": "'+CAST(c.start\_date AS NVARCHAR(MAX))+'"}'

)

, ', ') + ']' AS Publisher

FROM involved\_company AS ic

LEFT JOIN company AS c ON c.id = ic.company

WHERE g.id\_game = ic.game

AND ic.publisher = 'true'

) AS Publisher,

(

SELECT '['+ STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(c.id AS NVARCHAR(MAX))+',',

' "Name": "'+c.name+'",',

' "Country": "'+c.country+'",',

' "Summary": "'+c.description+'",',

' "Start\_Date": "'+CAST(c.start\_date AS NVARCHAR(MAX))+'"}'

)

, ', ') + ']' AS Porting

FROM involved\_company AS ic

LEFT JOIN company AS c ON c.id = ic.company

WHERE g.id\_game = ic.game

AND ic.porting = 'true'

) AS Porting,

(

SELECT '['+ STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(c.id AS NVARCHAR(MAX))+',',

' "Name": "'+c.name+'",',

' "Country": "'+c.country+'",',

' "Summary": "'+c.description+'",',

' "Start\_Date": "'+CAST(c.start\_date AS NVARCHAR(MAX))+'"}'

)

, ', ') + ']' AS Supporting

FROM involved\_company AS ic

LEFT JOIN company AS c ON c.id = ic.company

WHERE g.id\_game = ic.game

AND ic.supporting = 'true'

) AS Supporting

FROM game AS g

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

),

Subquery6 AS (

SELECT DISTINCT

g.id\_game,

'['+ STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(gm.id AS NVARCHAR(MAX))+',',

' "Name": "'+gm.name+'",',

' "Url": "'+gm.url+'"}'

)

, ', ')+']' AS Modo\_Juego

FROM game AS g

LEFT JOIN game\_mode\_game AS gmg ON gmg.id\_game = g.id\_game

LEFT JOIN game\_mode AS gm ON gm.id = gmg.id

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery7 AS (

SELECT DISTINCT

g.id\_game,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(t.id AS NVARCHAR(MAX))+',',

' "Name": "'+t.name+'",',

' "Url": "'+t.url+'"}'

)

, ', ')+']' AS Theme

FROM game AS g

LEFT JOIN themes\_game AS tg ON tg.id\_game = g.id\_game

LEFT JOIN theme AS t ON t.id = tg.id

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery8 AS (

SELECT DISTINCT

g.id\_game,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(c.id AS NVARCHAR(MAX))+',',

' "Name": "'+c.name+'",',

' "Url": "'+c.url+'"}'

)

, ', ')+']' AS Series

FROM game AS g

LEFT JOIN collections AS c ON c.id= g.collections

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery9 AS (

SELECT DISTINCT

g.id\_game,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(pp.id AS NVARCHAR(MAX))+',',

' "Name": "'+pp.name+'",',

' "Url": "'+pp.url+'"}'

)

, ', ')+']' AS Player\_Perspective

FROM game AS g

LEFT JOIN player\_perspective\_game AS ppg ON ppg.id\_game= g.id\_game

LEFT JOIN player\_perspective AS pp ON pp.id = ppg.id

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery10 AS (

SELECT DISTINCT

g.id\_game,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(f.id AS NVARCHAR(MAX))+',',

' "Name": "'+f.name+'",',

' "Url": "'+f.url+'"}'

)

, ', ')+']' AS Franchise

FROM game AS g

LEFT JOIN franchise\_game AS fg ON fg.id\_game= g.id\_game

LEFT JOIN franchise AS f ON f.id = fg.id

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery11 AS (

SELECT DISTINCT

g.id\_game,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(r.id AS NVARCHAR(MAX))+',',

' "Name": "'+r.nombre+'",',

' "Identifier": "'+r.identifier+'"}'

)

, ',')+']' AS Region,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(gl.id AS NVARCHAR(MAX))+',',

' "Name": "'+gl.name+'"}'

)

, ',')+']' AS Localized\_Titles

FROM game AS g

LEFT JOIN game\_localization AS gl ON gl.id\_game = g.id\_game

LEFT JOIN region AS r ON r.id = gl.id\_region

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

Subquery12 AS (

SELECT DISTINCT

g.id\_game,

'['+STRING\_AGG(

CONCAT(

'{"Id":'+ CAST(an.id AS NVARCHAR(MAX))+',',

'"Comment": "'+an.comment+ '",',

'"Alternative\_Name":"'+an.name+'"}'

)

