Бази Данни

Явор Николов

22 март 2003

Резюме

Този документ първоначално беше предназначен за подготовка по въпроса по бази данни за държавния изпит на специалност информатика към СУ Кл. Охридски, октомври 2002 година. Става дума за релационна алгебра, релационно смятане и нормални форми.

Това е второ издание, в което са отстранени някои грешки и са направени някои други незначителни изменения.

Вероятно има още грешки, надявам се не много – не отговарям за последствията от тях. Ако откриете такива или имате някакви други забележки и предложения, пишете на javor_nik@bulgaria.com.

Информацията се базира основно на книгата [2]. Почти целият материал е сравнително директен превод на текст от книгата, така че не претендирам за някакво особено творчество.

Съдържание

1 Релационен модел на данни (РМД) 3
- 1.1 Предисловие . . . 3
- 1.2 Дефиниции . . . 4
- 1.3 Реализация . . . 5
- 1.4 Операции върху РМД. Заявки . . . 6

2 Релационна алгебра (РА) 6
- 2.1 Ограничения на РА . . . 6
- 2.2 Операнди и операции на РА . . . 7
- 2.3 Допълнителни операции . . . 8

3 Релационно смятане 10
- 3.1 Релационно смятане върху домени . . . 10
  - 3.1.1 Формули на релационното смятане . . . 10

3.1.2 Релационно смятане върху домени (DRC) 12
3.1.3 От PA към DRC 12
3.1.4 Безопасни DRC формули 12
3.1.5 Привеждане на безопасни DRC към PA 14
3.2 Релационно смятане върху кортежи 14
3.2.1 Формули 14
3.2.2 Безопасни TRC формули 15
3.2.3 От PA към безопасно TRC 16
3.2.4 От TRC към DRC 16
3.3 Обобщение 16

4 Нормални форми **17**
4.1 Недостатъци на схемата на базата данни 17
4.2 Функционални зависимости 18
4.2.1 Дефиниции 18
4.2.2 Конвенции за означенията 19
4.2.3 Логически изводи от функционални зависимости . . . 19
4.2.4 Ключове 19
4.2.5 Аксиоми на Армстронг за функционални зависимости 20
4.2.6 Надеждност на аксиомите на Армстронг 20
4.2.7 Допълнителни правила за извод 21
4.2.8 Пълнота на аксиомите на Армстронг 22
4.2.9 Изчисляване на затваряне 22
4.2.10 Минимални покрития 23
4.3 Декомпозиции на релационни схеми 24
4.3.1 Декомпозиции, запазващи съединението 25
4.3.2 Декомпозиции, запазващи функционалните зависимости 26
4.4 Нормални форми за релационни схеми 28
4.4.1 Дефиниции 28
4.5 Декомпозиция в BCNF и 3NF 30
4.5.1 Декомпозиция в BCNF с беззагубно съединение 30
4.5.2 Декомпозиции в 3NF, които запазват зависимостите . 32
4.6 Многозначни зависимости 34
4.6.1 Аксиоми за функционални и многозначни зависимости 34
4.6.2 Допълнителни правила 35
4.6.3 Базис на зависимост 36
4.6.4 Беззагубни съединения 37
4.7 Четвърта нормална форма 37
4.8 Вложени многозначни зависимости 38

1. Релационен модел на данни (РМД)

1.1. Предисловие

Релационният модел на практика е господстващият модел данни в момента. Може би най-важната причина за неговата популярност е, че той поддържа мощни, но прости и декларативни езици, с които се изразяват операциите над данните. Тези способности се дължат на факта, че за разлика от конкурентни модели, РМД е *стойностно-ориентиран*.[1] Този факт, от своя страна, води до възможността да дефинираме операции върху релации, чиито резултати отново са релации. Тези операции може да се комбинират и да се използват каскадно[2], като се използва нотация, наречена "релационна алгебра".

За сравнение, езиците, базирани на обектно-ориентирани МД, нямат операции, които могат да се композират лесно. Причината е двустранна:

1. Независимо от МД, релациите са удобен начин за изразяване на резултати. При езиците на обектно-ориентираните МД, резултатът може да е от друг тип. Така че в общия случай резултати от операции в тези МД, не могат да се използват като операнди на други операции.
2. При МД, които поддържат АТД,[3] има още едно препятствие: резултатът от операция често е съвсем нов тип. На този тип тип първо трябва да му се дефинират операциите,[4] преди да може да стане операнд на друга операция.

Физическата реализация на РМД не е така ефективна, както на някои други МД (йерархичен, мрежови), но поради неговата гъвкавост и привлекателност за крайния потребител, успява да се наложи. Все пак, РМД се реализира достатъчно ефективно на физическо ниво и може да се каже, че е максималната абстракция на текущата технология в хардуера.[5]

[1] "Стойностно ориентиран" означава, че не се поддържа идентификация на обектите. Това е обратното на "обектно-ориентиран". Положението е следното: обектно-ориентираните МД имат добра интеграция с DML (Data Manipulation Language) и с host езика, но не се погаждат добре с декларативни езици. При стойностно-ориентираните модели е точно обратното, т.ч. РМД поддържа декларативни езици, но не се интегрира добре и DML с хост езика. Все пак практиката е показала, че потребителите предпочитат декларативни езици, които са по-лесни за използване и не изискват от потребителя да знае всички детайли за базата данни, за да прави полезни заявки към нея.

[2] Т.е. може да имаме вложение на операциите.

[3] Абстрактни Типове Данни

[4] Преди това той всъщност още не е тип данни.

[5] По отношение на хардуера: от около 70-те години досега нищо принципно не се е променило, устройствата за съхраняване на данни – дисковете, принципно са си същите. Докато това е така, няма изгледи да бъде измислено нещо по-добро от РМД.

Дефиниция 1.1 (Модел данни (МД)). *Моделът данни* е математически формализъм, който се състои от 2 части:

1. Нотация за описание на данните;
2. Множество от операции, използвани за манипулиране на данните.

Ще започнем с нотацията за описание на данните, в основата на която лежи математическата концепция за релация. По-нататък ще продължим с операциите, които ще дефинираме чрез езика на релационната алгебра и два вида релационно смятане. В крайна сметка ще се окаже че тези езици са еквивалентни (в случая на релационните смятания се прибягва до техни "безопасни" подмножества).

1.2. Дефиниции

Дефиниция 1.2 (Домен). *Доменът* просто е множество от стойности, подобно на тип данни (но не е тип, защото няма операции).

Дефиниция 1.3 (Релация). Нека са дадени домените $D_1, \ldots, D_k$. *Релация* наричаме всяко подмножество $R \subseteq D_1 \times \cdots \times D_k$.[6]

Т.е. това е множество от k-*кортежи* (или просто *кортежи*), както се наричат елементите на горното декартово произведение.[7]

Тук $\boldsymbol{k}$ наричаме *размерност* или *степен* на релацията.

Дефиниция 1.4 (Релацията като таблица, атрибути). Полезно е релацията да се разглежда като *таблица*:[8] Всеки ред е кортеж и всяка колона съответства на един компонент от съответния домен. На колоните често им се дават имена, наречени *атрибути*.

Дефиниция 1.5 (Релационна схема). *Релационна схема* наричаме множеството от имената на атрибутите в една релация. Ако релацията има име REL и атрибути $A_1, \ldots, A_k$, тогава означаваме релационната схема по следния начин: $REL(A_1, \ldots, A_k)$. Ако няма име, означаваме с $\{A_1, \ldots, A_k\}$ или $(A_1, \ldots, A_k)$. При горната дефиниция за релация имената може да се повтарят, защото редът им е фиксиран.

Алтернативна дефиниция за релация

Математическата дефиниция за релация като "множество от списъци" не винаги е достатъчно удобна. Сега ще дадем една друга дефиниция за релация, която понякога е доста полезна. Ако присъединим *уникални* атрибутни имена към колоните на релацията, тогава редът на колоните няма значение.

[6] В този смисъл и празното множество $\emptyset$ също е релация – празната релация.

[7] Информация за декартовото произведение има в [1].

[8] Понякога на релациите им казват *плоски таблици* (*flat tables*).

Така е възможно да се разглеждат кортежите като *изображения* от имената на атрибутите към стойностите в домените на атрибутите. В този смисъл релацията представлява *множество от изображения* от този вид.

С тази промяна в гледната точка е възможно различни таблици да представят една и съща релация, въпреки че релациите са различни от математическа гледна точка.

Пример 1.1. Да разгледаме кортежа (Мусала, Рила, 2925) за схемата PeakInfo(Peak, Mountain, Height). Този кортеж като изображение μ се дефинира чрез: $\mu(Peak) =$ Мусала, $\mu(Mountain) =$ Рила и $\mu(Height) = 2925$. □

Ако μ е кортеж и X е множество от атрибути, то с $\mu[X]$ ще означаваме компонентите на μ, които са сред атрибутите в X.

Дефиниция 1.6 (Схема на релационна база от данни). *Схема* на РМД наричаме набора от всички релационни схеми, използвани за представяне на информацията в базата данни.

Тъй като става дума за бази данни, които физически се съхраняват в някаква крайна памет, обикновено е безполезно да се работи с безкрайни релации.[9] Освен ако изрично не кажем друго, когато говорим за релации ще подразбираме, че са крайни.

1.3. Реализация на релационната база от данни

В релационния модел за първи път се появява идеята за "независимост на данните". В основни линии този принцип скрива вътрешната структура на имплементацията на базата данни от външната концептуална структура.

И йерархичният и мрежовият модел изключително използват указатели при имплементацията на връзките между записите (същностите). Тъй като релационният модел е стойностно ориентиран, при него не е така. Той също използва указатели, но като помощно средство за да търси по-ефективно записи, а не за да ги свърже.

В релационния модел данните се организират в плоски таблици, състоящи се от записи и файловите структури по подобен начин са били организирани във файлове със записи, представящи кортежите с данни. Това, че на концептуално ниво разглеждаме атрибутите в някакъв ред, изобщо не означава, че точно в този ред се организират като записи на диска.[10] Физическото ниво се реализира, така че да се оптимизират бързодействието, използваната памет и т.н. Отделно от това по се реализира изображението на тези физически структури в концептуалния логически модел. За да бъде подобрена

[9] Това, обаче, е възможно, ако разглеждаме нещата от чисто теоретична гледна точка.

[10] Например може да се сортират в зависимост от размера на полетата.

ефективността на заявките, се използват различни структури данни и *механизми за индексиране*, като например хеширане и B^+ дървета (и двете имат статични и динамични варианти).

1.4. Операции върху РМД. Заявки

Някои неща за операциите над РМД споменахме в предисловието (1.1). В РМД има два вида нотации за изразяване на операции върху релации:

1. *Алгебрическа*, наречена *релационна алгебра*, където заявките се изразяват чрез прилагане на специални операции към релации.
2. *Логическа*, наречена релационно смятане, където заявките се изразяват чрез писане на логически формули, които кортежите в резултата трябва да удовлетворяват.

