به نام خدا

پاسخنامه ی تکلیف هفتم

دی 1400

TRIGGERS

1. می خواهیم متوسط درصد رضایت مشتریان از فیلم ها را در جدول فیلم داشته باشیم .
   a. ستون Customer_Rate (به صورت عددی و کنترل بازه 0 تا 100) را به جدول Rental اضافه نمایید .مقدار پیش فرض آن را برای اطلاعات موجود با 50 پر کنید و همچنین یک ستون به نام Score به جدول Film اضافه نمایید که متوسط رضایت مشتریان آن فیلم را در خود ذخیره کند .

```
ALTER TABLE rental ADD COLUMN Customer_Rate
NUMERIC(3,0) DEFAULT 50 CHECK(Customer_Rate >= 0 and Customer_Rate <= 100)

ALTER TABLE film ADD COLUMN Score NUMERIC
```

   b. تریگری بنویسید که پس از بازپس دادن فیلم و دریافت سطح رضایت مشتری (آپدیت Rental) متوسط رضایت مشتریان مربوط به آن فیلم را آپدیت نماید .

```
CREATE OR REPLACE FUNCTION update_score_FUNC()
RETURNS trigger
LANGUAGE plpgsql
AS
$$
        declare updated_film_id int;
        BEGIN
        select film_id into updated_film_id from rental AS R
        INNER JOIN Inventory AS I ON I.Inventory_Id=R.Inventory_Id
        where R.rental_id = NEW.rental_id;

        update film set score = (Select Round((select sum(customer_rate)/count(customer_rate) from
rental AS R
                INNER JOIN Inventory AS I ON I.Inventory_Id=R.Inventory_Id
                        WHERE I.film_id = updated_film_id),2))
                where film_id = updated_film_id;
        RETURN NEW;
        END;
$$;
```

```
CREATE TRIGGER update_score_TRG
AFTER UPDATE OF Customer_Rate ON rental
FOR EACH ROW
EXECUTE PROCEDURE update_score_FUNC();
```

c. برای تست کردن تریگر خود یک اسکریپت جهت وارد کردن رضایت مشتری بنویسید و تصاویر قبل و بعد وارد کردن آن را در فایل تحویلی قرار دهید .

```
select rental_id,film_id,customer_rate from rental as r
INNER JOIN Inventory AS I ON I.Inventory_Id=R.Inventory_Id
where I.film_id = 1;

select film_id, score from film where film_id = 1;

update rental set customer_rate = 100 where rental_id = 4863;

select film_id, score from film where film_id = 1;
```

2. به سوالات زیر پاسخ دهید :

a. در اکثر DBMS ها محدودیت هایی در اجرای تریگر های تو در تو وجود دارد .به عنوان مثال در پایگاه داده Postgre محدودیت Stack depth limit را داریم . توضیح دهید چرا چنین محدودیت هایی تعیین شده اند ؟

اگر اجرای یک تریگر باعث اجرای همان تریگر یا تریگر دیگر شود میتواند باعث ایجاد لوپ بی نهایت شود و اجرای کوئری هیچ وقت به انتها نرسد . به همین منظور اکثر پایگاه های داده ای معروف یک محدودیت مشخصی برای این موضوع در نظر گرفته اند تا اگر تریگر مورد نظر دچار لوپ شد پس از Limit خاصی اجرای کوئری را متوقف کرده و خطا صادر کند که همان Stack depth limit است .

b. راهکاری برای جلوگیری از اجرای تو در توی تریگر ها بیابید .

با استفاده از فانکشن pg_trigger_depth که یکی از فاکنشن های built in پایگاه داده Postgre است میتوان تعداد مراحلی که تریگر به صورت تو در تو اجرا شده است را بدست آورد و سپس در تریگر مورد نظر آن را با استفاده از شرط (WHEN (pg_trigger_depth() = 0) کنترل نمود .

