Informationsarkitektur Databasutveckling IMDB

Kursuppgift inom	Informationsarkitektur och databasutveckling
Författare	Jacob Hedén
Utbildning	Systemutvecklare.net
Lärare	Marcus Näslund
Jönköping	2022-04-07

Innehållsförteckning

Inledning

Hur man med hjälp av en SQL databas tar fram intressant statistik från IMDB.

Syfte och frågeställning

Syftet med den här rapporten är att utifrån en större datamängd, lägga in i tabeller och skapa en relationell databas samt analysera data och svara på dessa frågor:

1. Flest produktioner utifrån Gener/år från 1989–2021?
2. Populäraste titlarna från de åren?
3. Topp fem mest regissörerna under åren 1989–2021?
4. Topp 10 regissörer med högst Genomsnitt betyg?
5. Hur ser säkerheten och prestandan ut för databasen och kan det förbättras?

Det viktigaste med uppgiften är att lära sig hur en relationsdatabas fungerar, lära sig syntax samt förstå
Hur och på vilket sätt frågorna besvarades på är viktigt och det finns väldigt bra dokumenterat i bifogat material eller här på github[1].

Litteraturöversikt

Utgår från informationen i kursen Informationsarkitektur och databasutveckling

- Föreläsningar
- Föreläsningsanteckningar
- Länkar som gets för mer information inom ämnena
- Annan information som hittats på nätet.

Metod- och källdiskussion

Metoden är en laboration. Källorna är av hög kvalité det kan förekomma en viss diskussion över hur man ska beskriva relationen mellan tabellerna. Där en del är kritiska till just venn-diagram. Ett annat alternativ skulle kunna vara ”join-diagram” som beskrives i den här artikeln (Martinson, 2022).

Tillvägagångsätt

1. Ladda ner TSV filer från IMDB[2].
2. Ladda in datan i tabeller, instruktioner finns i bifogade filer mappen Read from TSV file into tables. Där jag valde att filtrera bort data som är IsAdult på grund av att jag inte anser att den data är relevant för mitt syfte samt fick tänka till lite hur man kan läsa in och lägga till data med en ”Constraint” som säkerhet.

[1] https://github.com/jhdev3/IMDB_LEARN_SQL
[2] https://datasets.imdbws.com/ (20022-03-28)

3. Har man följt samma upplägg bör man ha ett databasdiagram som ser ut som bilaga 1[3].
4. Sedan togs resultat fram genom förfrågningar till databasen som kallas Query, de kan hittas i mappen Resultat Queries.

Bakgrund

SQL

Är ett standardiserat programmeringsspråk för att spara, hämta och modifiera data i en relationsdatabas (SQLWiki, 2022). En relationsdatabas består av tabeller. som binds samman av relationer och restriktioner. Det ger en strukturerad mängd data och kallas relationsdatabas.

För att få en lättadministrerbar databas har sammanhörande uppgifter en relation. Relationsförhållandena kan skapas mellan två eller flera tabeller med ett gemensamt nyckelattribut genom normalisering. Förhållandet kan vara ”one to one”, ”one to many” och ”many to many”.

Att framställa de bästa relationerna och de korrekta förhållandena dem emellan är en utmaning vid skapandet av databasen, framför allt eftersom det är kostsamt att ändra på förhållandena när databasen tagits i drift (Relationsdatabas, 2022).

Nycklar(keys)

För att normalisera en databas används nycklar som kallas ”Primary key” förkortas PK och ”Foreign key” förkortas FK.
De sätter olika begräsningar på tabeller för att databasen ska veta vad som gäller. PK måste vara en unik nyckel som identifierar en rad i en tabell och den får inte var tom (PK-w3school, 2022).
FK sätter en begräsning på en tabell som förhindrar att visa handlingar som utförs inte förstör relationer mellan olika tabeller och den pekar alltid på en PK i en annan tabell (FK-w3school, 2022).

Relationer och Joins

För att enkelt kunna visualisera relationer mellan tabeller gör det att kolla på figur 1 här under och genom att titta på Bilaga 1 går det att förstå vilken typ av relation tabellerna har.

Figur 1

Nyckeln indikerar att det bara finns en rad tabellen och att det är oftast en primary key och oändlighets symbol indikerar en ”many” relation. Figur 1 indikerar en ”One-to-Many” relation och tex en kontakt med 2 nycklar indikerar en ”One to One relation (SQL-Server-2019, 2022).

