# طراحی پایگاه داده‌ها

**نیم‌سال دوم ۰۳-۰۲**

دانشکده‌ی مهندسی کامپیوتر

استاد: مهدی آخی

تمرین چهارم

پاسخ‌دهنده: معین آعلی - ۴۰۱۱۰۵۵۶۱

## پاسخ مسئله‌ی ۱.

در این سوال فرض کرده‌ایم که همه ستون‌ها $Atomic$ هستند.

### الف

$$R = (X, Y, Z, S, T, U, V)$$

$$FDs = \begin{cases} S \to X \\ S \to V \\ T \to Y \\ X \to Y \\ XY \to TUZ \end{cases} = \begin{cases} S \to X \\ S \to V \\ T \to Y \\ X \to Y \\ XY \to T \\ XY \to U \\ XY \to Z \end{cases} = \begin{cases} S \to X \\ S \to V \\ T \to Y \\ X \to T \\ X \to U \\ X \to Z \end{cases}$$

همانطور که مشخص است، ستون $S$ می‌تواند باقی ستون‌ها را تعیین کند و از دیگر ستون‌ها نمی‌توان به $S$ رسید. پس حتما $S$ داخل کلید کاندید ما موجود است و چون باقی ستون‌ها از آن بدست می‌آیند، پس کلید کاندید ما فقط $\{S\}$ است.

همه ستون‌های دیگر به کلید کاندید وابستگی مستقیم یا غیرمستقیم دارند، چون کلید کاندید ما تک عضوی است، پس این دیتابیس $2NF$ است. اما دیتابیس از نوع $3NF$ نیست زیرا که به عنوان مثال ستون $Y$ به ستون $T$ که کلید کاندید نیست وابستگی دارد.

حال با شکستن روابط تعدی موجود، دیتابیس را به فرم نرمال مرتبه سوم تبدیل می‌کنیم:

$$R_1 = \{\underline{T}, Y\}$$

$$R_2 = \{\underline{S}, X, V\}$$

$$R_3 = \{\underline{X}, T, U, Z\}$$

حال چون فقط یک کلید کاندید داریم و دیتابیس از نوع $3NF$ هم هست، پس قطعا از نوع $BCNF$ هم هست.

## ب

$$R = (A, B, C, D, E)$$

$$FDs = \begin{cases} A \to BC \\ BC \to AD \\ D \to E \end{cases} = \begin{cases} A \to B \\ A \to C \\ BC \to D \\ BC \to A \\ D \to E \end{cases}$$

کلیدهای کاندید دیتابیس ما $\{A, BC\}$ هستند.

همه ستون‌های غیر کلید به کل کلید اصلی وابستگی تابعی دارد، پس از نوع $1NF$ نرمال شده است. اما چون ستون غیر کلیدی پیدا می‌شود که به ستون غیر کلید دیگری وابستگی تابعی دارد، پس از نوع $3NF$ نرمال شده نیست. به عنوان مثال ستون $E$ به ستون $D$ وابسته است.

پس جدول را به این شکل می‌شکنیم تا در هر رابطه‌ی تابعی، ستون سمت چپ یک کلید کاندید باشد. پس:

$$R_1 = \{\underline{A}, B, C\}$$

$$R_2 = \{\underline{B, C}, D\}$$

$$R_3 = \{\underline{D}, E\}$$

با توجه به این که در تمامی $FD$ها، ستون سمت چپ یک کلید کاندید است، پس دیتابیس ما از نوع $BCNF$ نرمال شده است.

## پاسخ مسئله‌ی ۲.

$R_1 = (X, Y, Z)$

$$FDs_1 = \begin{cases} Y \to X \\ XZ \to Y \\ X \to Z \end{cases}$$

ابتدا مجموعه $FDs$ را ساده می‌کنیم:

$$\longrightarrow FDs_1 = \begin{cases} Y \to X \\ X \to Y \\ X \to Z \end{cases} \quad \longrightarrow FDs_1 = \begin{cases} Y \to X \\ X \to YZ \end{cases}$$

### الف

این عبارت **نادرست** است. زیرا $Z$ خودش از $X$ بدست می‌آید و از آن نمی‌توان چیزی را بدست آورد.

### ب

این عبارت **نادرست** است. اگر رابطه $X \to Z$ را حذف کنیم، از روابط باقی‌مانده نمی‌توان آن را بدست آورد.

### ج

این عبارت **نادرست** است. زیرا خود $Z$ از $X$ بدست می‌آید. پس وجود $Z$ اضافی است و باید حذف شود.

### د

این عبارت **درست** است. با توجه به این که دیتابیس ما $1NF$ است و $CK$های ما همگی تک‌عضوی هستند، پس $2NF$ هم هست. با توجه به این که $X, Y$ کلیدهای کاندید ما هستند و ستون $Z$ به ستونی به جز کلیدهای کاندید وابسته نیست، پس دیتابیس $3NF$ هم هست. همچنین در سمت چپ تمامی $FD$های موجود، فقط کلید کاندید وجود دارد، پس دیتابیس ما در نهایت $BCNF$ است.

### ه

این عبارت **درست** است.

$R_2 = (A, B, C, D, E)$

$$FDs_2 = \begin{cases} A \to B \\ AB \to CD \\ D \to ABC \end{cases}$$

واضح است که کلیدهای کاندید این دیتابیس، $DE, AE$ هستند. همچنین دیتابیس اول هم دارای کلیدهای کاندید $X, Y$ هستند. پس کلید کاندید دیتابیس حاصل $Cartesian\ Product$ این دو، باید شامل یک کلید کاندید از اولی و یک کلید کاندید از دومی باشد. پس در کل ۲ × ۲ حالت داریم:

$CK = \{YAE, YDE, XAE, XDE\}$

## پاسخ مسئله‌ی ۳.

### الف

مراحل زیر را طی می‌کنیم:

۱. هر وابستگی تابعی در F را به طوری تجزیه کنید که در سمت راست فقط یک ستون وجود داشته باشد.
۲. صفات اضافی را با محاسبه بسته closure صفات سمت چپ به جز صفت مورد نظر، حذف کنید.
۳. وابستگی‌های تابعی زائد را حذف کنید.
۴. اطمینان حاصل کنید که سمت راست هر وابستگی تابعی فقط شامل یک صفت باشد.
۵. روابطی که با استفاده از تعدی به وجود آمده‌اند را حذف می‌کنیم.

### ب

پس از اجرای الگوریتم فوق، به مجموعه $FDs$ زیر می‌رسیم:

$$FDs = \begin{cases} msgID, wordPosition \rightarrow wordID \\ wordID \rightarrow wordText \\ wordText \rightarrow wordID \\ msgID \rightarrow visibility \\ msgID \rightarrow userID \end{cases}$$

### ج

این عبارت **درست** است. زیرا:

$$\begin{cases} msgID, wordID \rightarrow visibility \\ msgID \rightarrow userID \end{cases} \longrightarrow \Big\{ msgID, wordID, visibility \rightarrow userID$$

### د

می‌دانیم که کلید کاندید ما $\{msgID, wordPosition\}$ است. در ابتدا $Partial\ Dependency$ موجود در $wordID$ و سپس $Transitive\ Dependency$ موجود بین $wordText$ و کلید کاندید را از بین می‌بریم. پس در کل به ۴ رابطه نیاز داریم:

$R_۱ = \{\underline{msgID}, visibility\}$
$R_۲ = \{\underline{msgID}, userID\}$
$R_۳ = \{\underline{msgID, wordPosition}, wordID\}$
$R_۴ = \{\underline{wordID}, wordText\}$

با توجه به اینکه که مجموعه کلیدهای کاندید ما تک عضوی است، پس دیتابیس ما $BCNF$ هم هست.

در نتیجه چون نرمال‌سازی ما در سطح $BCNF$ است، پس $LossLess$ است و $Dependency$ها را حفظ می‌کند.

### ه

$$R = R_۱ \bowtie_{msgID} R_۲ \bowtie_{msgID} R_۳ \bowtie_{wordID} R_۴$$

## پاسخ مسئله‌ی ۴.

### الف

۱. INSERT

به عنوان مثال، ستون‌های $itemID$ و $memberID$ و... فقط در سمت راست $FD$ مربوط به امانات وجود دارند که این به معنی آن است که برای درج کردن یک عضو یا یک کالا یا نویسنده یا ...، حتما باید رابطه امانت گرفتن و یا نویسندگی برای آن‌ها برقرار باشد.

۲. DELETE

همان مواردی که در قسمت قبل برای $INSERT$ مشکل ایجاد کرده بودند، برای حذف کردن هم مشکلاتی به وجود می‌آورند. به عنوان مثال اگر یک رکورد از امانت‌ها را حذف کنیم، کالا یا عضو یا... مربوط به آن هم ممکن است به کل از دیتابیس حذف شود اگر در رکورد دیگری وجود نداشته باشد.

۳. UPDATE

اگر بخواهیم ستون‌های سمت راست $FD$ها را تغییر دهیم، نیازمند این هستیم که تمامی رکوردهای مربوط به آن مقدار خاص از یک ستون را تغییر دهیم که پروسه‌ی هزینه‌بر و زمان‌بری است.

### ب

می‌دانیم که در رابطه‌ی فعلی، ستون $loanID$ تنها PrimaryKey و تنها CandidateKey است. پس این رابطه $2NF$ است. اما چون در روابطمان خاصیت تعدی را داریم، باید آن را به $3NF$ تبدیل کنیم و ستون‌ها را بشکنیم.

