محمد عبدالهی 9530153

.1

الف)

atomicity : مطمین میشود که تمام عملیات ها داخل یک واحد عملیاتی به صورت atomic و یکپارچه انجام میگیرد و عملیات موفقیت آمیز است.در غیر این صورت transaction متوقف شده و خطا نشان داده میشود و transaction به حالت اولیه بازمیگردد.

Isolation : به واسطه این خاصیت transaction های مختلف مستقل از هم عمل کرده و عملکرد یک transaction از دیگری جداست.

Consistency: در هنگام و پس از اجرای موفقیت امیز یک transaction یکپارچگی دیتابیس حفظ میشود.

Durability: باعث میشود که نتیجه یک transaction موفقیت امیز در هنگامی که دچار یک system failure میشود در سیستم باقی بماند.

ب)

زیرا در این حالت این transaction یک bottleneck برای سیستم محسوب میشود و میتواند باعث کاهش performance شود در حالیکه به علت زیاد بودن تعداد درخواست‌ها در یک سیستم سرعت اجرای transaction ها از اهمیت بالایی برخوردار است پس اجرای همزمان انها باعث افزایش کارایی سیستم میشود.

ج)

یک serial schedule در واقع ترتیب انجام زمانبندی transactionهاست و در ان عملیات های داخل یک transaction همگی با یکدیگر یک گروه شده و به ترتیب اجرا میشوند و یک transaction پس از transaction دیگر اجرا میشود. در اینجا ترتیب انجام transactionها از اهمیت بالایی برخوردار است زیرا در صورت ایجاد تغییر ممکن است به نتیجه دیگری برسیم.

در یک serializable schedule یک serial schedule مانندe و s وجود دارد این دو ترتیب به یک نتیجه می انجامند.

د)

بله زیرا در یک serial چه T1 اول اجرا شود چه T2 باعث میشود که متغیر A یا B مقدارش یک شود و در transaction بعدی شرط اجرا نشود و مقدار یکی از دو متغیر صفر باقی میماند.

.2

الف) در simple view جدول جدیدی نداریم و با هر فراخوانی از view دستور نظیر ان view اجرا میشود اما در یک materialized عملا یک جدول جدید ساخته شده و درخواستهای ما از ان جدول فراخوانی میشود.

ب)هنگامی که دستور ما relation های زیادی دارد اما سریع اپدیت نشود و یا هنگامی که کویری به تعداد زیاد استفاده میشود بهتر است از materialized view استفاده شود. زیرا برخلاف حالت simple در حالتی که کویری زیادی داشته باشیم و جوین های زیادی به کار رفته باشد یک بار دستورات اجرا شده و نتایج در یک جدول ذخیره میشود و دفعات بعد هم از همان جدول ۹۱ ستفاده میشود.

ج) بله

i) اگر شامل جوین بین چندین تیبل باشد هنگام insert کردن در view تنها میتوانید در یک جدول insert کنید و نمیتوان سطری از view را حذف کرد.

ii) اطلاعات view هایی که از union یا مواردی مانند distinct یا group by استفاده میکنند را نمیتوان دستکاری کرد.

iii) ستونهایی را که شامل aggregate ها هستند را نمیتوان ویرایش کرد.

د)

اگر از materialized view استفاده شود به دلیل اینکه به روزرسانی جدول به صورت دوره ای است اگر جدول هایی که از انها در view استفاده شده جدول هایی باشد که اطلاعات انها سریع به روزرسانی میشود ممکن است اطلاعات با اطلاعات دیتابیس سینک نباشد.

در simple view عملا چون با هر بار اجرای اطلاعات به روزرسانی میشود و کویری ها هم با هر بار اجرا دوباره اجرا میشوند از نظر بهینه بودن و بار پردازشی برای کویری هایی که زیاد استفاده میشوند بهینه نیست.

3)

الف) spها در داخل select قابل اجرا نیستند اما functionها هستند.

ب)spها میتوانند هر نوع خروجی داشته باشند یا هیچ خروجی نداشته باشند ولی function ها حتما باید خروجی داشته باشند.

