Opgave 1a

I. Dit is op tabel niveau aangezien het meerdere attributen effect en in de rest van de tabel op gecheckt wordt of het uniek is.

II. Dit is op record niveau, als er een nieuwe record wordt toegevoegd dan gaat het eerst checken of de informatie wel klopt, het kijkt niet overal naar de tabel en kijk ook naar meer dan 1 attribuut.

Opgave 1b

1) Ja, voegt een nieuw boek toe zonder problemen.

2) Nee, naam van het boek en de auteur moeten een unieke combinatie zijn, de constraint van eerder geeft aan dat het niet mag.

3) Ja, de ISBN is gelijk aan 1000 en de uitleenperiode is groter dan 0 dus de constraint doet niks.

4) Ja, de ISBN is hoger dan 1000 en de uitleenperiode is groter dan 0 dus de constraint doet niks.

5) Ja, de combinatie van auteur naam en boek naam zijn uniek en wordt niet tegengehouden door de constraint.

6) Nee, de uitleenperiode moet groter dan 0 zijn en de constraint zal deze insert afvangen.

7) Ja, de ISBN is hoger dan 1000 en de uitleenperiode is groter dan 0 dus de constraint doet niks.

8) Ja, De combinatie van auteur naam en boek naam is nog steeds uniek het werkt dus nog steeds.

9) Nee, voorheen werd de insert van exemplaar met ID 1910 afgevangen omdat de uitleenperiode 0 was en bestaat dus niet in de database. Deze kan dus ook niet aangepast worden.

Opgave 1c

Dit ligt op database niveau aangezien de constraint naar meerdere tabellen gekeken moet om te bepalen of hij iets moet afvangen.

Opgave 1d

Lid

INSERT

PRINT

Uitlening

Opgave 1e

Bij het uitvoeren van een DELETE statement wordt alleen de "deleted" tabel gevuld. De "inserted" tabel wordt niet gebruikt bij een DELETE statement omdat er geen nieuwe rijen in de tabel worden toegevoegd, alleen bestaande rijen worden verwijderd.

In dit geval bevat de "deleted" tabel alle 5 records van de Reservering-tabel voordat deze werden verwijderd door de trigger. De "inserted" tabel is leeg omdat er geen nieuwe rijen zijn toegevoegd.

Opgave 2a

USE [Sakila]

CREATE INDEX IX\_RentalRate\_Length ON Film (RentalRate, Length);

Om de query te optimaliseren, kan een samengestelde index worden gemaakt op de kolommen "RentalRate" en "Length" van de "Film" tabel.

Met deze index kan de database efficiënter zoeken naar de rijen die voldoen aan de voorwaarden van de query. Door de index kan de database snel de rijen vinden die een "RentalRate" van 4.99 hebben en een "Length" van 112. Dit kan het uitvoeren van de query aanzienlijk versnellen als er veel rijen zijn in de "Film" tabel.

Het resultaat van de query na het maken van de index zou hetzelfde moeten zijn als voorheen, maar de query zou nu sneller moeten worden uitgevoerd omdat de database de index kan gebruiken om de juiste rijen sneller te vinden.

Opgave 2b

1.

Voor het optimaliseren van query's die enkel actieve producten ophalen, kan een index worden gemaakt op de kolom "Actief". Hierdoor kan de database efficiënter zoeken naar rijen met de waarde "Actief" en worden alleen de rijen opgehaald die voldoen aan de voorwaarde van de query.

Een nadeel van deze optimalisatie kan zijn dat als de tabel regelmatig wordt bijgewerkt, de indexen ook bijgewerkt moeten worden, dit gebruikt veel processing geheugen. Dit kan vooral een probleem zijn als er veel rijen in de tabel zijn en er regelmatig updates worden uitgevoerd op de "Actief" kolom.

2.

Een mogelijke optimalisatie zou kunnen zijn om een aparte tabel te maken voor de producten van een specifieke leverancier en deze bij te werken wanneer er nieuwe producten worden toegevoegd of bestaande producten worden verwijderd of bijgewerkt. Hierdoor kunnen de query's efficiënter worden uitgevoerd zonder dat er een grote hoeveelheid rijen hoeft te worden doorzocht. Dit kan echter wel leiden tot complexiteit in het onderhoud van de database en extra processing geheugen bij het bewerken van de aparte tabel.

3.

Voor het optimaliseren van query's die de gemiddelde brutoprijs per categorie berekenen, kan een index worden gemaakt op de kolommen "Categorie" en "BrutoPrijs". Door deze index kan de database de rijen efficiënter sorteren en groeperen op basis van de categorieën en de brutoprijzen. Dit kan het berekenen van de gemiddelde brutoprijs per categorie versnellen.

Een nadeel van deze optimalisatie kan zijn dat als de tabel regelmatig wordt bijgewerkt, de indexen ook bijgewerkt moeten worden, wat extra overhead kan veroorzaken. Dit kan vooral een probleem zijn als er veel rijen in de tabel zijn en er regelmatig updates worden uitgevoerd op de "Categorie" en "BrutoPrijs" kolommen.