Sedan är det vanligt att ”Joins” används när data ska hämtas från flera tabeller med en

[3]https://github.com/jhdev3/IMDB_LEARN_SQL/blob/main/SQL/Database_Diagrams/IMDB_DB_relationships.png

relation. Läs mer på w3schools där finns det venn diagram som kan visualisera olika typer av Joins (joins-w3schools, 2022).

Index

Användaren kan inte se index det används för snabba upp sök förfrågningar. När en databas och tabeller blir stora har ett index på en kolumn mycket större effekt. Genom att indikera en primary key för en tabell skapas automatiskt ett index.
Finns många index i en tabell blir prestandan på att lägga till, uppdatera och ta bort data samt att slå ihop tabeller mer kostsamt då även alla index måste uppdateras. (Petrovic, 2022). I Microsoft SQL server finns tre olika typer av Index, heap, Clusterd och nonclustered.

Heap

Heap är en tabell utan ett clustered index. Data som är sparas i heap utan en speciell ordningen. Det går att skapa en eller flera nonclusterd index i tabeller med en heap. (Sql-server-2019, 2022)

Clustered Index

Det kan bara finnas ett Clustered index per tabell och det måste vara på en kolumen med unika värden, det skapas automatiskt ett clustered index på en primary key. Clustered index är samma som ett lexikon där data ordnas efter en alfabetisk ordning eller stigande ordning om Index skulle vara ett nummer (snigdha_yambadwar, 2022).

Non- clustered Index

Non-Clustered Index är väldigt lika som index i en bok. Vilket gör att det är enkelt att slå upp det som man vill läsa om i stället för att bläddra igenom alla sidor i boken för hitta den informationen som man är ute efter (snigdha_yambadwar, 2022).

Figur 2 här hämtad från (snigdha_yambadwar, 2022) och visar skillnaden på Clusterd och non-clustered index.

CLUSTERED INDEX	NON-CLUSTERED INDEX
Clustered index is faster.	Non-clustered index is slower.
Clustered index requires less memory for operations.	Non-Clustered index requires more memory for operations.
In clustered index, index is the main data.	In Non-Clustered index, index is the copy of data.
A table can have only one clustered index.	A table can have multiple non-clustered index.
Clustered index has inherent ability of storing data on the disk.	Non-Clustered index does not have inherent ability of storing data on the disk.
Clustered index store pointers to block not data.	Non-Clustered index store both value and a pointer to actual row that holds data.
In Clustered index leaf nodes are actual data itself.	In Non-Clustered index leaf nodes are not the actual data itself rather they only contains included columns.
In Clustered index, Clustered key defines order of data within table.	In Non-Clustered index, index key defines order of data within index.
A Clustered index is a type of index in which table records are physically reordered to match the index.	A Non-Clustered index is a special type of index in which logical order of index does not match physical stored order of the rows on disk.

Figur 2 Clustered vs Non-Clustered index

SQL syntax

Ordlista samt information om de SQL syntax som jag använt i alla mina queries samt när jag skapar tabeller osv. Finns här på w3schools (w3school-sql, 2022).

Data integritet

Är en kritisk del i att spara data och en viktig del för en relationsdatabas och något en väll uppbygg SQL databas har.
Dataintegritet är motsatsen till datakorruption. Avsikten med all dataintegritetsteknik är densamma: se till att data registreras exakt som avsett. Dessutom, vid senare hämtning, se till att data är samma som när de ursprungligen registrerades. Kort sagt syftar dataintegritet till att förhindra oavsiktliga ändringar av information. Dataintegritet ska inte förväxlas med datasäkerhet vilket disciplinen att skydda data från obehöriga parter.

Alla oavsiktliga ändringar av data som ett resultat av en lagring, hämtning eller bearbetning, inklusive uppsåt, oväntat maskinvarufel och mänskliga fel, är fel i dataintegriteten. Om ändringarna är resultatet av obehörig åtkomst kan det också vara ett fel på datasäkerheten. Beroende på vilken information som är involverad kan detta manifestera sig som tex en enda pixel i en bild har en annan färg än vad den hade när den först sparades, till förlust av semesterbilder eller felaktiga uppgifter om pengar affärskritisk databas, till och med katastrofala förluster av människoliv i ett liv-kritiskt system(Data-integrity-wiki, 2022).

Resultat

Flest produktioner utifrån Gener/år från 1989–2021?

Från det totala resultat som finns som bifogad CSV fil, har det tagits fram lite resultat för att förenkla då CSVs filen har 165 Rader. Ännu mer information hittas i det bifogat material eller här på Github[4].