ج) داخل sp میتوان از function استفاده کرد ولی عکس این حالت ممکن نیست.

د) تعداد ورودی مجاز sp بیشتر از function است.

-4

الف)

ب) با توجه به شناور بودن قیمت بلیت و ظرفیت در سیستم و تغییر قیمت در طول زمان، بستگی به خریدی که مسافر انجام میدهد ممکن است قیمت های متفاوتی برای مسافر اعمال شود. یکی از مزیت هایی که ذخیره سازی به این شکل دارد track کردن قیمت پرداختی مسافر بسته به زمانی که بلیت را خرید کرده است می باشد

ز) در یک triggerعملیاتی که قرار است انجام شود به صورت خودکار انجام میشود. در واقع با فعال شدن triggerعملیات انجام شده و کنترل بیشتری بر روی این عملیات داریم ولی در حالتی که با استفاده از transactionها انجام میدهیم خود فردی که عملیات را انجام میدهد باید نظارت بر روی عملیات مورد نظر داشته باشد و در واقع عملیات triggerتوسط DBMS مدیریت می شود ولی در این حالت توسعه دهنده عملیات را مدیریت میکند. مشکلی که پیش می آید هنگامی است که عمل کرد یک triggerشاید موجب

فعال شدن یک triggerدیگر شود که در این حالت برای پیاده سازی با استفاده از transactionدچار پیچیدگی خواهیم شد

5)

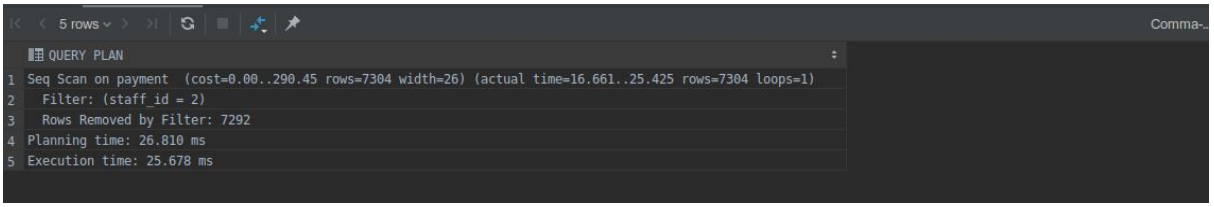
```
create recursive view fact(column1, (AS  column2)
  values(1, 1)
union
SELECT column1+1, column2 * (column1+1) from fact where column2 > 34 );
select column1 from fact ;
```

از int بیشتر شده و دچار overflow میشود.

6-روش اول ایجاد جداول history و انتقال حداقل داده مورد نیاز به آن‌هاست، اشکال این روش این است که در جداول history دیگر نمی‌توان از foreign key های جدول اصلی استفاده کرد ولی می‌توان روی این جداول query های لازم را اجرا کرد.
روش دیگر پارتیشن کردن داده‌های جدول با توجه به زمان آن‌هاست که می‌تواند سرعت load را بالا ببرد ولی مشکلات دیگری می‌تواند ایجاد کند.