, ', ')+']' AS XAlternative\_name

FROM game AS g

LEFT JOIN alternative\_name AS an ON an.id\_game = g.id\_game

WHERE g.id\_game>= @Inicio

AND g.id\_game<= @Fin

GROUP BY g.id\_game

),

MainQuery AS (

SELECT DISTINCT

S1.JUEGO,

S1.Primera\_Fecha\_Lanzamiento,

S2.Genero,

S3.Plataforma,

S1.Descripcion,

S1.Puntaje,

S1.Puntaje\_de\_Critica,

S1.Puntaje\_Total,

S1.Sinopsis,

S1.ID\_Juego,

S4.Release,

S5.Developer,

S5.Publisher,

S5.Porting,

S5.Supporting,

S6.Modo\_Juego,

S7.Theme,

S8.Series,

S10.Franchise,

S9.Player\_Perspective,

S11.Region,

S11.Localized\_Titles,

S12.XAlternative\_name

FROM Subquery1 S1

LEFT JOIN Subquery2 S2 ON S1.ID\_Juego = S2.id\_game

LEFT JOIN Subquery3 S3 ON S3.id\_game= S1.ID\_Juego

LEFT JOIN Subquery4 S4 ON S4.id\_game = S1.ID\_Juego

LEFT JOIN Subquery5 S5 ON S5.id\_game = S1.ID\_Juego

LEFT JOIN Subquery6 S6 ON S6.id\_game = S1.ID\_Juego

LEFT JOIN Subquery7 S7 ON S7.id\_game = S1.ID\_Juego

LEFT JOIN Subquery8 S8 ON S8.id\_game = S1.ID\_Juego

LEFT JOIN Subquery9 S9 ON S9.id\_game = S1.ID\_Juego

LEFT JOIN Subquery10 S10 ON S10.id\_game = S1.ID\_Juego

LEFT JOIN Subquery11 S11 ON S11.id\_game = S1.ID\_Juego

LEFT JOIN Subquery12 S12 ON S12.id\_game = S1.ID\_Juego

)

SELECT DISTINCT

g.JUEGO,

g.Primera\_Fecha\_Lanzamiento,

g.Genero,

g.Plataforma,

g.Descripcion,

g.Puntaje,

g.Puntaje\_de\_Critica,

g.Puntaje\_Total,

g.Sinopsis,

g.ID\_Juego,

g.Release,

g.Developer,

g.Publisher,

g.Porting,

g.Supporting,

g.Modo\_Juego,

g.Genero,

g.Theme,

g.Series,

g.Franchise,

g.Player\_Perspective,

g.Region,

g.Localized\_Titles,

g.XAlternative\_name

FROM MainQuery g

ORDER BY g.ID\_Juego

END;

# Proceso de Carga

# Creación de archivos

Se ejecuta la consulta y retorna un resultado en la consola

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se realiza clic derecho al resultado y se selecciona la opción “Save Result As.”, con el que posteriormente lo guardaremos en el lugarque querramos, en formato “csv”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# Inserción de datos

Se crea una base de datos, en este caso se llamará “gamestwitch”



Luego de creada la base de datos, se crean las colecciones, en este caso se llamrá “genreViewPlatform”

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el IDE de mongoDB, Mongo Compass, se selecciona la opción “Import Data” , y seleccionamos el archivo “csv” que queramos ingresar.

\* Cabe resaltar que al csv se le deben de agregar los encabezados

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El propio IDE intentará, automáticamente, agregar los tipos de dato a los valores, en base a los datos del resultado de la consulta.

Luego se selecciona “import” para que se empiecen a ingresar los datos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Al finalizar de ingresar los datos, los mismos se mostrarán ya ingresados de la siguiente manera

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para poder verlos en formato json, se deberá de presionar el botón del lado derecho con la figura de llaves abiertas y cerradas “{}”

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Modelos No-SQL

## PlatformDetails

Texto, Carta

Descripción generada automáticamente

## GenreViewPlatform

Diagrama

Descripción generada automáticamente

## LanguageSupportGame

Diagrama

Descripción generada automáticamente

RatingGame

Diagrama

Descripción generada automáticamente

## Ultima



# Consultas

## Consulta General - Información de Juego



db.mainViewGame.find({JUEGO: { $regex: /God of/, $options: 'i' } })

## Consulta 1 – Top 100 por rating

db.ratingGame.aggregate([

{

$sort: {

valoracion: -1

}

},

{

$limit: 100

},

{

$project: {

\_id: 0,

rating: 1,

nombre: 1,

generos: 1,

plataformas: 1,

valoracion: 1

}

}

])

Texto

Descripción generada automáticamente

## Consulta 2 – Buscar juego por nombre

Texto

Descripción generada automáticamente

db..find({Game: { $regex: / name /, $options: 'i' } })

