

«به نام خدا»

تکلیف دوم DB1 - مرضیه علیدادی - 9631983

(1)

الف) با استفاده از آن، schema ی هر جدول را می توان مشخص کرد. نوع و domain مربوط به هر فیلد را می توان تعیین کرد. Integrity constraints تعیین می شود که موجب حفظ مقادیر مجاز و ارتباطات مجاز بین جداول می شود. با استفاده از آن، می توانیم index تعریف کنیم. تعریف authorization و security را انجام می دهد. و حتی می توان physical storage structure را هم با استفاده از دستورات آن مشخص کرد.

ب) مجموعه فایل های کمکی هستند، که به کارایی DBMS کمک می کنند و باعث می شوند تا به data دسترسی سریع تری داشته باشیم. ما می توانیم این فایل ها را تعریف کنیم و از آنها استفاده کنیم. البته اگر این ها وجود نداشته باشند هم، DB کارش را انجام می دهد، منتها این ها کمک می کنند تا بعضی کارها مثل query ها و search های بزرگ، سریعتر و با کارایی بهتر انجام شوند.

(2) از مزایای این type این است که به صورت پویا در هنگام قرار یک مقدار برای یک فیلد از این type، طول آن فیلد متناسب با آن مقداری که در آن فیلد ریخته می شود، تعریف می شود. در نتیجه هم برای مقادیر کوچک صرفه جویی می شود و هم برای مقادیر بزرگ ریسک کم آمدن فضا وجود ندارد. اما این نوع متغیر درست است که موجب صرفه جویی می شود، ولی مشکلاتی را در کارایی و سختی هایی را در برخی عملیات موجب می شود. برای مثال وقتی مقداری را در یک فیلد با این type قرار می دهیم، باید طول آن را هم به صورت جداگانه ذخیره کنیم، که خود این هم یک فضای اضافه تر نیاز دارد. و مثلاً برای کارهای دیگر از قبیل search کردن اگر طول داده ها fix باشد با یک سری عملیات ساده می توانیم به عملیات مورد نظر خودمان برسیم، اما در حالتی که طول داده ها برابر نباشد این عملیات به این سادگی نخواهد بود و کندتر می شود.

3) در database اطلاعات با یکدیگر در ارتباط هستند، منتها ما برای کارایی بهتر و کاهش redundancy و مشکلاتی که قبلاً گفته شد، این اطلاعات مرتبط را در جداول جداگانه‌ای قرار می‌دهیم. ولی اطلاعات ذخیره شده در این جداول با یکدیگر در ارتباط هستند و به هم وابسته اند؛ لذا برای جلوگیری از دوگانگی و ناهمخوانی داده ها باید فیلدهای مشترکی بین آنها وجود داشته باشد تا آنها را به یکدیگر متصل کند. این فیلد های مشترک که همان foreign key هستند، به صورت حساب شده ای به اشتراک گذاشته می شوند و اهمیت زیادی در integrity پایگاه داده دارند.

4) زمانی که بین value هایی که می خواهیم روی آن ها عملیات avg را انجام بدهیم، مقدار null وجود داشته باشد، مقدار حاصل از این دو عملیات می تواند متفاوت باشد. چون avg، null را در محاسبه ی میانگین در نظر نمی گیرد و میانگین باقی مقادیر را برمی گرداند. در عملیات دیگر؛ sum هم در واقع مقادیر غیر null را با هم جمع می کند و بر می گردند. اما در count بستگی به این دارد که ما همان فیلد را بشماریم یا رکورد ها را بشماریم. اگر count را برای شمردن فیلدی که در آن null ظاهر شده است در نظر بگیریم و sum را بر آن تقسیم کنیم، حاصل دقیقاً مانند avg می شود؛ اما اگر در count تعداد کل رکورد ها را با هم در نظر بگیریم، در این صورت ممکن است آن رکوردی که مقدار فیلد مورد نظر در آن null است را هم بشمارد و در این صورت حاصل تقسیم sum به این عدد، مقداری متفاوت از نتیجه ی avg خواهد بود.

مورد دیگری که حاصل این دو عملیات ممکن است متفاوت باشد، زمانی است که مخرج عبارتی که قرار است این عملیات روی آن انجام بشود، صفر می شود. به طور کلی avg صفر در مخرج (تقسیم بر صفر) را در نظر نمی گیرد.

5) query ای است که یک table را با خودش join می کند.(self join)

برای مثال، ممکن است بخواهیم با استفاده از جدول emp_super که به شکل زیر است، نشان دهیم که مدیر هر شخص چه کسی است. در این صورت با یک select ساده این عمل قابل انجام است.

person	supervisor
Bob	Alice
Mary	Susan
Alice	David
David	Mary

اما اگر مثلاً بخواهیم مشخص کنیم که مدیرِ مدیرِ هر شخص چه کسی است، آنگاه باید در دو مرحله از این جدول استفاده کنیم. که در این صورت دو بار آن را rename می کنیم.

```
select S.person, S.supervisor, T.supervisor
from emp_super as S, emp_super as T
where S.supervisor = T. person
```

خروجی حاصل از این query به صورت زیر است:

S.person	S.supervisor	T.supervisor
Bob	Alice	David
Mary	Susan	Null
Alice	David	Mary
David	Mary	Susan

6 در query اول، با توجه به اینکه کلمه ی کلیدی **distinct** روی تابع تجمیعی **count** اعمال شده است و این تابع تنها یک عدد باز میگرداند، **distinct** هیچ اثری روی نتیجه ی query ندارد. و در نهایت جمع تعداد **v_code** ها به عنوان خروجی بر میگردد؛ یعنی نتیجه، تعداد **row** ها در جدول **Product** خواهد بود. ولی در query دوم، چون عملیات **distinct** روی **v_code** اعمال شده است، این قسمت از query یعنی **distinct v_code** ، **v_code** ها را به صورت یکتا برمی گرداند. در نهایت، **count** تعداد این مقادیر یکتا را محاسبه می کند. نتیجه ی این query هم یک عدد است که تعداد **row** های جدول با شرط یکتا بودن **v_code** های آنها را نشان می دهد. لذا عددی که query دوم حاصل می شود، کوچکتر یا مساوی عدد حاصل از query اول است.

7 زمانی که از دو **select** تودرتو استفاده می کنیم، اگر در **outer query** که همان **select** بیرونی است، یک متغیر جدید مثلاً یک جدول جدید تعریف کنیم، ما یک **correlation** بین این دو **select** برقرار کرده ایم. در این حالت به آن متغیری که در **outer query** ایجاد شده است، **correlation name** می گوییم و به **subquery** داخلی که از آن **correlation name** استفاده می کند، **correlated subquery** می گوییم.

(8

```
delete from Customer
where (ID, LastUpdate) in ((select Customer.ID, LastUpdate
                           from Customer)
except
(select ID, max(LastUpdate)
 from Customer
 group by ID))
```

.a (9

```
1 select dept_name
2 from department
3 where budget > (select budget
4                 from department
5                 where dept_name = 'Psychology')
6 order by dept_name asc
```




Notifications Messages Data Output Explain Scratch Pad

	dept_name [PK] character varying (20)	
1	Finance	
2	Physics	

.b (9

```
1 select takes.id, takes.course_id
2 from takes
3 where 3 <= (select count(T.sec_id)
4             from takes as T
5             where (takes.id, takes.course_id) = (T.id, T.course_id))
```

Notifications Messages Data Output Explain Scratch Pad

	 id character varying (5)	 course_id character varying (8) 
1	44881	362
2	39978	362
3	69581	362
4	39925	362
5	16969	362
6	39978	362
7	9993	362
8	27236	362
9	69581	362
10	49611	362
11	5414	362
12	49611	362
13	16480	362
14	5414	362
15	16969	362
16	39925	362
17	9993	362
18	49611	362

.c (9)

```
1 select I.id, I.name
2 from instructor as I
3 where not exists ((select C.course_id
4                    from course as C
5                    where C.dept_name = I.dept_name)
6 except
7 (select T.course_id
8  from teaches as T
9  where T.id = I.id))
```

Notifications Messages Data Output Explain Scratch Pad

	id [PK] character varying (5)		name character varying (20)	

.d (9)

```
1 select name
2 from student
3 where name like '___' and dept_name = 'History'
```

Notifications Messages Data Output Explain Scratch Pad

	name character varying (20)	
1	Yap	
2	Sud	
3	Maw	
4	Usi	
5	Ssu	

. a (10

```

1 select first_name, last_name, city
2 from customer C, address A, city CT, country CN
3 where country = 'Iran' and first_name like '_____'
4      and (C.address_id, A.city_id, CT.country_id) = (A.address_id, CT.city_id, CN.country_id)

```

Notifications

Messages

Data Output

Explain

Scratch Pad

	<div>first_name</div> <div>character varying (45)</div>	<div>last_name</div> <div>character varying (45)</div>	<div>city</div> <div>character varying (50)</div>
1	Harry	Arce	Najafabad
2	Tommy	Collazo	Qomsheh
3	Oscar	Aquino	Sirjan

. b (10

```
1 select title
2 from film, inventory, rental
3 where length < 100 and rental_rate < 2 and store_id = 2 and (return_date - rental_date) < '1 day'
4      and (film.film_id, inventory.inventory_id) = (inventory.film_id, rental.inventory_id)
```

Notifications

Messages

Data Output

Explain

Scratch Pad

▲

title

character varying (255)

🔒

1	Alone Trip
2	Anaconda Confessions
3	Arabia Dogma
4	Arabia Dogma
5	Arabia Dogma
6	Arabia Dogma
7	Armageddon Lost
8	Bound Cheaper
9	Bride Intrigue
10	Butterfly Chocolat
11	Caddyshack Jedi
12	Caddyshack Jedi
13	Canyon Stock
14	Cheaper Clyde
15	Club Graffiti
16	Coast Rainbow
17	Encounters Curtain
18	Encounters Curtain

.c (10

روش اول:

```
1  ----(1)
2  (select first_name, last_name
3    from actor A, film_actor FA, film F
4    where rental_rate > 4
5          and (FA.actor_id, FA.film_id) = (A.actor_id, F.film_id))
6  except
7  (select first_name, last_name
8    from actor A, film_actor FA, film F
9    where length > 180
10         and (FA.actor_id, FA.film_id) = (A.actor_id, F.film_id))
11  order by last_name, first_name
```

Notifications Messages Data Output Explain Scratch Pad

	first_name character varying (45)	last_name character varying (45)	
1	Debbie	Akroyd	
2	Kim	Allen	
3	Harrison	Bale	
4	Michael	Bening	
5	Scarlett	Bening	
6	Vivien	Bergen	
7	Christopher	Berry	
8	Henry	Berry	
9	Karl	Berry	
10	Kevin	Bloom	
11	Goldie	Brody	
12	Laura	Brody	
13	Laurence	Bullock	

10 c.

روش دوم:

```
1  ----(2)
2  select distinct first_name, last_name
3  from actor A, film_actor FA, film F
4  where rental_rate > 4
5         and (FA.actor_id, FA.film_id) = (A.actor_id, F.film_id)
6         and not exists (select actor_id
7                           from film_actor, film
8                           where length > 180
9                             and (film_actor.actor_id, film_actor.film_id) = (A.actor_id, film.film_id))
10 order by last_name, first_name
```

	first_name character varying (45)	last_name character varying (45)
1	Debbie	Akroyd
2	Kim	Allen
3	Harrison	Bale
4	Michael	Bening
5	Scarlett	Bening
6	Vivien	Bergen
7	Christopher	Berry
8	Henry	Berry
9	Karl	Berry
10	Kevin	Bloom
11	Goldie	Brody
12	Laura	Brody
13	Laurence	Bullock

```

1 (select distinct first_name, last_name
2   from actor A, film_actor FA, film F
3   where rental_rate > 4
4         and (FA.actor_id, FA.film_id) = (A.actor_id, F.film_id))
5 union
6 (select distinct first_name, last_name
7   from customer C, film F, inventory I, rental R
8   where rental_rate < 1 and (return_date - rental_date) < '1 day'
9         and (F.film_id, I.inventory_id, R.customer_id) = (I.film_id, R.inventory_id, C.customer_id))

```

Notifications Messages Explain Scratch Pad Data Output

	first_name character varying (45)	last_name character varying (45)
1	Rip	Crawford
2	Sue	Peters
3	Hilda	Hopkins
4	Caroline	Bowman
5	Jeremy	Hurtado
6	Kurt	Emmons
7	Vivian	Ruiz
8	Richard	Penn
9	Bill	Gavin
10	Nick	Wahlberg
11	Sidney	Burleson
12	Dana	Hart
13	Tom	Mckellen
14	Kenneth	Torn
15	Tim	Hackman

```

1 select distinct first_name, last_name
2 from actor A, film_actor FA, film F
3 where (FA.actor_id, FA.film_id) = (A.actor_id, F.film_id)
4       and rental_rate < all(select rental_rate
5                             from film
6                             where length > 184)

```





Notifications Messages Data Output Explain Scratch Pad

	first_name character varying (45)	last_name character varying (45)	
1	Adam	Grant	
2	Adam	Hopper	
3	Al	Garland	
4	Alan	Dreyfuss	
5	Albert	Johansson	
6	Albert	Nolte	
7	Alec	Wayne	
8	Angela	Hudson	
9	Angela	Witherspoon	
10	Angelina	Astaire	
11	Anne	Cronyn	
12	Audrey	Bailey	
13	Audrey	Olivier	
14	Bela	Walken	
15	Ben	Harris	
16	Ben	Willis	
17	Bette	Nicholson	

.f (10

```
1 with T(id, tot_amount, order_num) as
2 (select customer.customer_id, sum(amount), count(payment_id)
3  from customer, payment
4  where payment.customer_id = customer.customer_id
5  group by customer.customer_id)
6 select id, tot_amount, order_num
7 from T
8 where order_num < 15
```

Notifications Messages Data Output Explain Scratch Pad

	 id integer	 tot_amount numeric	 order_num bigint	
1	272	65.87	13	
2	315	67.86	14	
3	110	49.88	12	
4	281	32.90	10	
5	464	67.86	14	
6	318	27.93	7	
7	136	59.86	14	
8	48	67.86	14	
9	61	57.87	13	
10	124	57.86	14	
11	310	68.87	13	
12	248	37.87	13	

```

1 with tot_order(first_name, last_name, order_num) as
2 (select customer.first_name, customer.last_name, count(payment_id)
3  from customer, payment
4  where payment.customer_id = customer.customer_id
5  group by customer.first_name, customer.last_name),
6      avg_tot_order(avg_order_num) as
7 (select avg(order_num)
8  from tot_order)
9 select first_name, last_name
10 from tot_order, avg_tot_order
11 where order_num > avg_order_num

```

Notifications Messages Data Output Explain Scratch Pad

	first_name character varying (45)	last_name character varying (45)	
1	Sue	Peters	
2	Hilda	Hopkins	
3	Jeremy	Hurtado	
4	Maria	Miller	
5	Mitchell	Westmoreland	
6	Salvador	Teel	
7	Elsie	Kelley	
8	Bill	Gavin	
9	Miguel	Betancourt	
10	Gerald	Fultz	
11	Esther	Crawford	
12	Dana	Hart	
13	Lena	Jensen	

10) h.

برای هر ژانر، فیلمی که هم بیشترین طول و هم بیشترین امتیاز را دارد، برمی گرداند. دلیل اینکه بعضی ژانرها در خروجی نیامده اند، این است که، در آن ها فیلمی که هر دو این ویژگی ها را داشته باشد، وجود ندارد.

```
1 with max_of_genre(name, max_rate, max_len) as
2 (select name, max(rental_rate), max(length)
3  from film F, category C, film_category FC
4  where (F.film_id, FC.category_id) = (FC.film_id, C.category_id)
5  group by name)
6 select distinct C.name, title
7 from film F, category C, film_category FC, max_of_genre
8 where (F.film_id, FC.category_id) = (FC.film_id, C.category_id)
9       and C.name = max_of_genre.name
10      and (F.rental_rate, F.length) = (max_rate, max_len)
```