-8

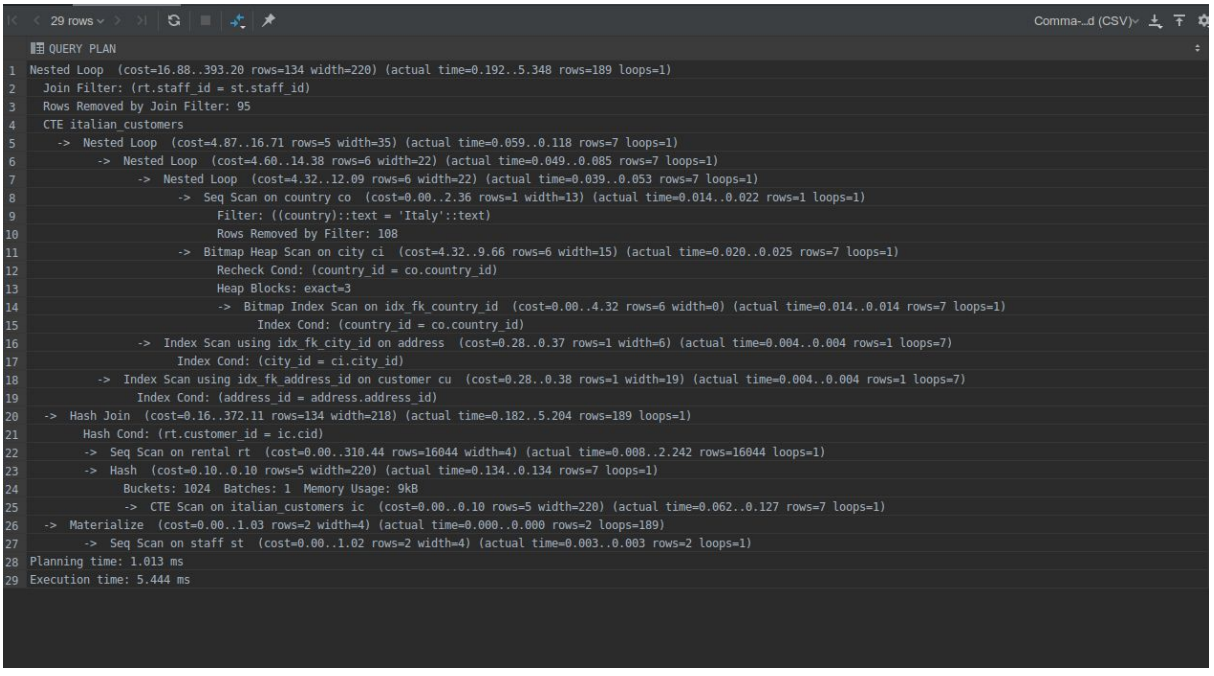
الف)



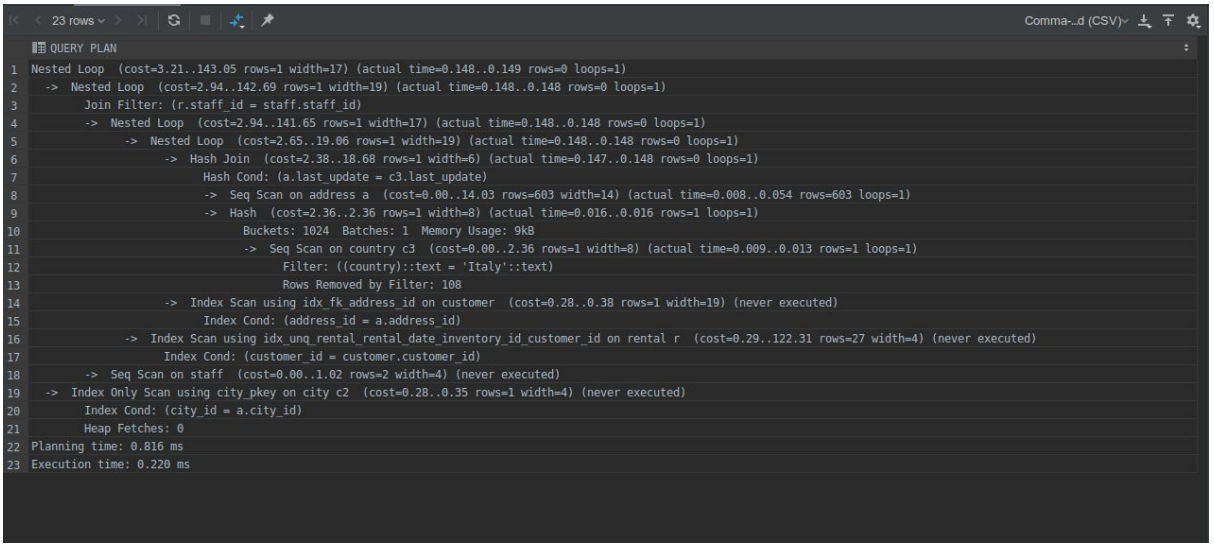
مقدار loops = 1 بیانگر این است که index scan دقیقاً یک بار انجام شده است و نهایتا 7304سطر پیدا شده و باید برگردانده شود. مقدار روبروی Filter بیانگر همان فیلترهایی است که ما در where بر روی سطری که selectکرده ایم اعمال کرده ایم و همچنین بیان شده است که 7292سطر بر اثر این فیلتر از خروجی نهایی حذف شده است. همچنین زمانی که صرف شده است تا planner بهترین راه برای اجرای بهینه ی این دستور را پیدا کند مقدار 52ms است planner time و زمانی که صرف شده تا راه حلی که planner پیدا کرده است، اجرا شود مقدار 20msاست