Opgave 3a

Ik de volgende gebruikers en rollen creëren:

1. Gebruiker "Customer": deze gebruiker kan alleen lezen van de database. Hij kan producten bekijken, maar hij heeft geen toegang tot het wijzigen of verwijderen van producten of andere gegevens.
2. Gebruiker "Administrator": deze gebruiker heeft volledige toegang tot alle gegevens in de database. Hij kan producten bekijken, wijzigen, toevoegen en verwijderen, evenals andere gegevens in de database beheren, zoals klantgegevens en bestellingen. Deze gebruiker heeft ook de mogelijkheid om de structuur van de database aan te passen.
3. Rol "Customer": deze rol is gekoppeld aan de gebruiker "Customer" en heeft alleen leesrechten voor de tabellen die relevant zijn voor de klantfunctionaliteit van de applicatie.
4. Rol "Administrator": deze rol is gekoppeld aan de gebruiker "Administrator" en heeft alle rechten op alle tabellen in de database, inclusief wijzigings- en beheerrechten. Deze rol heeft ook de mogelijkheid om de structuur van de database aan te passen.

Door deze inrichting van gebruikers en rollen kan de beveiliging en integriteit van de database worden gewaarborgd. De klanten hebben alleen toegang tot de gegevens die relevant zijn voor hun functionaliteit en kunnen geen ongewenste wijzigingen aanbrengen. De beheerder heeft volledige toegang tot alle gegevens en kan de structuur van de database aanpassen indien nodig.

Opgave 3b

Ja, er kunnen problemen ontstaan bij het uitvoeren van de twee transacties als het isolatieniveau van de database op "Snapshot" is ingesteld. Dit komt omdat "Snapshot" isolatieniveau vereist dat elke transactie wordt uitgevoerd in een consistent beeld van de database zoals het was op het moment dat de transactie werd gestart.

In de eerste transactie wordt eerst de som van de prijzen van de verkopen van een bepaalde account berekend en vervolgens wordt een update op dezelfde account uitgevoerd op basis van de berekende som. Als er op hetzelfde moment een andere transactie plaatsvindt die ook gegevens op de account wijzigt, kan de berekende som niet overeenkomen met de werkelijke som van de verkopen van die account. Dit kan leiden tot een onjuiste beslissing over het al dan niet toekennen van korting aan die account.

In de tweede transactie worden twee nieuwe verkopen ingevoegd voor dezelfde account. Als er op hetzelfde moment een andere transactie plaatsvindt die ook gegevens op de account wijzigt, kan er een conflict ontstaan over de volgorde van de ingevoegde verkopen. Dit kan leiden tot inconsistentie in de gegevens en verlies van gegevens.

Daarom wordt aanbevolen om in een omgeving met het "Snapshot" isolatieniveau, transacties zorgvuldig te ontwerpen en uit te voeren om conflicten en inconsistentie te voorkomen.

Opgave 3c

Optimistic concurrency locking is een aanpak waarbij de database veronderstelt dat er geen conflicten zullen optreden tussen gelijktijdige transacties en dat de transacties gewoon kunnen doorgaan. Als er toch een conflict optreedt, wordt de transactie afgebroken en kan de gebruiker het opnieuw proberen. Dit is de meest geschikte optie in deze situatie, omdat het aantal beschikbare exemplaren beperkt is en er hoogstwaarschijnlijk geen meerdere klanten tegelijkertijd hetzelfde spel zullen kopen.

Opgave 4

Wanneer het Isolation Level is ingesteld op READ UNCOMMITTED, kunnen transacties afwijkingen tonen van het ACID-principe:

* Atomicity (A): Atomiciteit vereist dat een transactie als een enkele, ondeelbare eenheid wordt behandeld. Als een transactie wordt uitgevoerd met het Isolation Level op READ UNCOMMITTED, kunnen onbevestigde wijzigingen van andere transacties worden gelezen, waardoor de atomiciteit van de transactie in gevaar kan komen. Bijvoorbeeld, als een transactie A begint en deel uitmaakt van de transactie ongedaan maakt, maar voordat deze kan worden bevestigd, een transactie B de gewijzigde rijen leest, kan de transactie A niet worden teruggedraaid, wat resulteert in een inconsistentie in de database.
* Consistency (C): Consistentie vereist dat de database zich altijd in een geldige staat bevindt. READ UNCOMMITTED kan inconsistenties veroorzaken omdat het transacties toestaat om onbevestigde wijzigingen van andere transacties te lezen. Bijvoorbeeld, als een transactie B een update uitvoert op een rij die wordt gelezen door een transactie A die op READ UNCOMMITTED draait, kan transactie A onjuiste of onvolledige gegevens lezen, waardoor de consistentie van de database in gevaar komt.
* Isolation (I): Isolatie vereist dat transacties onafhankelijk van elkaar worden uitgevoerd, zonder invloed op elkaar. READ UNCOMMITTED kan isolatieproblemen veroorzaken omdat het toestaat dat transacties onbevestigde wijzigingen van andere transacties lezen. Dit kan leiden tot onverwachte resultaten, zoals racevoorwaarden en dirty reads.
* Durability (D): Duurzaamheid vereist dat wijzigingen die zijn aangebracht door een bevestigde transactie, permanent zijn. READ UNCOMMITTED heeft geen invloed op de duurzaamheid van transacties, aangezien het alleen van invloed is op de leesbewerkingen van transacties en niet op hun schrijfbewerkingen.

Over het algemeen wordt het Isolation Level READ UNCOMMITTED afgeraden in productiesystemen vanwege de risico's van inconsistentie en isolatieproblemen.