Hur många år en genre var på TOP 5 listan:

Genre	Antal år TOP 5
Drama	33
Comedy	33
Documentary	33
Action	28
Horror	12
Animation	8
Music	7
Romance	5
Family	3
Crime	2
Thriller	1

Totalt antal produktioner/Genre:

Genre	Totalt antal produktioner
Drama	395339
Documentary	276868
Comedy	273681
Action	60511
Horror	58321
Music	21900
Romance	10869
Animation	8568
Thriller	4597
Family	1782
Crime	1082

[4] https://github.com/jhdev3/IMDB_LEARN_SQL/blob/main/SQL/Resultat%20Queries/Generes%20198 9-2021/SQLQuery_ListPopular_Generes_each_Year.sql

Populäraste titlarna från de åren?

All data finns att hämta i mappen Populäraste titlarna 1989-2021/Resultat Data samt all queries som används finns även där. Data här är uppdelad efter TvSeries och filmer.

De bästa filmerna åren 1989-2021

Noterbart är 2015 var ett dåligt år och fick inte med någon film på top 25 där filter var vart 8e år. Här är ett lite annat tillvägagångsätt än i flest produktioner utifrån Gener/år från 1989–2021. Här förekommer även 2 titlar samma år där de har lika average_rating.

start_year	original_Title	average_Rating	Geners
2021	Jai Bhim	9.4	Crime,Drama,Mystery
1994	The Shawshank Redemption	9.3	Drama
2020	Soorarai Pottru	9.3	Drama
2008	The Dark Knight	9.1	Action,Crime,Drama
1993	Schindler's List	9	Biography,Drama,History
2003	The Lord of the Rings: The Return of the King	9	Action,Adventure,Drama
2016	Dag II	9	Action,Drama,War
2001	The Lord of the Rings: The Fellowship of the Ring	8.9	Action,Adventure,Drama
1999	Fight Club	8.8	Drama
2002	The Lord of the Rings: The Two Towers	8.8	Action,Adventure,Drama
2010	Inception	8.8	Action,Adventure,Sci-Fi
1990	Goodfellas	8.7	Biography,Crime,Drama
2014	Interstellar	8.7	Adventure,Drama,Sci-Fi
1991	Terminator 2: Judgment Day	8.6	Action,Sci-Fi
1991	The Silence of the Lambs	8.6	Crime,Drama,Thriller
1995	Se7en	8.6	Crime,Drama,Mystery
1997	La vita è bella	8.6	Comedy,Drama,Romance
1998	Saving Private Ryan	8.6	Drama,War
2000	Gladiator	8.5	Action,Adventure,Drama
2000	Memento	8.5	Mystery,Thriller
2006	The Departed	8.5	Crime,Drama,Thriller
2006	The Prestige	8.5	Drama,Mystery,Thriller
2009	3 Idiots	8.5	Comedy,Drama
2011	Intouchables	8.5	Biography,Comedy,Drama
2012	Django Unchained	8.5	Drama,Western
2018	Avengers: Infinity War	8.5	Action,Adventure,Sci-Fi
2018	K.G.F: Chapter 1	8.5	Action,Crime,Drama
2019	Gisaengchung	8.5	Comedy,Drama,Thriller
2007	Taare Zameen Par	8.4	Drama,Family
2017	Coco	8.4	Adventure,Animation,Comedy
1989	Indiana Jones and the Last Crusade	8.3	Action,Adventure

1992	Reservoir Dogs	8.3	Crime,Drama,Thriller
2004	Eternal Sunshine of the Spotless Mind	8.3	Drama,Romance,Sci-Fi
2004	Swades: We, the People	8.3	Drama,Musical
2005	Babam ve Oglum	8.3	Drama,Family
2005	Batman Begins	8.3	Action,Crime,Drama
2013	Bhaag Milkha Bhaag	8.3	Biography,Drama,Sport
1996	Eskiya	8.2	Crime,Drama,Thriller
1996	Trainspotting	8.2	Drama

De bästa Tv-serierna åren 1989-2021

Liknande resultat som i bästa filmer men här förekommer dock inte 1991.