3. با توجه به سوال قبل می خواهیم یک امکان جدید به سیستم اضافه نماییم :

a. یک trigger بنویسید که **از این به بعد** هر مشتری پس از این که 5 فیلم اجاره کرد، یک فیلم جدید از یکی از دسته بندی فیلم های 5 اجاره آخر (بر اساس rental_date و به صورت تصادفی) به جز فیلم های اجاره شده توسط خود او و از فیلم های ارائه شده در همان فروشگاه به وی اجاره بدهد.(پس از ارائه این خدمت به مشتری شمارش فیلم های اجاره شده او برای خدمت بعدی از سر گرفته شود.)

راهنمایی: دسته بندی 5 فیلم آخری که مشتری اجاره کرده است را بدست بیاورید سپس آیدی Inventory های مربوط به فیلم های 5 دسته بندی آخر همان فروشگاه را بدست آورده و یک

عدد تصادفی بین 0 تا Count آیدی ها تولید نمایید و در ادامه با استفاده از LIMIT و OFFSET یک رکورد را واکشی نمایید.

همچنین برای شمارش تعداد فیلم ها ستونی به نام Rent_Count به جدول customer اضافه کنید.

```
CREATE OR REPLACE FUNCTION Rent_func()
        RETURNS TRIGGER
        LANGUAGE PLPGSQL
        AS
        $$
        DECLARE temp_count int;
        BEGIN

        select Rent_Count into temp_count
        from customer
        where customer_id=NEW.customer_id;

        IF temp_count < 4 THEN
        update customer
        set Rent_Count=Rent_Count+1
        where customer_id=NEW.customer_id;

        ELSIF temp_count = 4 THEN
        with ids as
(select distinct i.inventory_id as id
from film_category as fc inner join inventory as i on fc.film_id=i.film_id
 where i.store_id=(select store_id from staff as s where staff_id=NEW.staff_id ) AND category_id
        in
 (select c.category_id from category as c inner join film_category as fc on
        c.category_id=fc.category_id
        inner join film as f on f.film_id=fc.film_id inner join inventory as i on i.film_id=f.film_id
                inner join rental as r on r.inventory_id=i.inventory_id
                where r.customer_id=NEW.customer_id AND
                i.store_id=(select store_id from staff as s where staff_id=NEW.staff_id )
                order by r.rental_date desc
                                LIMIT 5)
 except select i.inventory_id from inventory as i
 inner join rental as r on r.inventory_id=i.inventory_id
 where r.customer_id=NEW.customer_id)

INSERT INTO public.rental(
```

```
rental_date, inventory_id, customer_id, return_date, staff_id, last_update)
VALUES (NEW.rental_date,
(select id from ids
limit 1 offset (SELECT ROUND(RANDOM() * ((select count(*) from ids) + 1)))),
NEW.customer_id, NEW.return_date, NEW.staff_id, NEW.last_update);

update customer
set Rent_Count=0
where customer_id=NEW.customer_id;
END IF;

RETURN NEW;
END;
$$
```

```
CREATE TRIGGER RentGiftTrg
AFTER INSERT
ON rental
FOR EACH ROW
WHEN (pg_trigger_depth() < 1)
EXECUTE PROCEDURE Rent_func();
```

b. کدی برای تست کردن trigger خود بنویسید و عکس مربوط به جدول rental که نشان دهنده آن است که رکورد به درستی اضافه شده را در پاسخ تحویلی خود قرار دهید.

```
insert into rental(rental_date,inventory_id,customer_id, return_date, staff_id)
values( NOW(), 7, 5,NOW()+INTERVAl '1 WEEK', 1);

insert into rental(rental_date,inventory_id,customer_id, return_date, staff_id)
values( NOW(), 8, 5,NOW()+INTERVAl '1 WEEK', 1);

insert into rental(rental_date,inventory_id,customer_id, return_date, staff_id)
values( NOW(), 9, 5,NOW()+INTERVAl '1 WEEK', 1);

insert into rental(rental_date,inventory_id,customer_id, return_date, staff_id)
values( NOW(), 10, 5,NOW()+INTERVAl '1 WEEK', 1) ;

insert into rental(rental_date,inventory_id,customer_id, return_date, staff_id)
```

values(NOW(), 13, 5,NOW()+INTERVAl '1 WEEK', 1) ;