Тези две нотации имат еквивалентна изразна мощ, т.е. всяка от тях може да изрази произволна заявка, която другата може, но не повече.

2. Релационна алгебра (РА)

2.1. Ограничения на релационната алгебра

Да си представим следната възможност: да можем да използваме произволна програми като заявки. Тогава обаче възникват следните проблеми:

a) Трябва да знаем всичко за използваните физически структури данни и

b) Трябва да пишем код, който зависи от конкретно избраните структури.

По този начин от една страна се отказваме от концепцията за независимост на данните и от друга не можем да настройваме и оптимизираме физическите структури (ако го направим, ще трябва да пренапишем всичките заявки отново). При тези езици обикновено не се налага да се правят оптимизации на заявките (а и до голяма степен това е невъзможно).

Затова почти винаги се предпочитат езици за заявки, които "говорят" само в термините на модела данни, а не за някоя конкретна имплементация на модела. Но съгласявайки се да работим само с релации (които са структурата , която представя данните в РМД), възниква друг проблем. Искаме операциите да имат *ефективна имплементация*. Ако разрешим прекалено голям набор от възможни операции това е невъзможно (не можем, например, да оптимизираме произволна програма на Prolog). Затова РА и другите езици на РМД имат ограничена и все пак достатъчна изразителна мощ т.е.:

1. Позволяват задоволително решение на оптимизационния проблем;

2. Осигуряват достатъчно богат език, че да може да изразяват достатъчно неща, за да бъде СУБД-то полезно.

Тези езици обикновено не могат да се справят с транзитивното покритие на бинарна релация. Друга особеност е, че релациите са *крайни* и очакваме резултатите от операциите да са крайни релации. Това създава определени трудности при дефинирането на езиците за заявки, например допълнението на една крайна релация е безкрайно. (Все пак проблемът се решава, като се дефинират определени критерии за "безопасност".)

2.2. Операнди и операции на РА

Операндите на РА са два вида:

– *константни* релации;

– *променливи*, представящи релации с фиксирана размерност.

Има 5 основни операции, с които се дефинира релационната алгебра. Първите три от тях имат стандартния математически смисъл и изисквания върху размерността на операндите и няма да го дискутираме.[11] Тези дефинициите са валидни за математическата дефиниция за релация като множество от списъци, т.е. не изискваме атрибутни имена и редът на колоните има значение.[12]

1. *Обединение.* $R \cup S$;
2. *Разлика.* $R - S$;
3. *Декартово произведение.* $R \times S$;
4. *Проекция.* $\pi_{i_1,\ldots,i_m}(R)$,
   където R е с размерност k, $i_j \in \{1,\ldots,k\}$, наричаме проекция на R върху компонентите $i_1,\ldots,i_m$ и дефинираме така:

   $$\pi_{i_1,\ldots,i_m}(R) = \{a_1 \ldots a_m \mid \exists\, b_1 \ldots b_k \in R :\ a_j = b_{i_j},\ j = 1,\ldots,m\}$$

   Например: $\pi_{3,1}(R)$ – това е релацията, която се състои от първата и третата колона на R.
   Ако имаме атрибути, може вместо номера да използваме атрибутни имена, например за $R(A,B,C,D)$ може да напишем $\pi_{C,A}(R)$.
5. *Селекция.* Нека F е логическа формула, включваща:
   (1) *Операнди*, които са *константи* или *номера на компоненти*: компонент i се представя като $\$i$;

[11] Информация за това има в [1].

[12] Разбира се, нищо не пречи да използваме и другата дефиниция, тъй като винаги можем да фиксираме някакъв ред на атрибутите.

(2) Операции за *аритметично сравнение*:

$$<, \leq, =, \geq, >, \neq$$

(3) *Логически* операции. $\vee$ (и), $\wedge$ (или) $\neg$ (не).

Селекцията се дефинира така:

$$\sigma_F(R) = \{\ \mu \in R \mid F[\,\$1/\mu[1], \ldots, \$k/\mu[k]\,] = true\ \},$$

където с $F[\,\$1/\mu[1], \ldots, \$k/\mu[k]\,]$ сме означили формулата F, в която всяко срещане на $\$i$ е заменено с $i^{-\text{тия}}$ компонент на μ. Предполагаме, че R е с размерност k.
Пример: $\sigma_{\$2>\$3}(R)$.

2.3. Допълнителни алгебрични операции

Дефинирани са и някои допълнителни операции в РА, които могат да бъдат изразени чрез основните.

- *Сечение.* $R \cap S = R - (R - S)$.
- *Частно.* Нека R е с размерност r, S е с размерност s, $r > s$ и $s \neq 0$. Тогава:

$$R \div S = \{\, a_1 \ldots a_{r-s} \mid (\forall\, a_{r-s+1} \ldots a_r \in S)\ a_1 \ldots a_r \in R \,\}.$$

Т.е. $R \div S$ е такава релация, че $(R \div S) \times S = R$. Как се дефинира частното чрез основните операции? Нека $T = \pi_{1,\ldots,r-s}(R)$. Това са всички кортежи, които евентуално биха могли да бъдат в $R \div S$. Тогава $(T \times S) - R$ е множеството от всички $r-$кортежи, чиито "леви части" (отрязани, докъдето трябва) не са в $R \div S$. Множеството от тези "леви части" е $V = \pi_{1,\ldots,r-s}((T \times S) - R))$. Така получаваме:

$$T - V = R \div S.$$

И окончателният израз само от основни операции е:

$$R \div S = \pi_{1,\ldots,r-s}(R) - \pi_{1,\ldots,r-s}((\pi_{1,\ldots,r-s}(R) \times S) - R)).$$

Това не е единствено възможният израз. Тук редът на изпълнение на основните операциите при пресмятане на частното може да има значение за ефективността на изпълнението. Например може първо да пресметнем декартовото произведение и изобщо да следваме схемата,

която се вижда от горния израз. Обаче може и да не пресмятаме цялото декартово произведение, а за всяка "лява част" от кортеж на R да видим дали всичките ѝ "слепвания" с кортеж от S, дава пак кортеж от R.

Проблемът на декартовото произведение е, че е тежка операция и резултатът може да е огромна релация.

– *Съединение (θ − съединение):* $R \underset{\$i\,\theta\,\$j}{\bowtie} S$. Нека R е с размерност r, S - с размерност s, а θ е една от шестте операции за аритметично сравнение. Тогава дефинираме:

$$R \underset{\$i\,\theta\,\$j}{\bowtie} S = \sigma_{\$i\,\theta\,\$(r+j)}\left(R \times S\right),$$

Ако θ е "=", операцията се нарича *еквисъединение*.

- Кортежи, следата на които се губи при θ − съединение се наричат *висящи кортежи*. [13]
- Можем вместо номера на компоненти да използваме имена на атрибути. В този случай имената се наследяват. Ако има съвпадащи имена на атрибути, проблемът се решава или чрез преименуване или чрез използване на префикс – името на релацията и "."; например $R.A < S.A$.

– *Естествено съединение.* $R \bowtie S$. Операцията е приложима, само когато и двете релации имат имена на атрибутите. Извършва се съединение между атрибутите с едни и същи имена. Как се изчислява:

1. Пресмята се $R \times S$.
2. За всеки атрибут A, който именува едновременно колона в R и колона в S, избираме от $R \times S$. тези кортежи, чиито стойности съвпадат по колоните за $R.A$ и $S.A$.
3. За всеки атрибут A горе махаме колоната $S.A$ и преименуваме оставащото $R.A$ на A.

Формално, ако $A_1, \ldots, A_k$ са всички общи имена на атрибути в R и S, тогава:

$$R \bowtie S = \pi_{i_1,\ldots,i_m}\left(\sigma_{R.A_1=S.A_1 \wedge \cdots \wedge R.A_k=S.A_k}\left(R \times S\right)\right),$$

където $i_1, \ldots, i_m$ е списъкът от всички компоненти на $R \times S$ в този ред без $S.A_1, \ldots, S.A_k$.

[13] Става дума за кортежи от едната релация, за които няма нито един съответен кортеж от другата релация, така че $\$i\,\theta\,\$(r+j)$ да е истина. При съединение информацията за тях се губи в новата релация.

– *Полусъединение.*

$$R \ltimes S = \pi_R (R \bowtie S) = R \bowtie \pi_{R \cap S} (S) .$$

Така на практика доказахме следното.

Теорема 2.1. *Допълнителните операции, които дефинирахме имат еквивалентни изрази от PA, в които участват само петте основни операции.* □

3. Релационно смятане

Форма на логиката, наречена релационно смятане, лежи в основата на повечето комерсиални езици за заявки, базирани на релационния модел.

3.1. Релационно смятане с променливи върху домени, Domain Relational Calculs, DRC

3.1.1. Формули на релационното смятане

Формулите са изрази, които определят релации, възможно и безкрайни. Всяка формула има множество от *"свободни"* и множество от *"свързани"* променливи. Възможно е едно и също име на променлива да се среща както свободно, така и свързано в една и съща формула. Ще разграничаваме свързаните участия от свободните участия на променливите.

Дефиниция 3.1. *Релационна схема за формула* е множеството от атрибути, съответстващи на свободните променливи във формулата.

Дефиниция 3.2 (формула (с променливи върху домени)). Ще дефинираме индуктивно формула и заедно с това ще определим свободните и свързаните променливи в нея. Първите две клаузи са основата и дефинират т.нар. "атомарни формули".

1. Всеки *литерал* $p(X_1, \ldots, X_n)$, където p е предикатен символ и $X_1, \ldots, X_n$ са *променливи* или *константи*, е формула. Всяко срещане на променливите сред $X_1, \ldots, X_n$ е свободно. Формулата представя релацията, която се пресмята от Algorithm.3.1 от книгата [2] за правило, чието тяло се състои от единствената подцел $p(X_1, \ldots, X_n)$.[14]

[14]Това сме го учили и на лекции. Доколкото си спомням, релацията беше еквивалентна на проекция върху свободните променливи, селекция по равенство на съответните компоненти с константите и по равенство за различните компоненти, за които е еизползвана една и съща променлива. Всичко това се прилага върху релацията на предикатния символ p – той си има съответна релация от екстенсионалната база данни.