## Consulta 3 - Información de Juego Agrupado por Plataforma

Texto

Descripción generada automáticamente

db.ultima.aggregate([

{

$match: {

Game: {$regex: /God of/, $options: 'i' }

}

},

{

$group: {

\_id: "$Platform",

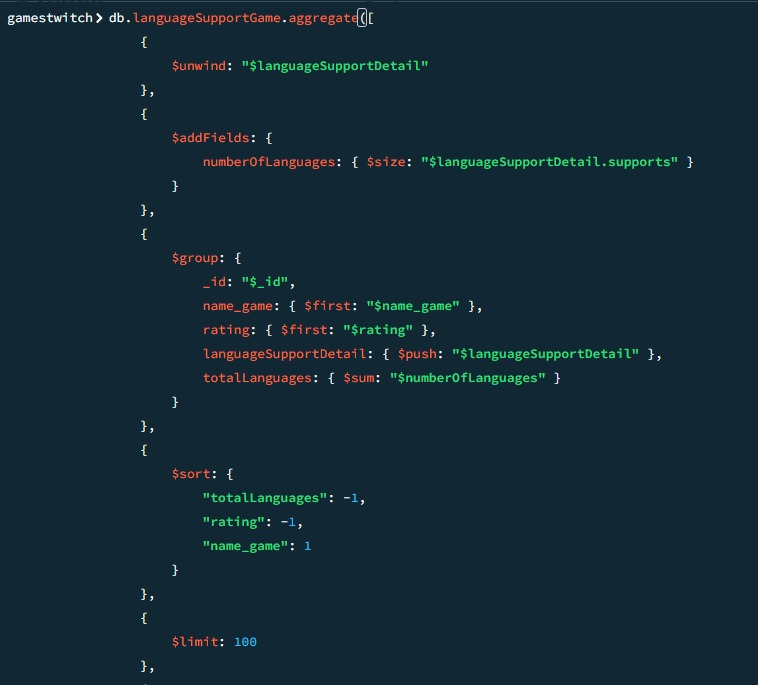
count: { $sum: 1 }

}

}

])

## Consulta 4 – Conteo de soportes de lenguaje



db.languageSupportGame.aggregate([

{

$unwind: "$languageSupportDetail"

},

{

$addFields: {

numberOfLanguages: { $size: "$languageSupportDetail.supports" }

}

},

{

$group: {

\_id: "$\_id",

name\_game: { $first: "$name\_game" },

rating: { $first: "$rating" },

languageSupportDetail: { $push: "$languageSupportDetail" },

totalLanguages: { $sum: "$numberOfLanguages" }

}

},

{

$sort: {

"totalLanguages": -1,

"rating": -1,

"name\_game": 1

}

},

{

$limit: 100

},

{

$project: {

\_id: 1,

name\_game: 1,

rating: 1,

supportedLanguages: "$languageSupportDetail.languageName",

supports: "$languageSupportDetail.supports",

totalLanguages: 1

}

}

])

## Consulta 5 – Top Juegos por género y plataformas ordenados por rating



db.genreViewPlatform.aggregate([

{

$sort: {

rating: -1

}

},

{

$limit: 100

},

{

$project: {

\_id: 0, // Excluir el campo \_id si no lo necesitas

rating: 1, // Mantener el campo valorNumerico

game\_id: 1,

game\_name: 1, // Incluir campo1

genre\_list: 1, // Incluir campo2

platform\_list: 1, // Incluir campo3

}

}

])

## Consulta Plataforma – Plataforma

Texto

Descripción generada automáticamente

db.platformDetail.find({PlatformName: {$regex: /Xbox/, $options: ‘I’} })

# Base de Datos SQL

## Troncamiento de bitácora

Se hace un troncamiento a la bitácora, para desechar la data, o el cache, que ya no nos sirve, para liberar espacio.

Ya que la base de datos está en la nube, específicamente en Amazon RDS, no se pudieron hacer Backups o troncamientos, ya que se necesita la dirección física, pero sí se pudieron realizar snapshots, que es una copia de seguridad de la instancia de la base de datos, con todos sus datos, incluidos sus configuraciones, al momento de hacer el mismo.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

## Collation

Los collation son las reglas de cómo se comparan y determinan los datos de cada texto, por lo que solo los atributos que sean de tipo “STRING” tendrán este campo con alguna definición.

Se utilizó el siguiente comando, con cada tabla, para saber cuál era el collation de cada uno.

SELECT TABLE\_NAME, COLUMN\_NAME, COLLATION\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = ‘game’

Tabla

Descripción generada automáticamente

## Fragmentación de Ïndices

Se mostrará lo fragmentado que están los índices de las tablas, junto con cuales son sus índices.

Se utilizó el siguiente comando para ver el estado de los mismos, en cada tabla.

SELECT

OBJECT\_NAME(ips.[object\_id]) AS TableName,

i.name AS IndexName,

i.type\_desc AS IndexDescription,

STUFF((SELECT ', ' + c.name

FROM sys.index\_columns ic

INNER JOIN sys.columns c ON ic.object\_id = c.object\_id AND ic.column\_id = c.column\_id

WHERE ic.object\_id = ips.[object\_id] AND ic.index\_id = i.index\_id -- Corregido aquí

FOR XML PATH('')), 1, 2, '') AS IndexKeys,

ps.avg\_fragmentation\_in\_percent

FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, NULL) ps

INNER JOIN sys.indexes i ON ps.[object\_id] = i.[object\_id] AND ps.index\_id = i.index\_id

INNER JOIN sys.objects ips ON ps.[object\_id] = ips.[object\_id]

WHERE ips.type = 'U';

Una captura de pantalla de una computadora

Descripción generada automáticamente

## Reducción de fragmentación

Se utilizó el siguiente comando para disminuir la fragmentación de los índices de las tablas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente