زهرا نخشندی ۹۸۵۲۲۱۵۷ «بنام خدا»

۱- ۹) در واقع برعکس normalization است. در normalization جداول را جدا می‌کنیم و جدول‌های بیشتری تولید می‌شود. اما در de normalization جدول‌های جدا را با هم join زده و مرتب می‌کند. و در واقع جدول‌ها در فرم نرمال پایین‌تری می‌سازد.

فرم نرمال بالاتر

و کاربرد آن به دلیل این است که تعداد جدول بالا، موقع استفاده، تعداد join های بیشتری میطلبد که هزینه‌ی زمانی دارد. پس در واقع de normalization انجام می‌دهیم تا از join جلوگیری کنیم.

b) Normalization یک فرایند تجزیه و تحلیل schema ی رابطه‌ای داده‌شده است که براساس primary key ها و FD های آن انجام می‌شود، تا بتوانیم به ویژگی‌های مطلوبمان برسیم. اگر طراحی نرمال نباشد ممکن است anomaly رخ دهد.

در واقع در کیفیت یک رابطه موارد زیر مهم است:

۱- درک معانی و ارتباط صفات یک رابطه و مفهوم بودن آن و مرتبط بودن همه‌ی آن‌ها به هم

۲- کاهش redundency و تکرار که باعث گرفتن حافظه اضافی می‌شود. در واقع anomaly های که در نتیجه‌ی آن ممکن است رخ دهد مربوط به زمان‌های deletion ، insertion و modifications (update) است. که با نرمال‌سازی این ناهنجاری‌ها حذف میشود.

۳- کاهش Null value tuples گرفتن حافظه‌ی اضافه و ایجاد اشکال در aggregate ها و ...).

۴- عدم ایجاد تاپل‌هایی که در دنیای واقعی معنی ندارند و نادرستند

با استفاده از normalization ما به این اهداف می‌رسیم و redundency را کاهش می‌دهیم

مزیت‌ها: از بین بردن تکرار و null values و بهبود مواردی که اشاره کردم در بالا و بهبود زمان.

معایب: در normalization، چون جداول را جدا کرده و جدول‌هایمان بیشتر شده‌اند، موقع استفاده مجبوریم جدول‌ها را با هم join بزنیم که هزینه دارد.

c) در واقع redundency به دلیل برنامه‌ریزی ناکافی و رعایت نکردن قوانین در طراحی دیتابیس بوجود می‌آید و به معنی تکرار اطلاعات در جداول است. اطلاعات تکراری و اضافی که حجم دیتابیس را افزایش میدهد و موجب ایجاد anomaly در موارد مختلف میشود.

وزمانی که normalization را به طور درست و کامل انجام دهیم redundancy از تمام جداول حذف می کند و بسیار به کاهش آن کمک می کند و این به شرط است که منظورمان از redundancy ایجاد نشدن spurious tuple باشد. که نرمال سازی آن را از بین می برد.
به دلیل اینکه در BCNF تمام FD ها وابستگی به ابرکلید را مشخص می کنند و FD دیگری وجود ندارد که این باعث می شود داده تکراری نداشته باشیم. (گاهی تکرار ناخواسته ایجاد شده مثلا نام سازی از افراد که
که آن مربوط به وابستگی ها نیست بلکه به دلیل اینکه آن داده خاص اینطور بوده است، و این مدل تکرار با نرمال سازی از بین نمی رود.

در واقع به صفر رساندن redundancy ممکن نیست و گاهی اوقات آن مقدار کم از خطرا قابل قبول است. چون گاهی ممکن است یک داده خاص بین همه tuple ها تکرار شود. اما مهم این است که spurious tuple نداشته باشیم که این ها با نرمال سازی از بین می روند.)

۲- a) درست. وقتی کلیدهای R ساده باشند یعنی فقط یک فیلد دارند. وقتی دانیم اگر R بخواهد در فرم BCNF باشد باید برای هر FD، $X \to Y$ ، X یک super key باشد، وقتی R در فرم 3NF است یعنی یا (۱) X است super key، یا (۲) Y عضو candidate key است. اگر شرط اول برقرار باشد که BCNF است. اگر شرط دوم برقرار باشد آنگاه چون کلیدها یک فیلد دارند و Y خودش کلید است تمام صفات غیرکلید به Y بستگی دارند و Y تمام آن‌ها را determine می‌کند، حالا می‌دانیم که $Y \to X$ و تمام صفات $Y \to$ پس با تعدی، تمام صفات $\to X$ که در تناقض با فرض 3NF است. پس در واقع در این حالت همه FD ها جزو شرط اول هستند یعنی X حتما super key است و در فرم BCNF است. ✓

b) درست. چون R در فرم 3NF است پس نمی‌توانیم در FD ها تعدی داشته باشیم در واقع نمی‌شود که X ، کلید باشد و $X \to Y$ و $Y \to Z$. همچنین می‌دانیم چون یک کلید داریم تمام صفت های دیگر تنها به این یک کلید وابسته هستند. پس در تمام FD ها این کلید در سمت چپ قرار دارد که نشان می‌دهد R در BCNF نیز قرار دارد چون سمت چپ تمام FD تنها همین کلید که super key است قرار دارد. ✓

c) غلط است زیرا هر رابطه R که در فرم BCNF باشد در فرم 3NF نیز هست. زیرا BCNF یعنی طرف چپ هر FD حتما یک super key است. که یکی از شرط‌های لازم که باعث می‌شود R در فرم 3NF حتما باشد پس هر R در که فرم BCNF باشد یعنی در فرم 3NF نیز هست. ✗

۳- R دارای صفات ABCDE است.

FD ها ظاهر نشده باشند super key ها آن هایی هستند که در سمت راست

اما همهٔ صفات به آن وابسته باشند. $\{AB \to C, CD \to E\}$

پس C و E نمی توانند superkey باشند.

$$\left.\begin{matrix} AB \to C \\ CD \to E \end{matrix}\right\} \Longrightarrow \boxed{ABD \to E}$$

$$AB \to C \Longrightarrow ABD \to CD \Longrightarrow \boxed{ABD \to C} \Longrightarrow \boxed{ABDE \to C}$$

پس ABD می تواند super key باشد. و ABDE هم همینطور (البته مینیمم نیست)

$$CD \to E \Longrightarrow ABCD \to ABE \Longrightarrow \boxed{ABCD \to E}$$

پس ABCD نیز می تواند superkey باشد.

خود ABCDE نیز می تواند super key باشد (بدیهیست)

ویژگی دیگر استفاده شده در جواب: $\text{If } X \supseteq Y \text{ then } X \to Y$

پس ۴ super key دارد.

۴- ① ابتدا باید سمت راست همهٔ FD ها تک صفت باشد.

$$G=\{ABH \to C, ABH \to K, A \to D, C \to E, BGH \to L, L \to A, L \to D, E \to L, BH \to E\}$$

② به ازای هر FD بررسی می کنیم می توانند سمت چپش با زیرمجموعه ای از آن جایگزین شود یا نه.

$ABH \to C$ حذف می شود و $BH \to C$ جایگزین می شود.

$ABH \to K$ حذف می شود و $BH \to K$ جایگزین می شود.

$BGH \to L$ حذف می شود

و یا $BH \to L$ جایگزین میشود.

(در واقع اینجا همه صفات به BH وابسته اند و BH میتواند super key باشد.)

$G = \{BH \to C, BH \to K, A \to D, C \to E, BH \to L, L \to A, L \to D, E \to L, BH \to E\}$

③ به ازای هر FD اگر $F-\{FD\}$ هم ارز با F بود آن را از F حذف می کنیم.

$BH \to E, E \to L \implies BH \to L$ ⟹ $BH \to L$ حذف می شود.

$BH \to C, C \to E \implies BH \to E$ ⟹ $BH \to E$ حذف می شود.

$L \to A, A \to D \implies L \to D$ ⟹ $D \to L$ حذف می شود.