start_year	original_Title	average_Rating	Geners
2021	Dhindora	9.8	Comedy,Drama
2020	Scam 1992: The Harshad Mehta Story	9.6	Biography,Crime,Drama
2008	Breaking Bad	9.5	Crime,Drama,Thriller
2002	The Wire	9.3	Crime,Drama,Thriller
2005	Avatar: The Last Airbender	9.3	Action,Adventure,Animation
2011	Game of Thrones	9.3	Action,Adventure,Drama
2011	Leyla ile Mecnun	9.3	Adventure,Comedy,Drama
2019	Kota Factory	9.3	Comedy,Drama
1999	The Sopranos	9.2	Crime,Drama
2013	Rick and Morty	9.2	Adventure,Animation,Comedy
2009	Hagane no renkinjutsushi	9.1	Action,Adventure,Animation
2010	Sherlock	9.1	Crime,Drama,Mystery
1992	Batman: The Animated Series	9	Action,Adventure,Animation
1994	Friends	9	Comedy,Romance
2006	Death Note: Desu nôto	9	Animation,Crime,Drama
1989	Seinfeld	8.9	Comedy
1998	Kaubôi bibappu	8.9	Action,Adventure,Animation
2012	Gravity Falls	8.9	Action,Adventure,Animation
2014	Fargo	8.9	Crime,Drama,Thriller
2014	Last Week Tonight with John Oliver	8.9	Comedy,History,News
2014	True Detective	8.9	Crime,Drama,Mystery
2018	Yellowstone	8.9	Drama,Western
1990	Twin Peaks	8.8	Crime,Drama,Mystery
1997	South Park	8.8	Animation,Comedy
2000	Curb Your Enthusiasm	8.8	Comedy
2001	Six Feet Under	8.8	Comedy,Drama
2003	Chappelle's Show	8.8	Comedy,Music
2004	House M.D.	8.8	Drama,Mystery

2007	Naruto: Shippûden	8.8	Action,Adventure,Animation
2015	Better Call Saul	8.8	Crime,Drama
2015	Narcos	8.8	Biography,Crime,Drama
2015	One Punch Man: Wanpanman	8.8	Action,Animation,Comedy
2016	The Grand Tour	8.8	Comedy,Sport,Talk-Show
2017	Dark	8.8	Crime,Drama,Mystery
1993	The X Files	8.7	Crime,Drama,Mystery
1995	Dragon Ball	8.6	Action,Adventure,Animation
1996	3rd Rock from the Sun	7.8	Comedy,Family,Sci-Fi

Topp fem mest regissörerna under åren 1989–2021?

Fullständig data och queries hittas i mappen TOP 5 Directors flest titlar.

Regissör	Antalet titlar
Johnny Manahan	12 658
Nivedita Basu	12 158
Saibal Banerjee	10 289
Anil v Kumar	8 861
Conrado Lumabas	8 024

Kända för

Regissör	title_type	primary_Title	original_Title
Anil v Kumar	tvSeries	Ganga Kii Dheej	Ganga Kii Dheej
Anil v Kumar	tvSeries	Itna Karo Na Mujhe Pyaar	Itna Karo Na Mujhe Pyaar
Anil v Kumar	tvSeries	Kayamath	Kayamath
Anil v Kumar	tvSeries	Kahiin To Hoga	Kahiin To Hoga
Conrado Lumabas	tvSeries	Saksi	Saksi
Johnny Manahan	movie	Si Aida, si Lorna, o si Fe	Si Aida, si Lorna, o si Fe
Johnny Manahan	movie	Oo na... Mahal na kung mahal	Oo na... Mahal na kung mahal
Johnny Manahan	tvSeries	ASAP	ASAP
Johnny Manahan	movie	Ang TV Movie: The Adarna Adventure	Ang TV Movie: The Adarna Adventure
Nivedita Basu	tvSeries	Kya Hadsaa Kya Haqeeqat	Kya Hadsaa Kya Haqeeqat
Nivedita Basu	tvSeries	24: India	24
Nivedita Basu	tvSeries	Itna Karo Na Mujhe Pyaar	Itna Karo Na Mujhe Pyaar
Nivedita Basu	tvSeries	Karma	Karma
Saibal Banerjee	movie	Sanjhbati	Sanjhbati
Saibal Banerjee	tvSeries	Ichche Nodee	Ichche Nodee
Saibal Banerjee	tvSeries	Fagun Bou	Fagun Bou
Saibal Banerjee	tvSeries	Ishti Kutum	Ishti Kutum

Topp 10 regissörer med högst Genomsnitt betyg?

Alla resultat samt queries finns i bifogat material och i mappen TOP 10 Directors AVG rating.

Regissör	Average_Rating
Matt Shakman	8.55
Michelle MacLaren	8.52
Alik Sakharov	8.48
Jeremy Podeswa	8.45
Alex Graves	8.44
Adam Bernstein	8.43
Ed Bianchi	8.41
Alan Taylor	8.39
Michael Slovis	8.37
Colin Bucksey	8.36

Kända för

Regissör	title_type	primary_Title	original_Title
Adam Bernstein	movie	Six Ways to Sunday	Six Ways to Sunday
Adam Bernstein	tvSeries	Breaking Bad	Breaking Bad
Adam Bernstein	tvSeries	30 Rock	30 Rock
Adam Bernstein	tvSeries	Fargo	Fargo
Alan Taylor	movie	Thor: The Dark World	Thor: The Dark World
Alan Taylor	movie	The Emperor's New Clothes	The Emperor's New Clothes
Alan Taylor	tvSeries	Game of Thrones	Game of Thrones
Alan Taylor	movie	Terminator Genisys	Terminator Genisys
Alex Graves	tvSeries	Proof	Proof
Alex Graves	tvSeries	Game of Thrones	Game of Thrones
Alex Graves	tvSeries	The West Wing	The West Wing
Alex Graves	movie	The Crude Oasis	The Crude Oasis
Alik Sakharov	tvSeries	Ozark	Ozark
Alik Sakharov	tvSeries	Game of Thrones	Game of Thrones
Alik Sakharov	tvSeries	The Sopranos	The Sopranos
Alik Sakharov	tvSeries	House of Cards	House of Cards
Colin Bucksey	tvSeries	Educating Marmalade	Educating Marmalade
Colin Bucksey	tvSeries	Person of Interest	Person of Interest
Colin Bucksey	tvSeries	Breaking Bad	Breaking Bad
Colin Bucksey	tvSeries	Fargo	Fargo
Ed Bianchi	tvSeries	The Wire	The Wire
Ed Bianchi	tvSeries	City on a Hill	City on a Hill
Ed Bianchi	tvSeries	Boardwalk Empire	Boardwalk Empire

Ed Bianchi	tvSeries	Deadwood	Deadwood
Jeremy Podeswa	tvMiniSeries	Station Eleven	Station Eleven
Jeremy Podeswa	tvSeries	Game of Thrones	Game of Thrones
Jeremy Podeswa	tvMiniSeries	The Pacific	The Pacific
Jeremy Podeswa	tvSeries	The Handmaid's Tale	The Handmaid's Tale
Matt Shakman	tvSeries	The Great	The Great
Matt Shakman	tvSeries	Game of Thrones	Game of Thrones
Matt Shakman	tvSeries	It's Always Sunny in Philadelphia	It's Always Sunny in Philadelphia
Matt Shakman	tvMiniSeries	WandaVision	WandaVision
Michael Slovis	tvSeries	Breaking Bad	Breaking Bad
Michael Slovis	movie	Nick and Norah's Infinite Playlist	Nick and Norah's Infinite Playlist
Michael Slovis	tvSeries	CSI: Crime Scene Investigation	CSI: Crime Scene Investigation
Michael Slovis	tvSeries	New Amsterdam	New Amsterdam
Michelle MacLaren	tvSeries	The Walking Dead	The Walking Dead
Michelle MacLaren	tvSeries	Breaking Bad	Breaking Bad
Michelle MacLaren	tvSeries	The X-Files	The X Files
Michelle MacLaren	tvSeries	Game of Thrones	Game of Thrones

Prestandan på databasen och kan det förbättras?

Databas status efter inläsning av TSV

Det här är en Tabell över statusen för databasen efter att alla filer lästs in och det är även så här allt är konfigurerat när alla förfrågningar gjordes till databasen för att fram resultaten.

Databas Storlek

TableName	rows	TotalSpaceMB
InvolvedIn	48 460 118	4635.02
Titles	8 527 053	1028.82
TitlesGenres	13 670 417	772.20
People	11 512 537	620.07
KnownFor	17 770 097	571.95
Professions	12 510 887	541.07
Writers	10 316 463	333.20
Episodes	6 429 733	247.88
Directors	6 509 101	210.32
TitleRatings	1 207 727	40.20

Databas index

Där Index är NULL är det en FK som inte har ett Index får då Heap som typ.

Table_Name	Index_Name	Index_Type	IndexSizeKB
Directors	NULL	HEAP	215 288
Episodes	PK__Episodes__847D171D8D3560DE	CLUSTERED	253 584
InvolvedIn	NULL	HEAP	4 745 776
InvolvedIn	fk_title_Id	NONCLUSTERED	4 745 776
KnownFor	NULL	HEAP	585 440
People	PK__People__543B44E7C771F5DC	CLUSTERED	634 808
Professions	NULL	HEAP	553 984
TitleRatings	PK__TitleRat__1063C5AFD23B8162	CLUSTERED	41 016
Titles	PK_Titles_title_Id	CLUSTERED	1 053 312
TitlesGenres	NULL	HEAP	790 296
TitlesGenres	index_TitlesGenres_FK	NONCLUSTERED	790 296
Writers	NULL	HEAP	341 120

Resultat Prestanda förbättringar

Testar en del queries som jag användt för att hitta förbättringar.
Där de långsammaste Queries är intressanta och vilken effekt Index har. Execution plans finns i mappen Prestanda databas/Exec plans

SQL Fil	Rad numer	Tid(s) Utan index	Tid(s) Med index
ListPopular_Generes_each_Year[5]	5–20	6/Query Totalt 3 min 17s	6/Query Totalt 3 min 17s
InvolvedIn_Create_Insert.sql[6]	40-46	17	1
TOP10_Directors_Rating..sql[7]	65-70	40	35 (TitleId)
TOP10_Directors_Rating..sql	65-70	35(TitleId)	<1 (PersonID och TitleId)
TOP5_Directors_Query.sql[8]	42-47	8	<1

[5] https://github.com/jhdev3/IMDB_LEARN_SQL/blob/main/SQL/Resultat%20Queries/Generes%20198 9-2021/SQLQuery_ListPopular_Generes_each_Year.sql
[6] https://github.com/jhdev3/IMDB_LEARN_SQL/blob/main/SQL/Read%20from%20TSV%20file%20into %20tables/Title_Principals/InvolvedIn_Create_Insert.sql
[7] https://github.com/jhdev3/IMDB_LEARN_SQL/blob/main/SQL/Resultat%20Queries/TOP%2010%20 Directors%20AVG%20rating/TOP10_Directors_Rating..sql
[8] https://github.com/jhdev3/IMDB_LEARN_SQL/blob/main/SQL/Resultat%20Queries/TOP%205%20Di rectors%20flest%20titlar/TOP5_Directors_Query.sql

Diskussion

Resultat frågeställningar 1–4

Inte så spännande resultat, kan dra små slutsatser, kan se att TV-serier generellt har högre medelbetyg än vad filmer har. Det kan bero på att det är vanligt att betyg sätts på en serie efter att man bara sett några avsnitt och inte när man kollat klart på hela serien.
Samt att Dokumentärer är en vanlig genre men inget som får höga betyg.
Samt att de kriterier som jag använder och kallar filter ger relevant och intressant data.

Hur ser säkerheten och prestandan ut för databasen och kan det förbättras

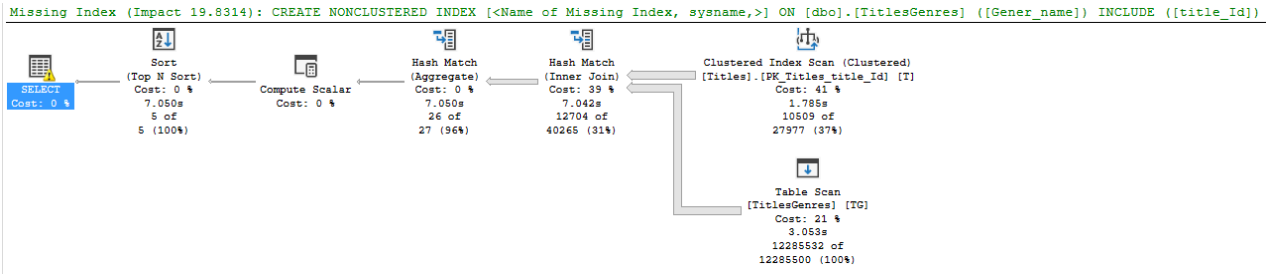
I tabellen Resultat Prestanda förbättringar finns det en hel del intressanta resultat, dels att det inte blev någon förbättring av att sätta genere_name som non-clustered Index, samt att använda sig av den största tabellen involved in för att tydligt visa på hur stor påverkan ett Index har på hastigheten för search queries. Samt inte minst skillnaden mellan TOP10 och TOP5 knownIn queries.
Anledningen att tiden dramatiskt blir snabbare beror på att i stället för att scana tabellen för att matcha Id så slås id upp vilket går mycket snabbare. Det går att se i execution plans, samt att även viss extra sortering inte behöver göras eftersom med hjälp av index vet vilken rad som informationen ligger på.

View

Största skillnaden mellan på tiden TOP5_Directors_Query.sql och TOP10_Directors_Rating..sql utan index är att TOP10 sparar en första Query i en View och TOP5 sparas data i en separat tabell. Det gör att i den separata tabellen bara innehåller 5 rader och att TOP10 är en View som kör sin Select Query varje gång man använder sig av den.
Men med relevanta index så går sökningen så pass snabbt att spara i en extra tabell bara tar onödig plats i databasen samt att det också skapas en onödig tabell. Genom att ha en view garanteras det att man alltid får den senaste data, skulle det läggas till titlar som gör att någon av regissörerna tappar sin plats så krävs det att top5 uppdaterar sin tabell hela tiden medan att ha en view garanterar uppdaterade data.
Det går även att använda views att förenkla SQL queries göra dem mer läsbara och på så sätt också bli lättare att förstå och uppdatera om så krävs.
Men views kan även förbättra säkerheten för databasen tex i detta fall om jag inte tar bort isAdult skulle det gå att skapa en view på title tabellen där man ska vara över 18 år för att se de titlar som är IsAdult.
De kan även användas som extra säkerhet genom att bara visa har tillgång till visa specifika kolumner i en tabell som kan innehålla känsliga data.

Index ingen effekt

ListPopular_Generes_each_Year att sätta index på Gener_name i tabellen TitlesGenres gav ingen effekt på queryn även fast execution planen själv tycker att det skulla vara trevligt och ha en effekt på 20% se figur 3.



Figur 3

Varför databasen inte valde index kan jag bara gissa mig till då jag även rensade cachade executions plans samt tog bort och återskapade index ett antal gånger. Men en gissning här är att skillnaden mot de andra tabellerna är att kolumnen i den här tabellen inte är en foreign key och därför kände inte databasmotorn att söka efter index inte är relevant.
Att göra om TitlesGneres till en tabell med many to many relation skulle då om den då vill använda sig av Index potentiellt göra sökningen snabbare. Det skulle även stärka data integriteten. Tex när en ny titel läggs till med en genre måste då en Genre finnas i tabellen Genre. Samt att skulle man vilja skapa någon Select input för att sortera på Genre eller lägga till en Genre så måste man nu söka igenom och gruppera en tabell som är 13 670 417 rader lång i stället data från en tabell med 28 rader. Samt skapas en ny Genre som tex VRMovies skulle det också bli smidigare samt stärka data integriteten.
Eftersom det inte gav någon effekt blir kostnaden på ungefär 770 mb väldigt stor och tydlig men även den kostnad som tillkommer för att uppdatera ett Index ännu mer kostsam.

Säkerhet och Data integritet

Bara genom att lägga till data från IMDBs TSV filer i en databas så blir den informationen krypterad och skyddad i detta fall är det bara en lokal databas och min dator som har tillgång till de tabellerna. Genom att kontrollera så att allt går rätt till när jag för inte informationen i mina tabeller samt hur de tabellerna är designade och vilka relationer de har skapas en relativ hög data integritet. Skulle det vara så att man vill ha databasen på Azure tex så skulle det gå att stärka säkerheten genom att ge tillgång till databasen för specifika IP-adresser, samt använda sig av Procedures och views för att låta användare med olika behörighet nivåer inte fritt få använda vilka querys de vill, samt begränsa vilka tabeller och kolumner de får tillgång till. Det skulle då även vara enkelt att skydda sig mot SQL-injektions.
I dessa querys är data integriteten väldigt viktig för att resultaten ska bli rätt och det tycker jag att jag har uppnått till en acceptabel nivå, finns lite förbättringar som går att göras.

Bonus

En del extra tankar kring en SQL databas och vad som kan vara viktigt att tänka på när man använder sig av en sådan databas.

Procedurers

Är inget som jag använt mig av eftersom det inte har funnits ett relevant användningsområde som jag nämnt tidigare är det bra i ett säkerhetsperspektiv. Men de skulle även kunna användas för att enklare testa queries och göra koden mer lätt försåtlig. Tex hade jag en tanke att skapa en linegraf med hjälp av Svelte och Chartjs i detta projekt . Där X-axel är årtal och Y axeln är antalet produktioner i en Genre.
För att slippa kontakt med en databas och publicera på Github var tanken att ”hårdkoda” resultaten från Flest produktioner utifrån Gener/år från 1989–2021. Men skulle jag använda mig av en databas och alltid ha de senaste resultaten är det väldigt smidigt att skapa en procedure för att räkna alla produktioner per år utifrån vilken Genre dvs man lägger Genre_namn som en parameter. Det skulle ge mig tillbaka de datasets som skulle behövs för att generar en linje i grafen.
Finns även ett annat väldigt bra användningsområde och det är delete, för en sak som gör att relationsdatabaser har en hög data integritet är att det inte går att ta bort rader från en tabell som är refererad till en annan tabell. I detta fall går det inte att ta bort en titel utan att ta bort title_Id där de förekommer som foreign key i de andra tabellerna först och det skulle en procedure kunna vara bra för och då bara att Admin är den enda som får använda sig av

den proceduren.
Delete på det sättet går även att åstadkomma genom att använda Cascade delete[9].

Cascade delete

Cascade delte är vanligt sätta på tabeller som har Foreign key för att ta bort den raden när raden från förälder tabellen tas bort. Det finns även en variant som förekommer där man sätter FK till NULL istället för att ta bort raden. I denna databas skulle det kunna vara relevant att ha det på TitleRating tabellen för skulle man av misstag ta bort titeln Nyckeln till frihet så är det inte så svårt att lägga tillbaka data i de andra tabellerna genom att googla informationen, men datan i TitleRatings är något som IMDB själva generar, då skulle Numvotes och Avrage rating vara borta mycket svårare att återställa. Men genom att sätta pk till NULL skulle man genom ett visst detektivarbete även där kunna få tillbaka avrage rating och numvotes utan några stora bekymmer. Utgår här att den backup de har på databasen också den blev uppdaterad när titeln nyckeln till frihet togs bort.

Code First

Det vi har gjort här är database first att vi skapar allt själva och har koll på databasen med cascade delete osv. Men i Code first och låta ett ramverk som tex entity[10] skapa alla tabeller och relationer automatiskt utifrån hur man designat sina data klasser. Där skulle default för cascade delete vara set NULL på grund av att tar man bort data så är den borta.
Lika lätt som det är att missa att skydda sig mot SQL-injections skulle det här kunna vara att glömma ta bort rader. Det är i sig inte ett stort problem då minne är billigt. Men i och med GDPR och att man har rätt till att bli glömd så skulle du här med att sätta en FK till NULL i stället för att ta bort hela raden i tabellen. Är att finns det då en kolumn som identifierar personen tex personnummer så finns det kvar där och om du inte rensar tabellen utifrån NULL värden på Foreign key inom en viss tid bryter du därmed mot GDPR.

[9] https://docs.microsoft.com/en-us/ef/core/saving/cascade-delete
[10] https://docs.microsoft.com/en-us/ef/core/

Källförteckning

Data-integrity-wiki. (den 08 04 2022). *wikipedia*. Hämtat från https://en.wikipedia.org/wiki/Data_integrity

FK-w3school. (den 04 04 2022). Hämtat från https://www.w3schools.com/sql/sql_foreignkey.asp

joins-w3schools. (den 09 04 2022). *w3schools*. Hämtat från https://www.w3schools.com/sql/sql_join.asp

Martinson, A. (den 08 04 2022). *towardsdatascience*. Hämtat från https://towardsdatascience.com/you-should-use-this-to-visualize-sql-joins-instead-of-venn-diagrams-ede15f9583fc

Petrovic, B. (den 08 04 2022). *sqlshack.com*. Hämtat från https://www.sqlshack.com/sql-index-overview-and-strategy/

PK-w3school. (den 08 04 2022). Hämtat från https://www.w3schools.com/sql/sql_primarykey.asp

Relationsdatabas. (den 08 04 2022). Hämtat från https://sv.wikipedia.org/wiki/Relationsdatabas

snigdha_yambadwar. (den 08 04 2022). *geeksforgeeks.org*. Hämtat från https://www.geeksforgeeks.org/difference-between-clustered-and-non-clustered-index/

Sql-server-2019. (den 08 04 2022). *Docs.Microsoft*. Hämtat från https://docs.microsoft.com/en-us/sql/relational-databases/indexes/heaps-tables-without-clustered-indexes?view=sql-server-ver15#:~:text=A%20heap%20is%20a%20table,heap%20without%20specifying%20an%20order.

SQL-Server-2019. (den 22 04 2022). *docs.microsoft.com*. Hämtat från https://docs.microsoft.com/en-us/sql/ssms/visual-db-tools/design-database-diagrams-visual-database-tools?view=sql-server-ver15

SQLWiki. (den 08 04 2022). Hämtat från https://sv.wikipedia.org/wiki/Structured_Query_Language

w3school-sql. (den 09 04 2022). Hämtat från https://www.w3schools.com/sql/default.asp

Bilagor

Bilaga 1- Relations diagram databas.