2. Всяко аритметично сравнение $X\,\theta\,Y$ е формула, където X и Y са променливи или константи, а θ е една от шестте аритметични операции за сравнение. Срещанията на X и Y, ако са променливи, са свободни. В много случаи $X\,\theta\,Y$ представя безкрайна релация – множеството от всички двойки (X, Y), които са в релация θ. Затова ще изискваме такива формула да са свързани с логическо "И" с друга формула, която дефинира крайна релация; в този случай $X\,\theta\,Y$ може да се разглежда като операция за селекция.
3. Ако F_1 и F_2 са формули, то:
   а) $F_1 \wedge F_2$ ("и двете F_1 и F_2 са истина.")
   б) $F_1 \vee F_2$
   в) $\neg F_1$ (отречена формула)

   също са формули със съответния интуитивен смисъл. Срещанията на променливите са свободни или свързани $\Longleftrightarrow$ са свободни или свързани в съответното им F_1 или F_2.[15]
4. Ако F е формула, в която X се среща свободно поне веднъж, то

$$(\exists\, X)F$$

   е формула с интуитивния смисъл: има поне една стойност на X, с която като заместим всяко свободно участие на X, получаваме истинна формула. Всички свободни участия на X във F се свързват от квантора $\exists$ и са свързани във формулата $(\exists\, X)F$.
5. Ако F е формула, в която X се среща свободно поне веднъж, то

$$(\forall\, X)F$$

   е формула със съответния интуитивния смисъл. Всички свободни участия на X във F се свързват от квантора $\forall$ и са свързани във формулата $(\forall\, X)F$.
6. Ако F е формула, то (F) е формула, при която се запазва същото значение и свързаноста на променливите.

При липса на () редът с намаляващ надолу приоритет е:

(1) $\neg$, $(\forall X)$, $(\exists X)$ са с най-висок приоритет и дясна асоциативност.

(2) $\wedge$ – лява асоциативност

(3) $\vee$ – лява асоциативност

[15] Може и, например, X да е свободно на едно място и свързано на друго.

3.1.2. Релационно смятане върху домени (DRC)

Формулите могат да бъда използвани да изразяват заявки по прост начин. Всяка формула с една или повече свободни променливи дефинира релация, чиито атрибути съответстват на тези свободни променливи.

Ще пишем $F(X_1, \ldots, X_n)$, за да означим, че свободните променливи на F са точно $X_1, \ldots, X_n$.

Тогава заявката, или изразът, означен от F е:

$$\{ X_1 \ldots X_n \mid F(X_1, \ldots, X_n) \} \tag{3.1}$$

това е множеството от кортежи $a_1 \ldots a_n$, които правят формулата $F[X_1/a_1, \ldots, X_n/a_n]$ истина.

Дефиниция 3.3 (Доменно релационно смятане (DRC)[16]). Езикът за заявки, състоящ се от изрази от вида (3.1) се нарича *доменно релационно смятане*.

Прилагателното "доменно" в случая означава, че променливите са компоненти на кортежи, т.е. променливите означават произволни членове от домените на техните компоненти. Тази форма на логиката се различава от кортежното релационно смятане, където променливите означават цели кортежи.

Трябва да се отбележи, че релациите, дефинирани чрез DRC не винаги са *крайни*, например $\{ XY \mid \neg p(X, Y) \}$ задава безкрайна релация. Затова се дефинира едно подмножество на DRC формулите, които са "безопасни".

3.1.3. От PA към DRC

Теорема 3.1. *Всяка заявка, която може да се изрази в PA може да се изрази и в DRC.*

Доказателство. Конструктивно доказателство с индукция по броя на операциите в алгебричния израз... □

3.1.4. Безопасни DRC формули

Както споменахме, DRC формулите понякога дават безкраен резултат. Затова се налага да се ограничим до едно тяхно подмножество, наречено "безопасни" формули. Освен, че трябва да дефинират само крайни релации, тези формули трябва да отговарят и на някои други изисквания. Трябва лесно да можем да определим, само от вида на формулата, дали е "безопасна"

[16] DRC = Domain Relational Calculs

или не. Формулите, които са изразими в реалните езици за заявки, базирани на релационното смятане, са "безопасни".

Интуитивно "безопасните" DRC формули изглеждат като резултат от прилагане на последователност от безопасни нерекурсивни дейталог правила.

Дефиниция 3.4 (Безопасни DRC формули). *Безопасните DRC формули* трябва да отговарят на следните изисквания:

1. Няма употреба на квантора $\forall$. Това не влияе на изразителната сила на езика, т.к. $(\forall X)F$ е еквивалентно на $\neg(\exists X)\neg F$.
2. При всяко използване на операцията $\vee$, двете свързани формули, да кажем $F_1 \vee F_2$, имат един и същ набор свободни променливи, т.е. те са във вида:
$$F_1(X_1, \ldots, X_n) \vee F_2(X_1, \ldots, X_n).$$
3. Да разгледаме произволна максимална[17] подформула, състояща се от конюнкция на една или повече формули:
$$F_1 \wedge \cdots \wedge F_m.$$
Тогава всички променливи, които участват свободно в която и да е от тези F_i трябва да бъде *ограничена* (лимитирана) в следния смисъл:
   a) Една променлива е ограничена, ако е свободна в някое F_i, където F_i не е аритметично сравнение и не е отречена.
   b) Ако F_i е $X = a$ или $a = X$, където a е константа, тогава X е ограничена.
   c) Ако F_i е $X = Y$ или $Y = X$ и Y е ограничена променлива, то X е ограничена.
4. Операцията $\neg$ може да се прилага само в член на конюнкция от вида, описан в правило (3). В частност, подформулата $\neg G$ нарушава "безопасността", освен ако не е част от по-голяма подформула
$$H_1 \wedge \cdots \wedge H_i \wedge \neg G \wedge I_1 \wedge \cdots \wedge I_j\,,$$
удовлетворяваща (3), така че поне една от $H^{-\text{те}}$ и $I^{-\text{те}}$ е положителна (не е отречена).

Например, всяка формула от доказателството на Теорема 3.1 (което всъщност не написахме) е безопасна.

[17] В случая се има в предвид, че няма други конюнкти отляво или отдясно на тази формула.

3.1.5. Привеждане на безопасни DRC към PA

Можем да докажем, че всяка безопасна формула има израз от PA, дефиниращ същата релация. Има и много *небезопасни* (т.е. опасни) формули, които също имат еквивалентни изрази от PA, но в общия случай не можем да кажем кои имат и кои не.

Теорема 3.2. *Множествата от функции, изчислени от изрази на PA, от безопасни нерекурсивни дейталог програми и от безопасните DRC формули, съвпадат.*

Доказателство. От Теорема 3.1 и нейното доказателство $\Longrightarrow$ PA $\subseteq$ безоп.DRC. От няколко теореми за дейталог модела (3.7 и 3.8 в [2]) $\Longrightarrow$ PA $\Longleftrightarrow$ безоп.дейталог.

Остава да се докаже, че безоп.DRC $\subseteq$ безоп. нерекурс. дейталог. Това става по индукция по броя на операциите в безопасната DRC формула... $\square$

3.2. Релационно смятане с променливи върху кортежи, Tuple Relational Calculs, TRC

TRC е вид релационно смятане, където променливите означават кортежи, а не компоненти от кортежи. Ще означаваме $i^{\text{-тия}}$ компонент на μ с $\mu[i]$. Ако е известен атрибут A за този компонент, може да пишем и $\mu[A]$.

3.2.1. Формули

Дефиниция 3.5. Дефиницията на TRC е рекурсивна и прилича на тази за DRC.

1. Ако p е име на предикат и μ е променлива за кортеж, тогава $p(\mu)$ е атомарна формула. Смисълът е: "кортежът μ е в релацията за p".
2. $X\,\theta\,Y$ е атомарна формула, ако θ е операция за аритметично сравнение и X и Y са константи или означения за компоненти; последните са от вида $\mu[i]$ за някоя кортежна променлива μ и номер на компонент или атрибут i.
3. Индуктивната дефиниция продължава както в DRC. Ако F_1 и F_2 са TRC формули и μ е кортежна променлива, която се появява свободно в F_1, то следните са TRC формули с очевиден смисъл и множества от свободни и свързани променливи:
   a) $F_1 \wedge F2$
   b) $F_1 \vee F_2$
   c) $\neg F_1$

d) $(\exists\, \mu) F_1$

e) $(\forall\, \mu) F_1$.

Дефиницията за релация на TRC формула е аналогична на тази при DRC.

Дефиниция 3.6. *Релацията за една TRC формула*а има по един компонент за всеки компонент на всяка свободна кортежна променлива, който действително е означен в F.[18] Стойността на релацията е множеството от кортежи, чиито стойности, когато бъдат заместени в съответните компоненти на кортежните променливи, правят F истина.

Дефиниция 3.7. *Заявка* на TRC е израз от вида:

$$\{\, \mu \mid F(\mu) \,\},$$

където μ е *единствената* свободна променлива на F. Този израз дефинира релацията на множеството от всички кортежи, които правят F истина.

Понякога ще използваме нотацията $\mu^{(i)}$, за да означим, че μ е от размерност i, тъй като това не винаги е ясно от контекста.

3.2.2. Безопасни TRC формули

Както в DRC и тук може да пишем формули като $\neg r\,(\mu)$, които означават безкрайни релации. И тук най-полезният подход е да дефинираме ограничено подмножество на TRC, наречено "безопасно" TRC. Искаме да дефинираме клас от формули, който отразява това, което е в реалните езици, и е еквивалентен на PA.

Дефиниция 3.8 (Безопасни TRC формули). Тъй като размерността на кортежните променливи не винаги е ясна от контекста в TRC формулите, ще считаме, че размерността на всяка променлива е дадена и не се променя за различните ѝ срещания, дори да са под различни квантори.

1. Няма $\forall$ квантори
2. Когато се използва $\vee$, двете свързани формули $F_1 \vee F_2$ имат само една свободна кортежна променлива и тя е една и съща
3. Да разгледаме произволна максимална конюнкция от една или повече формули:
$$F_1 \wedge \cdots \wedge F_m.$$
Тогава всички компоненти на кортежните променливи, които се срещат свободно в някое от тези F_i са ограничени в следния смисъл:

[18] Забележете, че може, например, в една подформула да е отбелязан $\mu[2]$, без да е отбелязан $\mu[1]$.

a) Ако F_i не е отречена, не е аритметично сравнение и има свободна кортежна променлива μ, то всички компоненти на μ са ограничени

b) Ако F_i е $\mu[j] = a$ или $a = \mu[j]$, където a е константа, то $\mu[j]$ е ограничена

c) Ако F_i е $\mu[j] = \nu[k]$ или $\nu[k] = \mu[j]$ и $\nu[k]$ е ограничена променлива, то и $\mu[j]$ е ограничена.

4. Операцията $\neg$ може да се появява само в член на конюнкция от вида, описан в правило (3).

3.2.3. От PA към безопасно TRC

Лема 3.3. *Всяка заявка, изразима в PA, е изразима в безопасно TRC.*

Доказателство. Конструктивно доказателство с индукция по броя на операциите в израз от PA... □

3.2.4. От TRC към DRC

Преминаването от безопасно TRC към безопасно DRC е директно: заменяме всяка кортежна променлива с колекция от доменни променливи – по една за всеки компонент на кортежната променлива.