`ب)

با with



بدون with:



```
QUERY PLAN
Nested Loop  (cost=3.21..143.05 rows=1 width=17) (actual time=0.148..0.149 rows=0 loops=1)
  ->  Nested Loop  (cost=2.94..142.69 rows=1 width=19) (actual time=0.148..0.148 rows=0 loops=1)
        Join Filter: (r.staff_id = staff.staff_id)
        ->  Nested Loop  (cost=2.94..141.65 rows=1 width=17) (actual time=0.148..0.148 rows=0 loops=1)
              ->  Nested Loop  (cost=2.65..19.06 rows=1 width=19) (actual time=0.148..0.148 rows=0 loops=1)
                    ->  Hash Join  (cost=2.38..18.68 rows=1 width=6) (actual time=0.147..0.148 rows=0 loops=1)
                          Hash Cond: (a.last_update = c3.last_update)
                          ->  Seq Scan on address a  (cost=0.00..14.03 rows=603 width=14) (actual time=0.008..0.054 rows=603 loops=1)
                          ->  Hash  (cost=2.36..2.36 rows=1 width=8) (actual time=0.016..0.016 rows=1 loops=1)
                                Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                ->  Seq Scan on country c3  (cost=0.00..2.36 rows=1 width=8) (actual time=0.009..0.013 rows=1 loops=1)
                                      Filter: ((country)::text = 'Italy'::text)
                                      Rows Removed by Filter: 108
                    ->  Index Scan using idx_fk_address_id on customer  (cost=0.28..0.38 rows=1 width=19) (never executed)
                          Index Cond: (address_id = a.address_id)
              ->  Index Scan using idx_unq_rental_rental_date_inventory_id_customer_id on rental r  (cost=0.29..122.31 rows=27 width=4) (never executed)
                    Index Cond: (customer_id = customer.customer_id)
        ->  Seq Scan on staff  (cost=0.00..1.02 rows=2 width=4) (never executed)
  ->  Index Only Scan using city_pkey on city c2  (cost=0.28..0.35 rows=1 width=4) (never executed)
        Index Cond: (city_id = a.city_id)
        Heap Fetches: 0
Planning time: 0.816 ms
Execution time: 0.220 ms
```

در حالت دوم ( )nestedبه دلیل تو در تو بودن ها و nested loopبیشتر، زمان Planning timeو execution timeهر دو بیشتر از حالتی است که با استفاده از withنوشته شده است. نتیجه ای که میتوان گرفت این است که در استفاده از withبه دلیل اینکه تعداد جوین های نهایی ممکن است کمتر شود و indexگذاری های مناسب تری ممکن است اتخاذ شود زمان اجرا میتواند کمتر باشد

ج)

عمل کرد hashبه منظور تبدیل یک رشته ی بزرگ به یک رشته ی کوچکتر توسط dbmsاستفاده میشود. پس از این کار میتوان موارد hashشده را در قالب جدول هایی و مجموعه هایی راحت تر دسته بندی و مرتب کرد و بنابراین میتوان آن ها را ً indexگذاری کرد و نهایتا برای retrieveکردن داده ها با استفاده از مقادیر indexها سریع تر داده ها را پیدا و استفاده کرد.

همچنین همانطور که از Query Planها پیداست عملیات HASHقبل از عملیات indexها و قبل از عملیات های نظیر جست و جو و... استفاده شده است