Notifications Messages Data Output Explain Scratch Pad

	name character varying (25)	title character varying (255)
1	Action	Darn Forrester
2	Comedy	Control Anthem
3	Documentary	Wife Turn
4	Family	King Evolution
5	Games	Chicago North
6	Music	Home Pity
7	New	Frontier Cabin
8	Sci-Fi	Soldiers Evolution

```

1  select name, count(rental_id)
2  from category C, film_category FC, film F, inventory I, rental R
3  where (C.category_id, FC.film_id) = (FC.category_id, F.film_id)
4         and (F.film_id, I.inventory_id) = (I.film_id, R.inventory_id)
5  group by name

```

Notifications Messages Data Output Explain Scratch Pad

	name character varying (25)	count bigint
1	Family	1096
2	Games	969
3	Animation	1166
4	Classics	939
5	Documentary	1050
6	New	940
7	Sports	1179
8	Children	945
9	Music	830
10	Travel	837
11	Foreign	1033
12	Drama	1060
13	Horror	846
14	Action	1112
15	Sci-Fi	1101
16	Comedy	941

```

1 with count_film(rating, name, count_f) as
2 (select rating, name, count(F.film_id)
3  from film F, category C, film_category FC
4  where (F.film_id, FC.category_id) = (FC.film_id, C.category_id)
5  group by rating, name),
6   max_film(rating, max_f) as
7 (select rating, max(count_f)
8  from count_film
9  group by rating)
10 select count_film.rating as rating, count_film.name as favorite_genre
11 from count_film, max_film
12 where count_film.rating = max_film.rating
13       and count_film.count_f = max_film.max_f
14 order by count_film.rating

```

Notifications Messages Explain Scratch Pad Data Output

	rating mpaa_rating	favorite_genre character varying (25)
1	G	Action
2	PG	Family
3	PG-13	Drama
4	R	Sci-Fi
5	NC-17	Music


```

1 with duration(title, rental_duration) as
2 (select title, concat(rental_duration, ' ', 'day')
3  from film),
4  later(name, count_late) as
5  (select name, count(R.rental_id)
6   from category C, film_category FC, film F, inventory I, rental R, duration
7   where (C.category_id, FC.film_id) = (FC.category_id, F.film_id)
8         and (F.film_id, I.inventory_id) = (I.film_id, R.inventory_id)
9         and (return_date - rental_date) > cast(duration.rental_duration as interval)
10  group by name),
11  sooner(name, count_soon) as
12  (select name, count(R.rental_id)
13   from category C, film_category FC, film F, inventory I, rental R, duration
14   where (C.category_id, FC.film_id) = (FC.category_id, F.film_id)
15         and (F.film_id, I.inventory_id) = (I.film_id, R.inventory_id)
16         and (return_date - rental_date) < cast(duration.rental_duration as interval)
17  group by name),
18  on_time(name, count_on_time) as
19  (select name, count(R.rental_id)
20   from category C, film_category FC, film F, inventory I, rental R, duration
21   where (C.category_id, FC.film_id) = (FC.category_id, F.film_id)
22         and (F.film_id, I.inventory_id) = (I.film_id, R.inventory_id)
23         and (return_date - rental_date) = cast(duration.rental_duration as interval)
24  group by name)
25 select category.name, count_late, count_soon, count_on_time
26 from later, sooner, on_time, category
27 where category.name = later.name and later.name = sooner.name and sooner.name = on_time.name

```

	name character	count_late bigint	count_soon bigint	count_on_time bigint
1	Action	554486	540323	191
2	Comedy	494542	437267	191
3	Drama	514743	538054	203
4	Foreign	512940	508487	573
5	Games	503861	450948	191
6	Music	423422	395387	191
7	Sci-Fi	563224	529370	406
8	Sports	614216	549593	191
9	Travel	397479	429127	394

شکل واضح تر از جدول:

	name character varying (25)	count_late bigint	count_soon bigint	count_on_time bigint
1	Action	554486	540323	191
2	Comedy	494542	437267	191
3	Drama	514743	538054	203
4	Foreign	512940	508487	573
5	Games	503861	450948	191
6	Music	423422	395387	191
7	Sci-Fi	563224	529370	406
8	Sports	614216	549593	191
9	Travel	397479	429127	394