رابطه‌های جدید به این صورت هستند:

$R_1 = \{\underline{memberID}, memberName, memberAddress, memberType\}$
$R_2 = \{\underline{loanID}, loanDate, dueDate, returnDate, lateDate, memberID, itemID\}$
$R_3 = \{\underline{itemID}, itemTitle, itemType, itemFormat, authorID\}$
$R_4 = \{\underline{authorID}, authorName\}$
$R_5 = \{\underline{itemType, memberType}, lateFee\}$

حال چون تمامی روابط فوق به صورت $PK \to X$ است، پس دیتابیس فوق به صورت $BCNF$ نیز می‌باشد و نیاز به تغییر ندارد.

### ج

در طراحی جدید می‌توان به راحتی موارد جدید را افزود. به عنوان مثال اگر بخواهیم $item$ جدیدی را اضافه کنیم، فقط کافیست آن را به جدول $items$ اضافه کنیم.

همچنین اگر بخواهیم یک ویژگی جدید به یک موجودیت اضافه کنیم، کافیست که فقط جدول مربوط به آن را تغییر دهیم و روابط دیگر دچار تغییر نمی‌شوند.

اگر بخواهیم یک رابطه جدید هم تعریف کنیم، تنها روابطی که با آن‌ها درگیر هستند نیازمند تغییر و اضافه کردن $FK$ هستند، نه همه‌ی روابط!

خلاصه که آنومالی‌های قسمت الف برطرف می‌شوند.

## پاسخ مسئله‌ی ۵.

### الف

آنومالی‌های رابطه‌ی فعلی:

۱. INSERT
در رابطه‌ی فعلی اگر بخواهیم یک کتاب اضافه کنیم باید حتما آن را قرض بدهیم و در جدول درج کنیم که این اشتباه است و نوعی آنومالی رایج است. همچنین برای نویسنده هم همچین مشکلی وجود دارد.

۲. UPDATE
همانند سوال قبلی اگر بخواهیم مقدار یکی از ستون‌های جدول را تغییر دهیم، باید در تعدادی زیادی رکورد این مقدار را تغییر دهیم که کار هزینه‌بری است.

۳. DELETE
همانند سوال قبلی اگر یک سطر را از جدول فعلی حذف کنیم، اطلاعات مربوط به یک نویسنده یا یک کتاب حذف می‌شود که ما این را نمی‌خواهیم.

حال برای تبدیل به $2NF$، اقدام به شکستن $Partial\ Dependency$ها می‌کنیم:

$Borrower(\underline{borrowerID}, borrowerName)$

$Borrow(\underline{borrowID}, dueDate)$

$BorrowBook(\underline{borrowID}, \underline{borrowerID}, \underline{bookID},)$

$BorrowerFeedback(\underline{borrowerID}, \underline{bookID}, feedback)$

$Book(\underline{bookID}, writer, genre)$

اکنون آنومالی مربوط به $INSERT, DELETE$ حل شد، حال با تبدیل به $3NF$ آنومالی مربوط به $UPDATE$ را حل می‌کنیم:

$bookWriter(\underline{bookID}, writer)$

$writerGenre(\underline{Writer, Genre})$

$Borrower(\underline{borrowerID}, borrowerName)$

$Borrow(\underline{borrowID}, dueDate)$

$BorrowBook(\underline{borrowID}, \underline{borrowerID}, \underline{bookID},)$

$BorrowerFeedback(\underline{borrowerID}, \underline{bookID}, feedback)$

### ب

هیچ کدام از وابستگی‌ها از بین نرفته است، پس هردو دارای $Dependency\ Preserving$ هستند.

## ج

همچنان برای $INSERT, DELETE$ آنومالی وجود دارد، به عنوان مثال برای حذف یک کتاب مجبوریم که تمامی فیدبک‌های آن را یکی‌یکی حذف کنیم.

حال برای تبدیل به $BCNF$، رابطه‌ی $BorrowFeedback$ را می‌شکنیم:

$BorrowFeedback(\underline{borrowerID}, Feedback)$

$BookFeedback(\underline{Feedback}, bookID)$

با اضافه کردن دو جدول فوق، رابطه‌ی

$(BorrowerID, BookID) \rightarrow FeedBack$ از بین رفت و $Dependency\ Preserving$ نقض شد.

## د

به عنوان مثال اگر در جدول $BorrowBook$ به جای استفاده از $PK$، از $BookID$ استفاده کنیم و جوین بزنیم، آنگاه اگر یک کتاب خاص چند بار به افراد متنوع قرض داده شده باشد، دیتاهایی را از دست میدهیم و دیگر $LossLess$ نیست!