$\implies G = \{BH \to C, BH \to K, A \to D, C \to E, L \to A, E \to L\}$

↓

minimal cover

---

④ - ابتدا FD ها را مشخص می کنیم. در اینجا (SID)(CID) primary key است.

$SID \to S\text{-name}$

$(SID)(CID) \to Grade$

$CID \to Factulty$, $CID \to C\text{-name}$

$Faculty \to F\text{-Phone}$

این رابطه در فرم 1NF است.

آن را به 2NF می بریم.

از آن جایی که $(SID)(CID)$ کلید است پس $SID \to S\text{-name}$ و $CID \to C\text{-name}$، $CID \to Faculty$ شرط 2NF را نقض می کند (partial dependency). پس R را به $(CID)(C\text{-name})(Faculty)(F\_Phone)$, $(SID)(S\text{-name})$, $(SID)(CID)(Grade)$ می شکنیم.

SID	CID	Grade
۱	۱۵۳۱۸	A
۱	۱۵۳۰۱	B
۲	۱۵۳۱۸	A
۳	۱۵۳۱۸	B
۴	۱۵۳۰۱	A
۴	۱۵۳۱۸	B

CID	C-name	Faculty	F-Phone
۱۵۳۱۸	Database	Howser	۶۰۱۹۴
۱۵۳۰۱	Program	Langley	F۵۸۹۹

Pk = CID

Pk = (SID)(CID) ، Fk = SID, CID

داده های تکراری درس حذف شد.

SID	S-name
۱	Adams
۲	jones
۳	Smith
۴	Baker

→ Primary key = SID

حالا 3NF را بررسی میکنیم.

Faculty → F-Phone شرط 3NF را نقض می کند.

زیرا CID → Faculty ، Faculty → F-Phone و تعدی داریم.

پس باید این جدول را به (Faculty)(F-Phone) و (CID)(C-name)(Faculty) بشکنیم.

CID	C-name	Faculty
۱۵۳۱۸	Database	Howser
۱۵۳۰۱	Program	Langley

Faculty	F-Phone
Howser	۶۰۱۹۴
Langley	F۵۸۹۹

در آخر جداولی که دورشان خط کشیدیم می ماند.

می بینیم که سمت چپ همه FD ها یک کلید قرار دارد.

پس در فرم BCNF نیز هست.

Pk = CID

Fk = Faculty

Pk = Faculty

(CID)(C-name)(Faculty) ، (SID)(S-name) ، (SID)(CID)(grade)

(Faculty)(F-Phone)

* در 591 چک کردم *

۶- الف غلط است. باید قبل از c در Count کلمه Distinct گذاشته شود. در این صورت میتوان گفت ابتدا بر اساس b، Group by کرده و سپس چک می کند در هر گروه با b یکسان c های متفاوت نباشد، یعنی هرگاه b یکسان بود، c ها هم باید یکسان باشند و به عبارتی $b \to c$ (count(Distinct c))

اما با نداشتن کیورد Distinct باعث می شود Count(c) تعداد تمام c های آن گروه (در واقع تعداد تمام tuple های این ستون b با آن b خاص دارند را برمی گرداند که اگر تعداد آن b خاص از یک بیشتر باشد یعنی مثلا دو سطر با b برابر داشته باشیم حتی اگر این دو سطر دارای c برابر باشند Count(c) را ۲ برمی گرداند که نباید اینطور باشد و باید تعداد c های متفاوت را برگرداند

۷- با این دستور، جدول customer ساخته می شود.

Name | Industry | Project1-id | Project1-feedback | project2-id | project2-feedback | contact-person-id | contact-Person-and-role | Phone-number | address | zip | city

* (تمام query های این سوال برای ساخت و تغییر جداول در فایلی در کنار Pdf پیوست شده)*

این رابطه در فرم 1NF نیست زیرا این صفت دو مقداری است

و باید به دو صفت جدا تقسیم شود. همچنین projectid ها و project-feedback ها دوبار تکرار شده اند و این رابطه حتی primary key نیز ندارد که برای relational database model ضروریست.

ابتدا یک Pk به این رابطه اضافه می کنیم (اضافه کردن ستون جدید)

```
Alter Table customers Add ID Int primary key identity(1,1)
NotNull;
```

حالا به جای contact-person-and-role دو ستون جداگانه می گذاریم. ابتدا این ستون را تبدیل به contact-person کرده و ستون دیگری با نام contactprole اضافه می کنیم.

```
Go
Exec sp_rename 'customers.contact-person-and role', 'contact-person', 'COLUMN';
Alter Table customers Add contactprole varchar(۲۰۰);
```

حالا به دلیل تکرار Project یک جدول جدید Projects می‌سازیم و اطلاعات مربوط به Project را آنجا ذخیره می‌کنیم. برای اینکار ابتدا باید صفت‌های مربوطرا از customer پاک کرده و از آن جایی که هر customer چند پروژه دارد یک جدول Project ساخته که به customer، Foreign key بزند. (رابطه ۱ به N)

```
Alter Table Customers
Drop Column project1_id, project1_feedback,
            project2_id, project2_feedback;

Create Table Project(
P_ID Int Identity(1,1) Not Null,
C_ID Int Foreign key References customers(ID) Not Null,
FeedBack varchar(۲۵۵),
Primary key(P_ID, C_ID))
```

مرحله ۲ : 2NF : برای اینکه R در فرم 2NF باشد باید تمام صفت ها به طور کامل total به PK وابسته باشند. در این رابطه، contact-person، contactp-role، Phone number در واقع مربوط به آن contact person خاص است و به contact-person-id نیز وابسته اند و وابستگی total به ID ندارند. و با تغییر صرفا contact-person-id تغییر می‌کنند، پس ما می‌توانیم این ها را به یک جدول دیگر منتقل کنیم و contact-person-id

totaly dependent های pk را primary key آن جدول کنیم، تا همه صفات به pk

بشوند.

```
Alter Table Customers
Drop Column contact-person, contactp_role, phone-numbers;

Creat Table Contact-person(
CP_ID int Primary key Identity(1,1) Not null,
contact-person varchar(۳۰۰),
contactp-role varchar(۳۰۰),
Phone-number varchar(۱۲)).

AlterTable Customers
Add Foreign key (contact-person-id) References
Contact-person(CP-ID);
```

③ برون به 3NF : حالا همهٔ صفات به طور totaly به pk ها وابسته اند. درواقع همهٔ FD هایی که می توانیم بنویسیم همگی وابستگی به pk را بیان می کنند جز یکی از آن ها. این موضوع نقدی (دونفر با zip برابر مساوی City، Address برابر دارند). zip → city
zip → Address
بوجودی آورد:

ID → zip , zip → city , zip → Address

پس باید بازهم این جدول را بشکنیم. از آن جایی که zip یک صفت خاص است آن را به عنوان pk جدول جدید در نظر گرفته و zip در جدول customers را به zip در جدول جدید Foreig key می زنیم.

```
Create Table zip(
    zip-code varchar(5) primary key,
    city varchar(255)
    Address varchar(255))
Alter Table Customers
Drop column city, address

Alter Table customers
Add Foreign key (zip) References zip(zip-code)
```

حالا تمام FDهایی که وجود دارد تنها وابستگی به primary key را بیان می‌کنند و در واقع می‌توان گفت در فرم BCNF نیز هست!

Column Name	Data Type	Allow Nulls
name	varchar(255)	☑
industry	varchar(255)	☑
contact_person_id	int	☑
zip	varchar(5)	☑
ID	int	☐
		☐

Column Name	Data Type	Allow Nulls
P_ID	int	☐
C_ID	int	☐
FeedBack	varchar(255)	☑
		☐

Column Name	Data Type	Allow Nulls
CP_ID	int	☐
contact_person	varchar(300)	☑
contactp_role	varchar(300)	☑
phone_number	varchar(12)	☑
		☐

Column Name	Data Type	Allow Nulls
zip_code	varchar(5)	☐
city	varchar(255)	☑
address	varchar(255)	☑
		☐