RANKING & WINDOWING

4. دستوری بنویسید که رتبه فیلم ها را بر اساس کل میزان مبلغ پرداختی کرایه آنها (نزولی)، یک ستون رتبه فیلم بین همه فیلم ها، یک ستون رتبه فیلم به تفکیک rating و مشخص کند که این فیلم از نظر میزان فروشش در چارک اول هست یا خیر (YES or NO) نتیجه را به ترتیب عنوان فیلم(صعودی) نمایش دهد.

خروجی مورد انتظار:

film_title,film_rating,rank_in_all,rank_in_rating,sum_amount,is_in_first_quartile

```
SELECT
    f.title as film_name ,
    f.rating as film_rating,
    rank() over (ORDER BY sum(amount) DESC) as rank_in_all,
    rank() over (PARTITION BY f.rating ORDER BY sum(amount) DESC) as
rank_in_rating,
    sum(amount) as sum_amount,

    (CASE
        WHEN (ntile(4) over (ORDER BY sum(amount) DESC)) = 1 THEN 'YES'
        ELSE 'NO'
        END) as is_in_first_quartile

  FROM payment as p
  JOIN rental as r ON r.rental_id = p.rental_id
  JOIN inventory as i ON i.inventory_id = r.inventory_id
  join film as f ON f.film_id = i.film_id
  GROUP BY f.film_id
  ORDER BY f.title
```

film_name	film_rating	rank_in_all	rank_in_rating	sum_amount	is_in_first_quartile
Academy Dinosaur	PG	718	147	33.79	NO
Ace Goldfinger	G	508	89	52.93	NO
Adaptation Holes	NC-17	699	137	34.89	NO
Affair Prejudice	G	261	47	83.79	NO
African Egg	G	557	99	47.89	NO
Agent Truman	PG	118	24	111.81	YES
Airplane Sierra	PG-13	263	64	82.85	NO
Airport Pollock	R	239	46	86.85	YES
Alabama Devil	PG-13	354	80	71.88	NO
Aladdin Calendar	NC-17	66	18	131.77	YES
Alamo Videotape	G	761	132	30.78	NO

5. می خواهیم گزارشی تحلیلی بنویسیم که در آن، جمع مبلغ پرداختی برای کرایه فیلم ها را در ماه به تفکیک درجه سنی (rating) فیلم ها مشاهده کنیم و در هر رکورد علاوه بر عدد ماه و درجه سنی، مقدار فروش ماه قبل و ماه بعد آن را نیز نیاز داریم نتیجه نهایی به ترتیب ماه صعودی باشد.
(اعداد ماه ها دقیقا پشت سر هم نیستند و لذا طبق رکورد ها ماه بعد از 2 که رکوردی برای آن موجود می باشد ماه 6 است)
(راهنمایی: در تهیه این گزارش تنها از توابع window استفاده نمایید)

```
SELECT
    EXTRACT(MONTH FROM payment_date) as month_number,
    f.rating,
    SUM(amount) as sum_amount,
    LAG(sum(amount),1) OVER (PARTITION by f.rating ORDER BY
EXTRACT(MONTH FROM payment_date)) prev_month_sales,
    LEAD(sum(amount),1) OVER (PARTITION by f.rating ORDER BY
EXTRACT(MONTH FROM payment_date)) next_month_sales


FROM payment as p
JOIN rental as r ON r.rental_id = p.rental_id
JOIN inventory as i ON i.inventory_id = r.inventory_id
JOIN film as f ON f.film_id = i.film_id
GROUP BY month_number,f.rating
ORDER BY month_number;
```

month_number	rating	sum_amount	prev_month_sales	next_month_sales
2	R	1745.78	(Null)	4782.76
2	PG	1658.99	(Null)	4757.52
2	NC-17	1667.89	(Null)	5085.03
2	PG-13	1856.58	(Null)	5316.63
2	G	1422.60	(Null)	3944.62
3	PG-13	5316.63	1856.58	6563.76
3	R	4782.76	1745.78	5461.78
3	NC-17	5085.03	1667.89	5768.44