Лема 3.4. *За всяка безопасна TRC формула има безопасна формула от DRC, която дефинира същата релация.*

Доказателство. Идеята, както отбелязахме, е да заместим всяка кортежна променлива $\mu^{(k)}$ с k доменни променливи $X_1, \ldots, X_k$ и нека X_j да бъде използвано, точно където е използвано $\mu[j]$. Квантификацията $\exists$, да кажем

$$(\exists\, \mu^{(k)})\, F$$

се замества от:

$$(\exists\, X_1), \ldots, (\exists\, X_k)\, F',$$

където F' е преведената към DRC F. □

3.3. Обобщение

Да обобщим:

Теорема 3.5. *Следните 4 езика дефинират един и същ клас функции:*

1. *Изразите на PA.*
2. *Безопасни нерекурсивни дейталог програми с отрицание.*
3. *Безопасни DRC формули.*
4. *Безопасни TRC формули.*

4. Нормални форми

4.1. Недостатъци на схемата на базата данни

Да разгледаме следния пример: да обединим релациите
SUPPLIERS(SNAME, SADDR) и
SUPPLIES(INAME, SNAME, PRICE) в една обща релация със следната схема:

SUP_INFO(SNAME, SADDR, INAME, PRICE),

която включва цялата информация за доставчиците. В тази схема могат да се забележат няколко проблема:

1. *Излишество.* Адресът на доставчика се повтаря за всеки доставян артикул (item).
2. *Потенциална несъгласуваност (аномалии при обновяване).* Като следствие от излишеството – може да обновим адреса на доставчик в един кортеж, докато в същото време го оставим непроменен в друг кортеж. Така няма да имаме единствен адрес за всеки доставчик, както интуитивно смятаме, че трябва да бъде.
3. *Аномалии при добавяне.* Не можем да запишем адрес за доставчик, ако този доставчик текущо не предлага поне един артикул. Можем да поставим null стойности в колоните ITEM и PRICE кортежа за тази релация, но тогава при въвеждане на артикул за този доставчик дали ще се сетим да изтрием кортежа с null стойностите? Още по-лошо, ITEM и SNAME формират ключ за релацията и може да е невъзможно да извършваме търсене чрез първичен ключ, ако има null стойности в ключовото поле ITEM.
4. *Аномалии при изтриване.* Съществува и проблем, противоположен на (3) – когато изтриваме всички артикули, доставяни от един доставчик, ние неволно изгубваме информацията за адреса на доставчика.

В горния пример всички изброени проблеми изчезват, ако заменим
SUP_INFO с двете релационни схеми:
SUPPLIERS(SNAME, SADDR)
SUPPLIES(SNAME, INAME, PRICE)

Все пак тази декомпозиция си има и свои недостатъци. Например, за да намерим адресите на доставчиците на Brie, трябва да направим съединение, което е скъпа операция.

Процесът на получаване на напълно нормализиран модел данни включва премахване на излишеството чрез разделяне на релациите по такъв начин, че

резултантните релации да могат да бъдат рекомбинирани без загуба на информация – това е принципът на декомпозицията с беззагубно съединение. Също може да искаме да се запазят определени зависимости между атрибутите на резултантните релации. Тези въпроси ще ги засегнем по-нататък.

4.2. Функционални зависимости

4.2.1. Дефиниции

Релациите могат да се използват за моделиране на "реалния свят" по няколко начина; например – всеки кортеж от релация може да представя същност с нейните атрибути или връзка между същности. В много случаи известните факти за реалния свят определят, че не всяко крайно множество от кортежи може да е текуща стойност на дадена релация, дори да имат правилната размерност и стойности от правилните домени. Можем да различим два вида ограничения върху релациите:

1. *Ограничения, определени от семантиката на елементите на домена.* Тези ограничения зависят от разбирането за това какъв е смисъла на компонентите на релацията. Например, никой не е 20 метра висок. Полезно е СУБД-то да може да проверява за такива неправдоподобни стойности, които вероятно са възникнали при въвеждане или изчисляване на данни. Това са т.нар. "ограничения по цялостност" и не ни казват почти нищо за дизайна на схемите на базата данни, затова няма да ги разглеждаме сега.
2. *Ограничения върху релациите, зависещи само от равенство или неравенство на стойности.* Тези ограничения не зависят от това каква стойност има даден кортеж в даден компонент, а само от това дали два кортежа са съгласуване по определени компоненти. Първо ще разгледаме най-важните от тези ограничения, наречени *функционални зависимости*.

Дефиниция 4.1 (Функционални зависимости). Нека $R(A_1, \ldots, A_n)$ е релационна схема и X и Y са подмножества на $\{A_1, \ldots, A_n\}$. Казваме, $X \to Y$ и четем *"X функционално определя Y"*, или *"Y функционално зависи от X"*, ако за всяка релация r, която е текуща стойност на R е невъзможно r да има два кортежа, които да съвпадат по всички атрибути на X и да се отличават в един или повече атрибути на Y; т.е. в сила е следното:

$$(\forall r \text{ за } R)(\forall \mu \in r)(\forall \nu \in r)(\mu[X] = \nu[X] \implies \mu[Y] = \nu[Y]).$$

Докато не стигнем до разглежданията за многозначни зависимости в секция 4.6, под зависимости ще разбираме точно функционални зависимости.

4.2.2. Конвенции за означенията

Нататък ще използваме следните означения:

1. Главните букви в началото на азбуката означават единични атрибути.
2. Главните букви в края на азбуката $U, V, \cdots, Z$ означават множество от атрибути, възможно и синглетони (единични множества).
3. R се използва за означаване на релационна схема. Също ще именуваме релации чрез техните схеми, например релация с атрибути A, B и C може да наричаме ABC.
4. Ще използваме r за релация – текущ екземпляр на схемата R.
5. Конкатенацията се използва като обединение. Така $A_1 \ldots A_n$ означава множеството $\{A_1, \ldots, A_n\}$ и XY е $X \cup Y$. Също XA или AX, където X е множество от атрибути и A е единичен атрибут, означава $X \cup \{A\}$.

4.2.3. Логически изводи от функционални зависимости

Да предположим, че R е релационна схема и A, B и C са някои от атрибутите ѝ. Нека също функционалните зависимости $A \rightarrow B$ и $B \rightarrow C$ са в сила за R. Тогава твърдим, че $A \rightarrow C$ също е в сила за R. Това лесно се доказва с допускане на противното.

Дефиниция 4.2 (Логически извод от множество функционални зависимости). Нека F е множество от функционални зависимости за релационната схема R и нека $X \rightarrow Y$ е функционална зависимост. Казваме, че F *логически извежда* $X \rightarrow Y$ и пишем $F \models X \rightarrow Y$, ако всяка релация r за R, която удовлетворява зависимостите в F, също удовлетворява и $X \rightarrow Y$.

Дефиниция 4.3 (Затваряне на множество от зависимости). Дефинираме F^+, *затварянето* на F, като множеството от функционални зависимости, които логически следват от F; т.е.:

$$F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \}.$$

4.2.4. Ключове

Когато говорим за множества същности приемаме, че ключът е множество от атрибути, които уникално идентифицират една същност. Има аналогична концепция за релации с функционални зависимости.

Дефиниция 4.4 (Ключ). Ако R е релационна схема с атрибути $A_1 \ldots A_n$ и функционални зависимости F и $X \subseteq A_1 \ldots A_n$, казваме, че X е *ключ* на R, ако:

1. $X \to A_1 \dots A_n$ е в F^+.
2. За никое $Y \neq \emptyset,\ Y \subset X$ не е вярно, че $Y \to A_1 \dots A_n$ е в F^+.

Тъй като може да има повече от един ключ за една релация, понякога означаваме един като *"първичен ключ"*. Понякога терминът *"кандидат-ключ"* или *"възможен ключ"* се използва за означаване на произволен ключ според горната дефиниция, а терминът "ключ" се запазва за един набелязан ("първичен") кандидат ключ [19]. Също ще използваме термина *суперключ* за произволно супермножество на някой ключ.

4.2.5. Аксиоми на Армстронг за функционални зависимости

Сега ще представим набор от аксиоми, които ни позволяват от една или повече функционални зависимости да извеждаме други.

Да предположим, че са дадени релационна схема с множество от атрибути U [20] и множество от функционални зависимости F, използващи само атрибути от U. Правилата за извод са:

(A1). *Рефлексивност.* Ако $Y \subseteq X \subseteq U$, то $X \to Y$ логически се извежда от F. Това правило ни дава *тривиалните зависимости* – те зависят само от U, но не от F.

(A2). *Нарастване (Augmentation).* Ако $X \to Y$ е в сила и $Z \subseteq U$ е произволно, то $XZ \to YZ$ също е в сила. Важно е да се отбележи, че $X \to Y$ или е в F, или е изведено от F по аксиомите.

(A3). *Транзитивност.* Ако $X \to Y$ и $Y \to Z$ са в сила, то $X \to Z$ също е в сила.

4.2.6. Надеждност *(soundness)* на аксиомите на Армстронг

Лема 4.1. *Аксиомите на Армстронг са* надеждни*, т.е., ако $X \to Y$ е изведена от F чрез аксиомите, то $X \to Y$ е вярно за всяка релация, в която важат зависимостите F.*

Доказателство.
- A1. очевидно е коректна: не може да има релация r с два различни кортежа, които да съвпадат по X и да не съвпадат по някое негово подмножество.
- A2. Да допуснем, че релацията r удовлетворява $X \to Y$ и че μ и ν са два кортежа, т.ч. $\mu[XZ] = \nu[XZ]$, но $\mu[YZ] \neq \nu[YZ]$. Тъй като е невъзможно μ и ν да се различават в някой атрибут от Z, то те се

[19] Ние ще се придържаме към дефиницията (4.4) за "ключ", която дадохме.
[20] Универсалното множество от атрибути.

различават в някой атрибут от Y, което е противоречие с факта, че $X \to Y$ е в сила за r.

– A3. Нека r е релация в която са в сила $X \to Y$ и $Y \to Z$ и μ и ν са два произволни кортежа, такива че $\mu[X] = \nu[X]$. Тогава:

$$\begin{aligned} \mu[Y] &= \nu[Y] && (\text{защото } \mu[X] = \nu[X] \text{ и } X \to Y) \\ \mu[Z] &= \nu[Z] && (\text{защото } \mu[Y] = \nu[Y] \text{ и } Y \to Z) \end{aligned}$$

Тъй като μ и ν са произволни, то $X \to Z$ е в сила за r.

□

4.2.7. Допълнителни правила за извод

Лема 4.2. *В сила са следните полезни правила за извод, които следват от аксиомите:*

a) Обединение. $\{X \to Y, X \to Z\} \models X \to YZ$.

b) Псевдотранзитивност. $\{X \to Y, WY \to Z\} \models WX \to Z$.

c) Декомпозиция. *Ако* $X \to Y$ *е в сила и* $Z \subseteq Y$*, то* $X \to Z$ *е в сила.*

Доказателство. a) Обединение.

$$\left.\begin{array}{ll} X \to Y \models X \to XY & (\text{по A2}) \\ X \to Z \models XY \to ZY & (\text{по A2}) \end{array}\right\} \models X \to ZY \quad (\text{по A3})$$

b) Псевдотранзитивност.

$$\left.\begin{array}{rll} (\text{по A2}) & X \to Y \models & WX \to WY \\ (\text{по условие}) & & WY \to Z \end{array}\right\} \models WX \to Z \quad (\text{по A3})$$

c) Декомпозиция.

$$\left.\begin{array}{rll} (\text{по A1}) & Z \subseteq Y \implies & Y \to Z \\ (\text{по условие}) & & X \to Y \end{array}\right\} \models X \to Z \quad (\text{по A3})$$

□

От правилата *обединение* и *декомпозиция* получаваме следното важно следствие:

Следствие 4.3. *Ако* $A_1, \ldots, A_n$ *са атрибути, то* $X \to A_1 \ldots A_n$ *е в сила* $\iff$ $X \to A_i$ *е в сила за всяко* $i = 1, \ldots, n$.

4.2.8. Пълнота на аксиомите на Армстронг

Дефиниция 4.5 (Затваряне на множество от атрибути). Нека F е множество от функционални зависимости върху множеството от атрибути U и нека $X \subseteq U$. Тогава X^+ – *затварянето* на X (спрямо F), дефинираме като множество от атрибути по следния начин:

$$X^+ = \{\, A \mid X \to A \text{ може да се изведе от аксиомите на Армстронг} \,\}.$$

Лема 4.4. *Нека F е дадено множество от функционални зависимости. Тогава $X \to Y$ следва от аксиомите на Армстронг $\iff Y \subseteq X^+$, където X е взето спрямо F.*

Доказателство. Твърдението следва от правилата за обединение и декомпозиция. □

Теорема 4.5 (Пълнота на аксиомите на Армстронг). *Аксиомите на Армстронг са надеждни и пълни.* [21]

Следствие 4.6 (Еквивалентна дефиниция за X^+).

$$X^+ = \{\, A \mid F \models X \to A \,\}$$

Следствие 4.7 (Еквивалентна дефиниция на F^+). *Дефинирахме F^+ като множеството от зависимости, които логически следват от F – вече може да гледаме на F като множеството от зависимости, които следват от F по аксиомите на Армстронг.*

4.2.9. Изчисляване на затваряне

Оказва се, че изчисляването на F^+ е времеотнемаща задача по принцип, просто защото множеството от зависимости в F^+ може да е много голямо, дори F да е малко.

Нека $F = \{A \to B_1, \ldots, A \to B_n\}$. Тогава F включва всички зависимости $A \to Y$, където $Y \subseteq \{B_1, \ldots, B_n\}$ – такива има 2^n на брой!

От друга страна, обаче, пресмятането на X^+ не е трудно; отнема време пропорционално на дължината на всички зависимости в F. От Лема 4.4 и от факта, че аксиомите на Армстронг са надеждни и пълни, можем да определим дали $X \to Y$ е в F като изчислим X^+ спрямо F и проверим дали Y е вътре. Лесен начин да намерим X^+ е следният:

Алгоритъм 4.1. Изчисляване на затварянето на множество атрибути спрямо множество от функционални зависимости.

[21] Надеждността вече я доказахме. Доказателството на пълнотата е малко по-обемно и няма да го правим.

ВХОД: Крайно множество от атрибути U, множество от функционални зависимости F върху U и множество $X \subseteq U$.

ИЗХОД: X^+ – затварянето на X спрямо F.

МЕТОД: Изчисляваме множества от атрибути $X^{(0)}, X^{(1)}, \ldots$ по следните правила:

1. $X^{(0)} = X$.
2. $X^{(i+1)} = X^{(i)} \cup \{A \,|\, (\exists Y \to Z \in F)\, A \in Z \,\&\, Y \subseteq X^{(i)}\}$

Тъй като $X = X^{(0)} \subseteq \cdots \subseteq X^{(i)} \subseteq \cdots \subseteq U$ и U е крайно, в един момент стигаме такова j, че: $X^{(j)} = X^{(j+1)} = \cdots$. Няма смисъл да пресмятаме повече след като разберем, че $X^{(j)} = X^{(j+1)}$. Може да се докаже, че $X^+ = X^{(j)}$. □

Пример 4.1. За множеството от зависимости:

$$\begin{array}{cccc} AB \to C & C \to A & BC \to D & ACD \to B \\ D \to EG & BE \to C & CG \to BD & CE \to AG \end{array}$$

по този алгоритъм получаваме, че $(BD)^+ = ABCDEG$. □

Теорема 4.8. *Алгоритъм 4.1 коректно изчислява* X^+.

4.2.10. Минимални покрития

Дефиниция 4.6. Нека F и G са множества от зависимости. Казваме, че F и G са еквивалентни, ако $F^+ = G^+$.

Лема 4.9. *F и G са еквивалентни $\iff$ всяка зависимост във F е в G^+ и всяка зависимост в G е във F^+.*

Лема 4.10. *Всяко множество от функционални зависимости F е еквивалентно на множество зависимости G, в което никоя дясна страна няма повече от един атрибут.*

Доказателство. Нека $G = \{X \to A \,|\, (\exists\, X \to Y \in F)\, A \in Y\}$. Лесно се проверява, че F и G са еквивалентни. □

Дефиниция 4.7 (Минимално множество от зависимости). Казваме, че множеството от зависимости F е *минимално*, ако:

1. Всяка дясна страна на зависимост е единичен атрибут.
2. За никое $X \to A \in F$ няма множество $F - \{X \to A\}$, еквивалентно на F.
3. За никое $X \to A \in F$ и $Z \subset X, Z \neq \emptyset$, не е вярно, че $(F - \{X \to A\}) \cup \{Z \to A\}$ е еквивалентно на F.

Интуитивно (2) гарантира, че никоя зависимост в F не е излишна. Лесно може да се провери дали $X \to A$ е излишно чрез изчисляване на X^+ спрямо $F - \{X \to A\}$. Ако $A \in X^+$, то зависимостта е излишна.

Условието (3) гарантира, че няма атрибут от никоя лява страна, който да е излишен. Следният тест проверява излишните атрибути отляво:

Лема 4.11. *Атрибутът $B \in X$ е излишен за зависимостта $X \to A \iff A \in (X - \{B\})^+$, където затварянето е взето спрямо F.*

Дефиниция 4.8 (Минимално покритие за множество от зависимости). Ако G е множество от зависимости, което е минимално в горния смисъл и G е еквивалентно на F, то казваме, че G е *минимално покритие* за F.

Теорема 4.12. *Всяко множество от зависимости F има минимално покритие.*

Доказателство. По Лема 4.10 можем да считаме, че всяка дясна страна има само един атрибут. Многократно търсим нарушения на (2) и (3) и модифицираме множеството от атрибути по съответен начин. При всяка модификация се изтрива зависимост или атрибут – това не може да продължава безкрайно и в един момент се стига до множество от зависимости, което удовлетворява (1), (2) и (3).

За условие (2) разглеждаме всяка зависимост $X \to Y$ в текущото F и ако $F - \{X \to Y\}$ е еквивалентно на F, [22] изтриваме $X \to Y$ от F.

За условие (3) разглеждаме всяка зависимост $A_1 \dots A_n \to B$ в текущото F. Премахваме всяко A_i отляво, за което:

$$(F - \{A_1 \dots A_n \to B\}) \cup \{A_1 \dots A_{i-1} A_{i+1} \dots A_n \to B\}$$

е еквивалентно на F.[23]

Може да се докаже, че е достатъчно първо да се елиминират всички нарушения на (3) и след това всички на (2), но не и обратното.

Трябва да отбележим, че редът, по който елиминираме нарушенията има значение за крайния резултат – т.е. може да получим различни минимални покрития. □

4.3. Декомпозиции на релационни схеми

Дефиниция 4.9 (Декомпозиция на релационна схема). Наричаме *декомпозиция* на релационната схема $R = \{A_1, \dots, A_n\}$ нейната замяна с колек-

[22] Това е така, ако $\{X \to Y\} \in (F - \{X \to Y\})^+$, което е еквивалентно на $Y \in X^+$ спрямо $F - \{X \to Y\}$.

[23] Това е така, ако $B \in (A_1 \dots A_n)^+$ спрямо $(F - \{A_1 \dots A_n \to B\}) \cup \{A_1 \dots A_{i-1} A_{i+1} \dots A_n \to B\}$.

ция $\rho = \{R_1, \ldots, R_k\}$ от подмножества на R, такива че:

$$R = R_1 \cup R_2 \cup \ldots \cup R_k \,.$$

Не се изисква R_i да са взаимно чужди. Една от мотивациите за декомпозицията е, че може да елиминира някои проблеми, отбелязани в секция 4.1.

4.3.1. Декомпозиции, запазващи съединението

Дефиниция 4.10 (Декомпозиция с беззагубно съединение). Ако R е релационна схема, декомпозирана на $R_1, \ldots, R_k$ и D е множество от зависимости, казваме че декомпозицията има беззагубно съединение (спрямо D), ако за всяка релация r за R, която удовлетворява D е в сила:

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_k}(r)$$

Беззагубното съединение е необходимо свойство, за да може декомпозираната релация да бъде възстановена.

Способността да съхранява "висящи кортежи"[24] е предимство декомпозицията. Това предимство трябва да се балансира с необходимостта често да се отговаря на заявки, изискващи съединение. Когато всичко е взето в предвид, обикновено се смята, че декомпозицията е полезна, когато трябва да се решават проблеми като излишеството, описани в секция 4.1, но не и в противен случай.

Проверка за беззагубни съединения

Алгоритъм 4.2. Проверка за беззагубно съединение.

ВХОД: Релационна схема $R = A_1 \ldots A_n$, множество от функционални зависимости F и декомпозиция $\rho = (R_1, \ldots, R_n)$

ИЗХОД: Решение дали ρ е с беззагубно съединение.

МЕТОД: Конструираме таблица с n колони и k реда; колона j съответства на атрибут A_j и ред i съответства на релационната схема R_i. В ред i и колона j поставяме a_j, ако A_j е в R_i, ако не – поставяме b_{ij} там.

Многократно "разглеждаме" всяка от зависимостите $X \to Y \in F$, докато таблицата се променя. Всеки път като разглеждаме $X \to Y$, търсим редове, които съвпадат във всички колони за атрибутите на X. Ако намерим два такива реда, уеднаквяваме символите на тези редове за атрибутите на Y. Когато уеднаквяваме два символа, ако единият от тях е a_j – правим и другия a_j. Ако двата са b_{ij} и b_{lj} няма значение кой

[24] Кортежи, за които липсват съответните им кортежи от друга релация се нар. висящи.

ще изберем. Важно е да се разбере, че когато уеднаквяваме два символа, всички срещания в таблицата стават същите; не е достатъчно да ги уеднаквим само в срещането, където се нарушава зависимостта $X \to Y$.

След модифициране на редовете на таблицата по описания начин, проверяваме дали някои ред е станал $a_1 \ldots a_n$. Ако да – тогава съединението е беззагубно, иначе – не е беззагубно.

□

Пример 4.2. Нека $R = ABCD$, $\rho = \{AD, AB, BE, CDE, AE\}$ и $F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$. Оказва се, че ρ е беззагубно спрямо F.

Също така $\{SA, SIP\}$ е декомпозиция на релацията $SAIP$ с беззагубно съединение спрямо $\{S \to A,\ SI \to P\}$. □

Теорема 4.13. *Алгоритъм 4.2 коректно определя дали дадена декомпозиция има беззагубно съединение.*

Алгоритъм 4.2 може да се приложи за произволен брой релационни схеми, но в случая на декомпозиция на две схеми можем да дадем и по-прост тест.

Теорема 4.14. *Ако $\rho = (R_1, R_2)$ е декомпозиция на R и F е множество от функционални зависимости, тогава ρ има беззагубно съединение спрямо $F \iff$*

$$F \models (R_1 \cap R_2) \to (R_1 - R_2) \text{ или } F \models (R_1 \cap R_2) \to (R_2 - R_1).$$

Тези зависимости не е задължително да са в F, достатъчно е да са в F^+.

4.3.2. Декомпозиции, запазващи функционалните зависимости

Видяхме, че е желателно за една декомпозиция да има беззагубно съединение, защото това гарантира, че произволна релация може да бъде възстановена от нейните проекции. Друго важно свойство на декомпозицията $\rho = (R_1, \ldots, R_n)$ на една релационна схема R върху $\rho = (R_1, \ldots, R_n)$ е способността множеството от зависимости F да се извежда от проекциите на F върху R_i-тата.

Дефиниция 4.11 (Проекция на множество от функционални зависимости). *Проекцията* на множество от функционални зависимости F върху множество от атрибути Z е:

$$\pi_Z(F) = \left\{ X \to Y \mid X \to Y \in F^+ \,\&\, XY \subseteq Z \right\}.^{25}$$

[25] Забележете, че може $X \to Y \notin F$.

Дефиниция 4.12 (Декомпозиция, запазваща функционалните зависимости). Казваме, че декомпозицията $\rho = (R_1, \ldots, R_n)$ запазва множеството от зависимости F, ако $\bigcup_{i=1}^{n} \pi_{R_i}(F)$ логически извежда всички зависимости в F. С други думи:

$$F \subseteq \left(\bigcup_{i=1}^{n} \pi_{R_i}(F) \right)^{+}.$$

Причината да желаем ρ да запазва F, е зависимостите в F да бъдат разглеждани като ограничения по цялостност за релацията R. Ако проектираните зависимости не извеждат F, тогава ако представяме R чрез $\rho = (R_1, \ldots, R_n)$, може да се окаже, че текущите стойности на R_i представят релация за R, която не удовлетворява F, дори ρ да има беззагубно съединение спрямо F. Алтернативно – всяка промяна на едно от R_i ще изисква съединение, за да се провери дали зависимостите не са нарушени.

Трябва да отбележим, че декомпозицията може да има беззагубно съединение спрямо F, но да не запазва F. Например:

Пример 4.3. $\rho = (SZ, CS)$ при $R = CSZ$ и $F = \{CS \to Z, Z \to C\}$. [26] □

Също е възможно декомпозицията да запазва F, но да няма беззагубно съединение.

Пример 4.4. Например това е така за $F = \{A \to B,\ C \to D\}$, $R = ABCD$ и $\rho = (AB, CD)$. □

Проверка за запазване на зависимостите По принцип изглежда лесно да се провери дали декомпозицията $\rho = (R_1, \ldots, R_k)$ запазва F. Просто изчисляваме F^+ и го проектираме върху всяко R_i. След това вземаме обединението от резултантните множества зависимости и проверяваме дали е еквивалентно на F.

Обаче пресмятането на F^+, както споменахме по-рано е тежка задача – често броят зависимости в F^+ зависи експоненциално от размера на F. Добрата новина е, че има начин да направим необходимия ни тест без действително да пресмятаме F^+; този метод отнема полиномиално време спрямо размера на F.

Алгоритъм 4.3. Проверка за запазване на зависимости.

ВХОД: Декомпозиция $\rho = (R_1, \ldots, R_k)$ и множество от функционални зависимости F.

[26] Тук сме означили с C – CITY, S – STATE, Z – ZIP. Смисълът на дадените зависимости е, че ZIP кодът еднозначно определя улицата, а градът и улицата еднозначно определят ZIP кода. (за ZIP кода може да си мислите като за пощенски код).

ИЗХОД: Решение дали ρ запазва F.

МЕТОД: Дефинираме $G = \cup_{i=1}^{n} \pi_{R_i}(F)$. Забележете, че не изчисляваме G – просто искаме да разберем дали е еквивалентно на F. За да определим дали G е еквивалентно на F, трябва да разгледаме всяко $X \rightarrow Y \in F$ като определим дали $Y \subseteq X^+$, където затварянето е спрямо G. Тази проверка става по следния начин:

```
Z := X;
while( има промени в Z )
{
     for( i := 1 to k )
     {
          Z := Z ∪ ((Z ∩ R_i)^+ ∪ R_i);
          /* затварянето е спрямо F */
     }
}
```

Ако $Y \subseteq Z$, то $X \rightarrow Y \in G^+$. Ако всяко $X \rightarrow Y$, което е във F е също и в G^+ – отговорът е "да", иначе – "не".

□

Теорема 4.15. *Алгоритъм 4.3 коректно определя дали* $X \rightarrow Y \in G^+$.

4.4. Нормални форми за релационни схеми

Дефинирани са различни свойства, или "нормални форми" за релационни схеми с дефинирани зависимости. Най-значителни от тези са "трета нормална форма" и "нормална форма на Бойс-Код". Целта е да се избегнат проблемите с излишеството и аномалиите, дискутирани в секция 4.1.

4.4.1. Дефиниции

Всяка следваща от дефинициите за нормални форми, в реда, в който които ще ги дадем, поставя все по-силни изисквания; например всяка релационна схема, която е в 3NF е също и във 2NF.

Дефиниция 4.13 (Основни атрибути). Наричаме атрибута A в релационната схема R, *основен атрибут*, ако A е член на някой ключ в R.

Дефиниция 4.14 (Първа Нормална Форма, 1NF, First Normal Form). Една релационна схема R е в *първа нормална форма*, ако всичките ѝ домени се състоят от неделими стойности.

Ако една релационна схема не е в 1NF, се казва че е *ненормализирана*. Подобна схема има два проблема:

1. Семантиката ѝ се определя трудно. Като погледнем в общия случай една релационна схема $R(A_1, \ldots, A_n)$ няма начин да разберем дали един атрибут е съставен или не.
2. Операциите в релационния модел работят само върху домени с прости стойности.

Въпросът доколко един домен се състои от неделими стойности зависи от конкретната предметна област, която се моделира и начина на използване на базата данни. В един случай, например, може да е удобно да пакетираме стойности за дати, като неделими, а в друг – да ги разделим на година, месец и т.н.

За да постигнем 1NF – когато открием съставен атрибут го махаме и го поставяме в нова същност. Също съхраняваме и ключ от оригиналната същност в новата, за да могат двете да бъдат свързани.

Дефиниция 4.15 (Втора Нормална Форма, 2NF). [27] Една релационна схема R е във *втора нормална форма*, ако за всяка зависимост $X \to A$, която е в сила за R и $A \notin X$ имаме, че A е основен атрибут или X не е собствено подмножество на никой ключ.

Дефиниция 4.16 (Трета Нормална Форма, 3NF, Third Normal Form). Една релационна схема R е в *трета нормална форма*, ако винаги когато $X \to A$ е в сила за R и $A \notin X$, тогава X е суперключ за R или A е основен.

Дефиниция 4.17 (BCNF - Boyce-Codd normal form). Една релационна схема R със зависимости F е в *нормална форма на Бойс-Код (BCNF)*, ако винаги когато $X \to A$ е в сила за R и $A \notin X$, тогава X е суперключ. С други думи, единствените нетривиални зависимости са тези, в които ключ определя един или повече други атрибути. Единствената разлика от $3NF$, е че тук липсва клаузата "е основен", което прави $BCNF$ по-силна.

Трябва да търсим нарушения за зависимостите $X \to A$ не само във F, а в цялото F^+. Обаче, може да се докаже, че ако всички зависимости в F са с единични атрибути вдясно, тогава ако няма нарушения в F, то няма нарушения и в F^+. Това важи и за 3NF и за BCNF.

Нормалните форми не винаги са полезно нещо. Те обикновено се използват само като последна мярка за преодоляване на аномалии от типа на тези, които посочихме.

[27] 2NF представлява само исторически интерес и няма да я разглеждаме по-нататък.

4.5. Декомпозиция в BCNF и 3NF

Оказва се, че всяка релационна схема има декомпозиция с беззагубно съединение в BCNF и има декомпозиция в 3NF, която има беззагубно съединение като запазва също и зависимостите. Обаче може и да няма декомпозиция на релационна схема в BCNF, която да запазва зависимостите.

Пример 4.5. Релационната схема CSZ от Пример 4.3 не е в BCNF, защото е в сила $Z \to C$. Ако декомпозираме CSZ по какъвто и да е начин, при който CSZ не е в никоя от схемите на декомпозицията, тогава зависимостта $CS \to Z$ не може да се изведе от проектираните зависимости. □

4.5.1. Декомпозиция в BCNF с беззагубно съединение

Лема 4.16. *Нека R е релационна схема с функционални зависимости F. Нека $\rho = (R_1, \ldots, R_n)$ е декомпозиция на R с беззагубно съединение спрямо F. Нека $\sigma = (S_1, S_2)$ е декомпозиция с беззагубно съединение на R_1 спрямо $\pi_{R_1}(F)$. Тогава декомпозицията на R в $(S_1, S_2, R_2, \ldots, , R_n)$ също има беззагубно съединение спрямо F.* □

Можем да приложим Лема 4.16, за да получим прост но време-отнемащ алгоритъм за декомпозиция на релационната схема в BCNF. Ако намерим нарушение на BCNF, да кажем $X \to A$, декомпозираме R на схемите $R - A$ и XA. И двете са по-малки от R.[28] Съединението на $R - A$ и XA е беззагубно по Теорема 4.14, защото:

$$\left.\begin{aligned} (R-A) \cap XA &= X \\ X \to A \models X &\to XA \\ \models X &\to XA - (R-A) \\ F \models X &\to A \end{aligned}\right\} F \models XA \cap (R-A) \to XA - (R-A).$$

Изчисляваме проекциите на зависимостите за R върху $R - A$ и XA и прилагаме тази декомпозиция рекурсивно върху двете схеми. Това продължава докато не стигнем само до схеми в BCNF. Лема 4.16 ни гарантира беззагубното съединение.

Проблемът е, че проекцията на зависимостите може да отнеме експоненциално време спрямо размера на R и множеството зависимости. Оказва се, че можем да намерим някои беззагубни декомпозиции за полиномиално време. Тази техника, обаче, понякога ще декомпозира схеми, които вече са в $BCNF$.

Лема 4.17. *В сила са следните твърдения:*

[28] Ако $XA = R$ се получава, че X е ключ, което е противоречие с нарушението на BCNF.

a) Всяка 2-атрибутна схема е в BCNF.

b) Ако R не е в $BCNF$, тогава можем да намерим атрибути A и B в R, за които $(R - AB) \to A$ е в сила. (Зависимостта $(R - AB) \to B$ може да е в сила, но може и да не е). [29]

Според Лема 4.17(b) е достатъчно да разгледаме всяка двойка $\{A, B\}$, такива има $n(n-1)/2$ на брой, и да изчислим $(R - AB)^+$ спрямо F. Ако за никое A и B $(R - AB)^+$ не съдържа нито A, нито B, тогава от лемата следва, че R е в BCNF.

Лема 4.18. *Нека имаме множество от зависимости F върху R и ги проектираме върху $R_1 \subseteq R$, за да получим F_1. След това проектираме F_1 върху $R_2 \subseteq R_1$, за да получим F_2. Тогава $F_2 = \pi_{R_2}(F)$. С други думи:*

$$\pi_{R_2}(\pi_{R_1}(F)) = \pi_{R_2}(F) .$$

Доказателство. $XY \subseteq R_2 \implies (X \to Y \in F^+ \iff X \to Y \in F_1^+)$. □

Лема 4.18 има важно следствие. Тя казва, че ако декомпозираме схемата, както в Лема 4.16, тогава всъщност никога не е нужно да изчисляваме проектираните зависимости, когато декомпозираме. Достатъчно е да работим с дадените зависимости и да изчисляваме затворенията на множествата атрибути по Алгоритъм 4.1, когато се наложи.

С това наблюдение и Лема 4.17(b) стигаме до следния алгоритъм, който е с полиномиална сложност спрямо размера на релационната схема R и зависимостите F.

Алгоритъм 4.4. Декомпозиция с беззагубно съединение в BCNF.

ВХОД: Релационна схема R и функционални зависимости F.

ИЗХОД: Декомпозиция на R с беззагубно съединение, т.ч. всяка релационна схема в декомпозицията е в BCNF спрямо проекцията на F върху тази схема.

МЕТОД: а) Главна програма

```
Z := R;
repeat
    декомпозираме Z на Z-A и XA, където XA е в BCNF
       и X->A; /* използваме процедурата от (b) */
    добавяме XA към декомпозицията;
    Z := Z - A;
until Z не може да бъде декомпозирано според Лема 4.17(b);
добавяме Z към декомпозицията;
```

[29] Обратното твърдение в общия случай не е вярно.

b) Декомпозираща процедура

```
if Z не съдържа нито A, нито B,
           т.ч. A е в (Z − BA)+, then
    /* всички затваряния се вземат спрямо  F */
    return "Z е в BCNF и не може
                  да бъде декомпозирано";
else begin
    намираме едни такива A и B;
    Y := Z - B;
    /* използваме същите променливи за A и B,
       т.е. новите стойности заместват старите */
    while Y съдържа A и B, т.ч.(Y − AB)+ → A [30] do
        Y := Y - B;
    return декомпозицията Z-A и Y;
    /* Y тук е XA в главната програма */
end
```

□

Проблеми с произволните BCNF декомпозиции

Може да се получи разцепване на релации, които е естествено да бъдат цели, т.е. за тях често ще се налага да се прави съединение. Още повече, че горният алгоритъм понякога разцепва релации, които вече са в BCNF.

Друг проблем е, че някои декомпозиции не запазват функционалните зависимости.

4.5.2. Декомпозиции в 3NF, които запазват зависимостите

Алгоритъм 4.5. Декомпозиция в 3NF, която запазва зависимостите.

ВХОД: Релационна схема R и функционални зависимости F, които без ограничение на общността считаме, че са минимално покритие (заради Теорема 4.12).

ИЗХОД: Декомпозиция, която запазва зависимостите на R, т.ч. всяка релационна схема е в 3NF спрямо проекцията на F върху тази схема.

МЕТОД: Ако има атрибути, които не са включени в никоя зависимост на F, нито отдясно, нито отляво, тогава такъв атрибут по принцип може да

[30] Това условие е еквивалентно на $A \in (Y - AB)^+$. Т.е. цикли се докато Y не стане в BCNF.

формира релационна схема чрез себе си и ще го елиминираме от R.[31] Ако някоя от зависимостите в R включва всички атрибути на R, извеждаме самото R. Иначе резултантната декомпозиция ρ ще се състои от схеми: XA за всяко $X \to A$ в F, т.е.:

$$\rho = \{\, XA \mid \forall\ X \to A\ \in F\}.$$

□

Теорема 4.19. *Алгоритъм 4.5 води до декомпозиция в 3NF, която запазва зависимостите.*

Пример 4.6. Да разгледаме релационната схема CTHRSG, които имат зависимости с минимално покритие F:

$$C \to T \qquad HR \to C \qquad HT \to R \qquad CS \to G \qquad HS \to R$$

Алгоритъм 4.5 води до релационните схеми: CT, CHR, THR, CSG и HRS.
□

Има модификация на Алгоритъм 4.5, която избягва едни ненужни декомпозиции. Ако $X \to A_1, \ldots,\ X \to A_n$ са зависимости в минимално покритие, тогава може да използваме релационната схема $XA_1 \ldots A_n$, вместо n-те схеми $XA_1, \ldots,\ XA_n$. Може да се докаже, че $XA_1 \ldots A_n$ е в 3NF.

Декомпозиция в 3NF с беззагубно съединение и запазваща зависимостите

Теорема 4.20. *Нека σ е декомпозиция в 3NF на R от Алгоритъм 4.5 и нека X е ключ за R. Тогава $\tau = \sigma \cup \{X\}$ е декомпозиция на R с всички релационни схеми в 3NF, запазваща зависимостите и имаща свойството беззагубно съединение.*

Очевидно понякога τ не е минимално множество от релационни схеми със свойствата от Теорема 4.20. Може да изхвърлим релационни схеми от τ, ако всички свойства се запазват след това.[32] Може да се получат много различни схеми на базата данни в зависимост от реда по който изхвърляме схемите, т.к. елиминирането на една може да забрани изхвърлянето на друга схема.

[31] Понякога е желателно да имаме два или повече атрибута, да кажем A и B, които да са заедно в една релационна схема. Може просто да има връзка много към много между тях. Идея: да добавим изкуствен атрибут θ и функционална зависимост $AB \to \theta$, за да форсираме тази асоциация. След приключване на дизайна, атрибутът θ се елиминира.

[32] Например, ако една от схемите е подмножество на друга, то по-малката е излишна.

4.6. Многозначни зависимости

В предните секции допуснахме, че единствено възможния вид зависимости на данните е функционалният. Всъщност има много видове правдоподобни зависимости. Един от тях – многозначната зависимост, често се среща в "реалния свят".

Да предположим, че са дадени релационната схема R и $X \subseteq R$ и $Y \subseteq R$. Интуитивно казваме, че $X \twoheadrightarrow Y$ и четем "X *многозначно определя* Y" или "има *многозначна зависимост* на Y от X", ако за дадени стойности на атрибутите на X има множество от нула или повече асоциирани стойности за атрибутите на Y, и това множество от Y-стойности не е свързано по никакъв начин със стойностите на атрибутите в $R - X - Y$.[33] По-формално:

Дефиниция 4.18 (Многозначна зависимост). Казваме, че $X \twoheadrightarrow Y$ е в сила за R, ако за произволна релация r за R и μ, ν – кортежи в r, т.ч. $\mu[X] = \nu[X]$ е вярно, че r също съдържа кортежи ϕ и ψ, където:

1. $\phi[X] = \psi[X] = \mu[X] = \nu[X]$;
2. $\phi[Y] = \mu[Y]$ и $\phi[R - X - Y] = \nu[R - X - Y]$;
3. $\psi[Y] = \nu[Y]$ и $\psi[R - X - Y] = \mu[R - X - Y]$.[34]

С други думи: можем да разменим Y стойностите на μ и ν и да получим два нови кортежа ϕ и ψ, които също трябва да са в r. Забележете,че не изискваме X и Y да са чужди в горната дефиниция.

4.6.1. Аксиоми за функционални и многозначни зависимости

Сега ще представим надеждно и пълно множество от аксиоми за правене на изводи от множество функционални и многозначни зависимости над множество от атрибути U. Първите три са аксиомите на Армстронг за функционални зависимости, тук просто ги повтаряме.

(A1). *Рефлексивност.* Ако $Y \subseteq X \subseteq U$, то $X \to Y$ логически се извежда от F.

(A2). *Нарастване (Augmentation).* Ако $X \to Y$ е в сила и $Z \subseteq U$ е произволно, то $XZ \to YZ$ също е в сила.

(A3). *Транзитивност.* Ако $X \to Y$ и $Y \to Z$ са в сила, то $X \to Z$ също е в сила.

(A4). *Допълнение за многозначни зависимости.*

$$\{X \twoheadrightarrow Y\} \models X \twoheadrightarrow (U - X - Y).$$

[33] Асоциативността е лява, т.е. $R - X - Y = (R - X) - Y$

[34] Забележете, че може да елиминираме клауза (3). Съществуването на кортеж ψ следва от съществуването на ϕ като приложим дефиницията с разменени μ и ν.

(A5). *Нарастване (Augmentation) за многозначни зависимости.* Ако $X \twoheadrightarrow Y$ е в сила и $V \subseteq W$, то $WX \twoheadrightarrow VY$.

(A6). *Транзитивност за многозначни зависимости.*

$$\{X \twoheadrightarrow Y,\, Y \twoheadrightarrow Z\} \models X \twoheadrightarrow (Z - Y).$$

(A7). $\{X \to Y\} \models X \twoheadrightarrow Y$.

(A8). Ако $X \twoheadrightarrow Y$ е в сила, $Z \subseteq Y$ и за някое W, $W \cap Y = \emptyset$, имаме $W \to Z$, тогава $X \to Z$ също е в сила.