OLAP

6. با توجه به پایگاه داده DVD Rental :

a. میزان اجاره هر فیلم را به ازای پارامتر های شهر، فروشگاه، کارمند با استفاده از Cube به صورت نزولی بر اساس میزان اجاره به دست بیاورید .

```
select (select title from film where film_id=f.film_id),(select city from city where
city_id=c.city_id),st.store_id,(select first_name||last_name from staff where
staff_id=s.staff_id) as Staff, Count(*) as Rent
from film as f inner join inventory as i on i.film_id=f.film_id
inner join rental as r on r.inventory_id=i.inventory_id
inner join staff as s on s.staff_id=r.staff_id
inner join store as st on st.store_id=s.store_id
inner join address as ad on ad.address_id=st.address_id
inner join city as c on c.city_id=ad.city_id
group by f.film_id,cube(c.city_id,st.store_id,s.staff_id)
order by Count(*) desc
```

b. سعی کنید همین نتیجه را بدون استفاده از Cube بدست بیاورید .

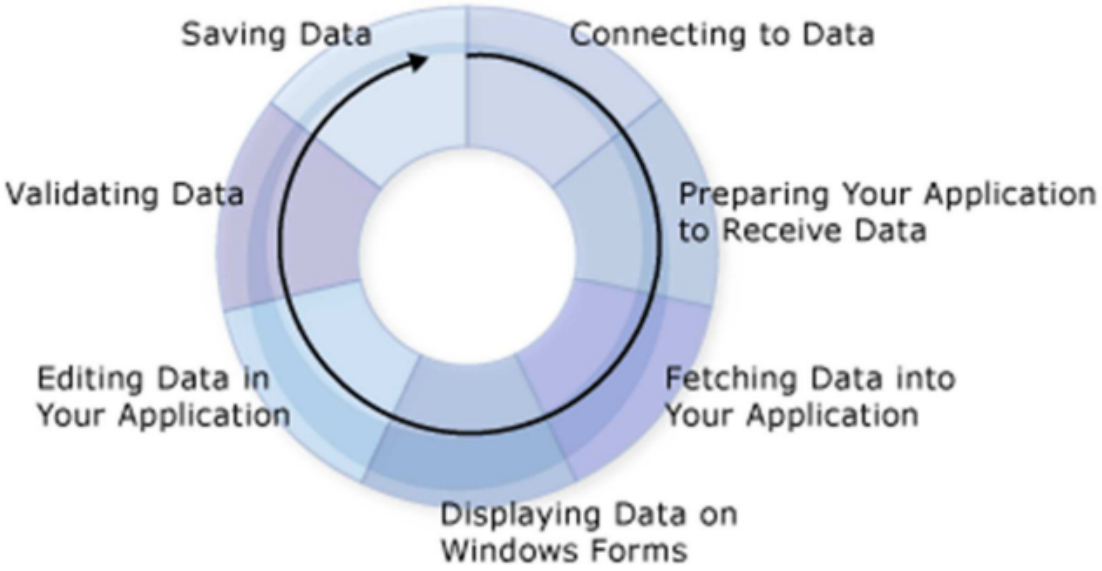
```
select (select title from film where film_id=f.film_id),(select city from city where
city_id=c.city_id),st.store_id,(select first_name||last_name from staff where staff_id=s.staff_id) as Staff,
Count(*) as Rent
from film as f inner join inventory as i on i.film_id=f.film_id
inner join rental as r on r.inventory_id=i.inventory_id
inner join staff as s on s.staff_id=r.staff_id
inner join store as st on st.store_id=s.store_id
inner join address as ad on ad.address_id=st.address_id
```

```
inner join city as c on c.city_id=ad.city_id
group by f.film_id,Grouping SETS(
(st.store_id, s.staff_id, c.city_id)
,(st.store_id, s.staff_id)
,(st.store_id, c.city_id)
,(s.staff_id, c.city_id)
,(st.store_id)
,(s.staff_id)
,(c.city_id)
,()
)
order by Count(*) desc
```

*اگر تنها آیدی ها در خروجی پاسخ بودند نیز صحیح در نظر گرفته خواهد شد .

هدف از سوالات این بخش، آشنایی شما با مباحث APP DEVELOPMENT و مباحث پیشرفته MSSQL است.

7. به سوالات زیر پاسخ دهید
   a. گام های استاندارد استفاده اپلیکیشن ها از درایور های دیتابیس را بنویسید



   b. درایور های ODBC و OLEDB را باهم مقایسه کنید.

ODBC	OLEDB
Originally designed for relational (databases. (since changed	Originally designed for non-relational and relational .databases
On-going support for SQL	SQL support void 2019
Component-based	Procedural-based
More difficult to deploy	Easier to deploy

c. ORM چیست و چه دغدغه هایی را برطرف میکند.چند مثال از ORM های موجود بزنید.

ORM یا Object-Relational Mapping یک لایه مترجم بین زبان برنامه‌نویسی و پایگاه داده رابطه‌ای که این دو را به هم تبدیل می‌کند. و در واقع در زبان به شکل شی گرا به پایگاه داده وصل می شود.

از مزایا آن می توان گفت برنامه نویس دیگر درگیر اسکریپت های sql و نوع دیتابیس و پیچیدگی های دیتابیس نباشد. مثال:

Eloquent ORM - PHP

#Entity Framework - C

Sequelize ORM - JS

8. به سوالات زیر پاسخ دهید

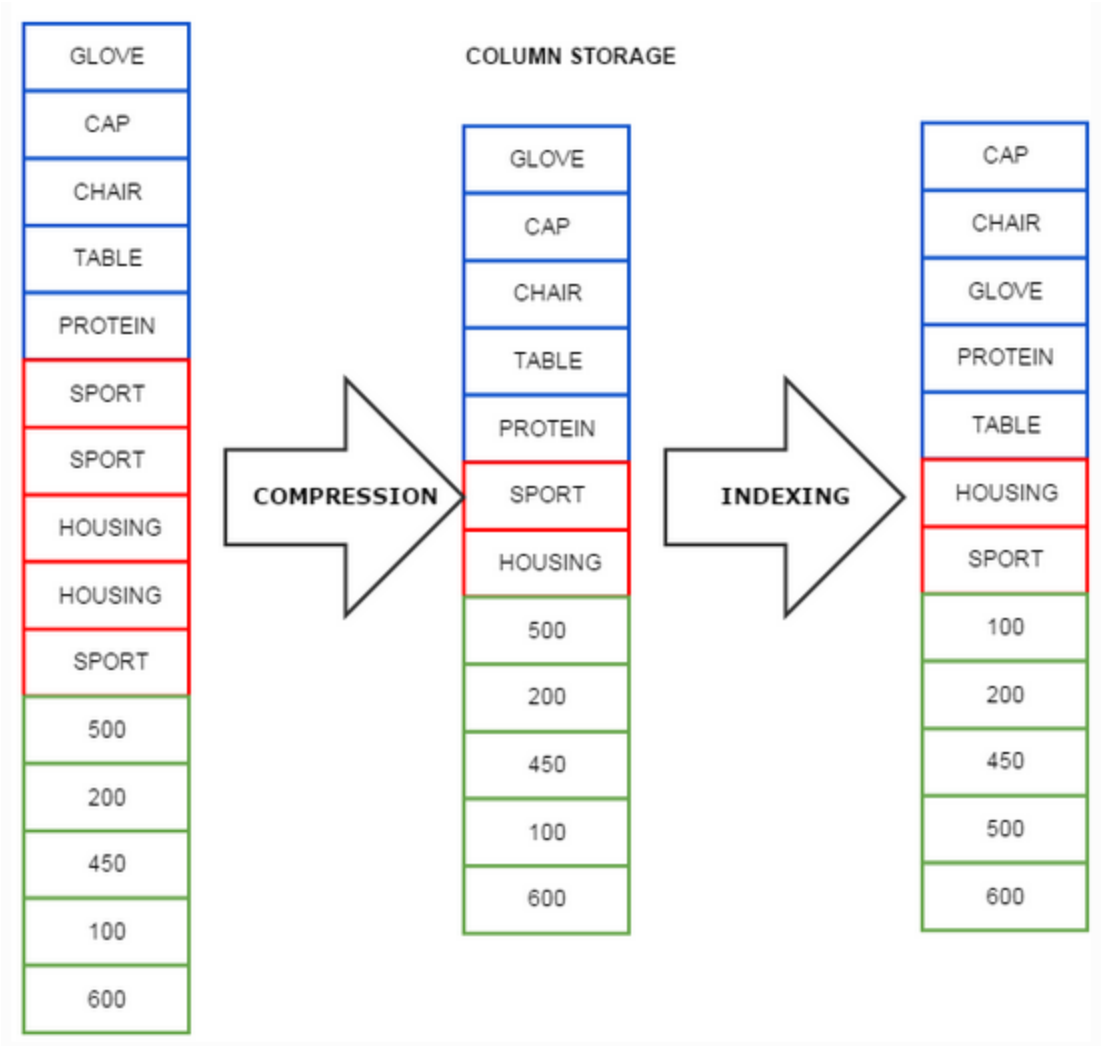
a. نحوه ذخیره سازی رکورد ها به شیوه RowStore و ColumnStore را مختصرا توضیح دهید .

در روش ذخیره سازی RowStore اطلاعات تمام ستون های هر سطر به صورت متوالی در حافظه ذخیره میشوند مانند تصویر زیر :

Row Id	Storage
Row 1	100
	Burger
	10000
	100
	2020-01-01
Row 2	101
	Chicken
	15000
	50
	2020-01-01
Row 3	102
	Burger
	11000
	110
	2020-01-02
Row 4	103
	Salad
	500
	25
	2020-01-02

اما در روش ذخیره سازی ستونی اطلاعات هر ستون به صورت متوالی قرار میگیرند و بعد از فشرده سازی در حافظه قرار میگیرند .در این روش چون اطلاعات هر ستون از یک نوع هستند امکان فشرده سازی داده ها وجود دارد .

Column	Storage
ID	[0]:100, [1]:101, [2]:102, [3]:103
Item_title	[0]:'Burger', [1]:'Chicken', [2]:'Burger', [3]:'Salad'
Price	[0]: 10000, [1]:15000, [2]:11000, [3]:500
Quantity	[0]:100, [1]:50, [2]:110, [3]:25
Date	[0]:' 2020-01-01', [1]:' 2020-01-01', [2]:' 2020-01-02', [3]:' 2020-01-02'



b. دو مورد از محدودیت ها و مشکلاتی که در ذخیره سازی اطلاعات به صورت RowStore ممکن است رخ دهد را شرح دهید.

1-به دلیل پردازش های سنگین و طولانی انجام عملیات تحلیلی بر روی جداول سطری میتواند منجر به تاثیرات نامطلوب بر روی عملیات روزمره شود.

2-مصرف منابع در فرایند بازیابی اطلاعات (اسکن دیسک و مصرف CPU) در عملیات تحلیلی بر روی بانک های اطلاعاتی سطری بسیار بالاست

3-به دلیل نوع طراحی بانک های اطلاعاتی سطری Horizontal Scaling با مشکلات فراوان مواجه است .

c. فرض کنید جهت تحلیل اطلاعات یک سازمان میخواهیم فروش ماهانه 3 سال گذشته را مقایسه کنیم به نظر شما کدام یک از روش های ذخیره سازی اطلاعات در این مورد مناسب تر است ؟ چرا ؟

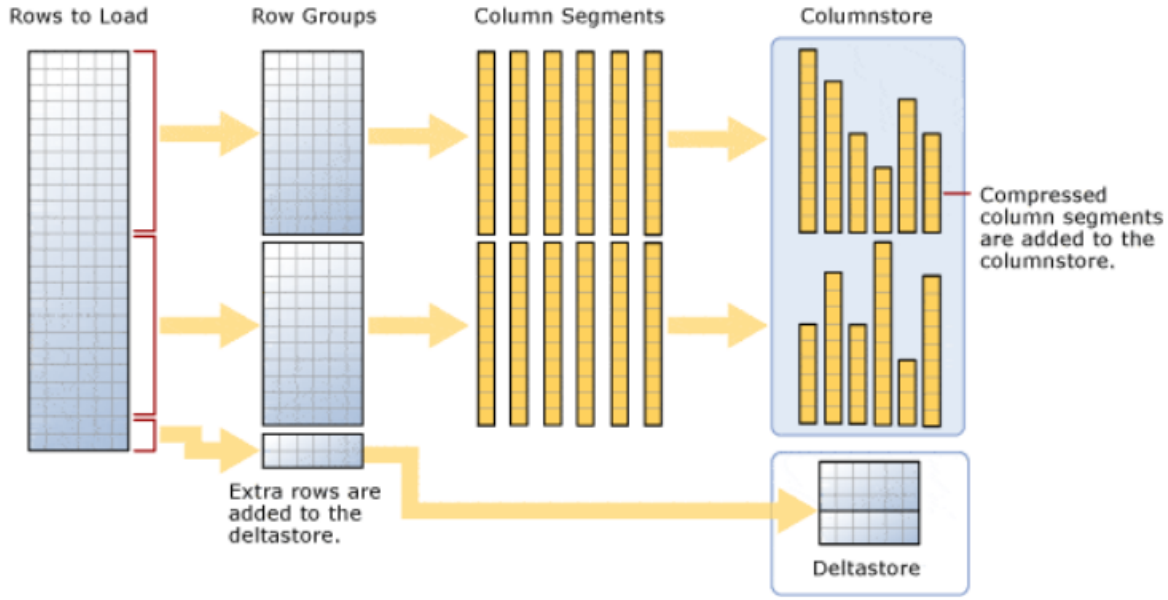
ColumnStore - زیرا در این روش ذخیره سازی مصرف منابع پایین تر است و تعداد دفعاتی که عملیات IO نیاز داریم کمتر است همچنین در تحلیل این اطلاعات تنها به ستون های مقادیر فروش نیاز داریم و نیازی به فراخوانی ستون های دیگر نداریم در نتیجه سرعت اجرای آن سریع تر از RowStore است

d. عملیات های Insert و Update در کدام یک از روش های مذکور بهینه تر عمل میکند ؟ چرا ؟

RowStore-زیرا در این روش کافیست در انتهای رکورد ها در حافظه یک سطر جدید تخصیص داده شود و یا تغییر پیدا کند و کاری با دیگر داده ها نداریم اما در روش ذخیره سازی ColumnStore نیاز داریم تا برای تمام

ستون ها یک بار داده ها را از حالت فشرده خارج کرده دیتا را وارد کرده و یا تغییر دهیم و سپس دوباره آن را فشرده سازی کنیم که از RowStore کند تر عمل میکند.

e. ColumnStore Index روش ترکیبی مایکروسافت است که از Sql Server 2012 به بعد مورد استفاده قرار گرفت . در این نوع ایندکس سعی می شود برخی مزایای هر دو روش را تا حدودی داشته باشیم . درباره آن تحقیق کرده و چگونگی عملکرد آن را مختصرا شرح دهید .



https://www.red-gate.com/simple-talk/databases/sql-server/t-sql-programming-sql-server/what-are-columnstore-indexes/