Надеждност и Пълнота на Аксиомите

Теорема 4.21 (Beeri, Fagin и Howard (1977)). *Аксиомите A1–A8 са надеждни и пълни за функционални и многозначни зависимости. Т.е., ако D е множество от функционални и многозначни зависимости над множество атрибути U, и D^+ е множеството от функционални и многозначни зависимости, които следват логически от D,*[35] *тогава D^+ е точно множеството от зависимостите, които следват от D по A1–A8.* □

4.6.2. Допълнителни правила за извод за многозначни зависимости

Има много други правила, които са полезни при правене на изводи от функционални и многозначни зависимости. Разбира се, правилата за обединение, декомпозиция и псевдотранзитивност, посочени в Лема 4.2 все още важат за функционални зависимости. Някои други правила са:

1. *Обединение за многозначни зависимости.*

$$\{X \twoheadrightarrow Y, X \twoheadrightarrow Z\} \models X \twoheadrightarrow XZ$$

2. *Псевдотранзитивност за многозначни зависимости.*

$$\{X \twoheadrightarrow Y, WY \twoheadrightarrow Z\} \models WX \twoheadrightarrow (Z - WY)$$

3. *Смесена псевдотранзитивност.*

$$\{X \twoheadrightarrow Y, XY \twoheadrightarrow Z\} \models X \twoheadrightarrow (Z - Y)$$

4. *Декомпозиция за многозначни зависимости.* Ако $X \twoheadrightarrow Y$ и $X \twoheadrightarrow Z$ са в сила, то $X \twoheadrightarrow (Y \cap Z)$, $X \twoheadrightarrow (Y - Z)$ и $X \twoheadrightarrow (Z - Y)$ също са в сила.

[35] Това означава, че всяка релация, която удовлетворява D, също удовлетворява и зависимостите в D^+.

4.6.3. Базис на зависимост

Теорема 4.22. *Ако U е множеството от всички атрибути, тогава можем да разбием $U - X$ на множества $Y_1, \ldots, Y_k$ от атрибути, т.ч. ако $Z \subseteq (U - X)$, тогава $X \twoheadrightarrow Z \Leftrightarrow Z$ е обединение от някои от Y_i-мата.*

Доказателство. Започваме с $U - X$ в един блок. Да предположим, че в някой момент имаме разбиването $W_1, \ldots, W_n$ и $X \twoheadrightarrow W_i$, за $i = 1, \ldots, n$. Ако $X \twoheadrightarrow Z$ и Z не е обединение от някои W_i-та, тогава заменяме всяко W_i, за което $W_i \cap Z \neq \emptyset$ и $(W_i - Z) \neq \emptyset$, с $W_i \cap Z$ и $(W_i - Z)$. От правилото за декомпозиция: $X \twoheadrightarrow (W_i \cap Z)$ и $X \twoheadrightarrow (W_i - Z)$. Тъй като не може безкрайно да разбиваме едно крайно множество от атрибути, в един момент ще се окаже, че всяко Z, т.ч. $X \twoheadrightarrow Z$, е обединение от някои блокове от разбиването.

По правилото на обединението X многозначно определя обединението на произволно множество от блокове на разбиването. □

Дефиниция 4.19 (Базис на зависимост). Наричаме горните множества $Y_1, \ldots, Y_k$, конструирани за X от множество зависимости D, *базис на зависимост* за X (спрямо D).

Теорема 4.23. *При изчисляване на базиса на зависимост на X спрямо D, е достатъчно да изчислим базиса спрямо множество от многозначни зависимости M, където M се състои от:*

1. *Всички многозначни зависимости в D и*
2. *За всяка функционална зависимост $X \to A_1 \ldots A_n$ в D, множеството от многозначни зависимости $X \twoheadrightarrow A_1, \ldots, X \twoheadrightarrow A_n$, където A_i е единичен атрибут.*

□

Следващата теорема ни дава начин да извлечем нетривиалните функционални зависимости от базиса на зависимост, изчислен спрямо множеството от многозначни зависимости M.

Теорема 4.24. *Ако $A \notin X$, то $X \to A$ е в сила $\Leftrightarrow$*

1. *A е единично множество от базиса на зависимост за X спрямо множеството от зависимости M и*
2. *$\exists Y$ – множество от атрибути, т.ч. $A \notin Y$, $Y \to Z$ е една от дадените зависимости в D и $A \notin Z$.*

□

Алгоритъм 4.6. Изчисляване на базис на зависимост

ВХОД: Множество от зависимости M над множество атрибути U и $X \subseteq U$.

ИЗХОД: Базисът на зависимост за X спрямо M.

МЕТОД: Започваме с колекция от множества S, която впоследствие става базиса на зависимост, който търсим. В началото $S = \{U - X\}$. Докато са възможни промени за S, търсим зависимости $V \longrightarrow\!\!\!\!\!\rightarrow W$ в M и $Y \in S$, т.ч. $Y \cap W \neq \emptyset$, но $Y \cap V = \emptyset$. Заместваме Y с $Y \cap W$ и $Y - W$ в S.

□

4.6.4. Беззагубни съединения

Алгоритъм 4.2 прави проверка дали дадена декомпозиция има беззагубно съединение, при положение че единствените зависимости са функционални. Този алгоритъм може да се обобщи, за да поддържа и многозначни зависимости, но няма правим това сега. В случая на декомпозиция на две схеми има прост тест за беззагубно съединение.

Теорема 4.25. *Нека R е релационна схема и $\rho = (R_1, R_2)$ е декомпозиция на R. Нека D е множество от функционални и многозначни зависимости върху атрибутите на R. Тогава ρ има беззагубно съединение спрямо $D \iff (R_1 \cap R_2) \longrightarrow\!\!\!\!\!\rightarrow (R_1 - R_2)$.*[36] □

4.7. Четвърта нормална форма

Има обобщение на BCNF, наречена четвърта нормална форма, която се отнася за релационни схеми с многозначни зависимости.

Дефиниция 4.20 (Четвърта нормална форма, 4NF). Нека R е релационна схема и D е множеството зависимости за R. Казваме, че R е в *четвърта нормална форма*, ако $\forall\, X \longrightarrow\!\!\!\!\!\rightarrow Y \in D^+$, където $Y \not\subseteq X$, и $XY \neq R$, е случай, в който X е суперключ на R.[37]

Забележете, че ако R е в 4NF, то R е и в $BCNF$.

Доказателство. Да предположим, че R не е в BCNF, защото има някоя функционална зависимост $X \to A$, където X не е суперключ и $A \notin X$. Ако $XA = R$, то със сигурност X е суперключ $\Rightarrow XA \neq R$. По (A7) $\Rightarrow X \to A$ извежда $X \longrightarrow\!\!\!\!\!\rightarrow A$. Т.к. $XA \neq R$ и $A \notin X$, то $X \longrightarrow\!\!\!\!\!\rightarrow A$ е нарушение на 4NF. □

Можем да намерим декомпозиция на R $\rho = (R_1, \ldots, R_n)$, която е с беззагубно съединение спрямо D и всяко R_i е в 4NF, по следния начин. Започваме с $\rho = \{R\}$ и многократно декомпозираме релационните схеми,

[36] Според аксиомата за допълнение, това е еквивалентно на $(R_1 \cap R_2) \longrightarrow\!\!\!\!\!\rightarrow (R_2 - R_1)$.

[37] Дефинициите за "ключ" и "суперключ" не са се изменили; "суперключ" все още означава множество от атрибути, които функционално определят R.

когато намерим нарушение на 4NF подобно на дискусията за простия, но времеотнемащ алгоритъм за BCNF, предхождаща Алгоритъм 4.4. Ако има релационна схема $S \in \rho$, която не е в 4NF спрямо D, проектирано върху S, тогава трябва да има зависимост $X \twoheadrightarrow Y$ за S, където X не е суперключ на S, $Y \neq \emptyset$, $Y \not\subseteq X$ и $XY \neq S$. Можем да приемем, че $X \cap Y = \emptyset$, т.к. $X \twoheadrightarrow Y \models X \twoheadrightarrow (Y - X)$ с прилагане на A1, A7 и правилото за декомпозиция. Тогава заместваме S със $S_1 = XY$ и $S_2 = S - Y$, които трябва да са две релационни схеми с по-малко атрибути от S. По Теорема 4.25, т.к. $(S_1 \cap S_2) \twoheadrightarrow (S_1 - S_2)$, съединението на S_1 и S_2 е беззагубно спрямо $\pi_S(D)$, което в тази секция ще приемем да е множеството от функционални и многозначни зависимости, които следват от D и използват само атрибути от множеството S.

Може да се докаже обобщение на Лема 4.16 за множества от функционални и многозначни зависимости; т.е., че многократната декомпозиция, както е описана по-горе, генерира множество от релационни схеми, които имат беззагубно съединение спрямо D. Единственият важен детайл, който остава, е да определим как се изчислява $\pi_S(D)$.

Теорема 4.26 (Aho, Beeri, Ullman (1979)). *Нека са дадени R, D и S, както е по-горе. Тогава $\pi_S(D)$ може да се пресметне по следния начин:*

1. *Изчисляваме D^+.*
2. *$\forall X \to Y \in D^+$, ако $X \subseteq S$, то $X \to (Y \cap S)$ е в сила за S.*[38]
3. *$\forall X \twoheadrightarrow Y \in D^+$, ако $X \subseteq S$, то $X \twoheadrightarrow (Y \cap S)$ е в сила за S.*
4. *Няма други функционални или многозначни зависимости за S, които да бъдат изведени от факта, че D е в сила за R.*

4.8. Вложени многозначни зависимости

Още едно усложнение се появява, когато се опитваме да декомпозираме една релационна схема R в 4NF, е че може да има определени многозначни зависимости, които очакваме да са в сила, когато проектираме произволна правдоподобна релация r за R върху $X \subseteq R$, като все още не очакваме тези зависимости да са в сила за самото r. Такива зависимости се казва, че са ***вложени*** в R и трябва да внимаваме, когато поставяме всичките ограничения, които вярваме, че са в сила в релациите r за R, без да игнорираме вложената многозначна зависимост. Между другото, вложени функционални зависимости никога не се случват. Може да се покаже, че ако $Y \to Z$ е в сила, когато релацията r над R се проектира върху X, то $Y \to Z$ е в сила и

[38] Забележете, че т.к. $X \to Y \cap S$ също е в D^+, това правило е еквивалентно на правилото за проектиране на функционални зависимости, дадено по-рано.

за r също. Това обаче не е в сила за многозначните зависимости. Ето една по-формална дефиниция:

Дефиниция 4.21 (Вложена многозначна зависимост). Една релация r над схемата R удовлетворява вложената многозначна зависимост $X \twoheadrightarrow Y|Z$, ако многозначната зависимост $X \twoheadrightarrow Y$ се удовлетворява от релацията $\pi_{X \cup Y \cup Z}(r)$.

Забележете, че няма изискване X, Y и Z да са взаимно чужди и по правилата за обединение, декомпозиция и допълнение $X \twoheadrightarrow Y$ е в сила за $\pi_{X \cup Y \cup Z}(r) \iff X \twoheadrightarrow Z$ е в сила; следователно $X \twoheadrightarrow Y|Z$ означава същото като $X \twoheadrightarrow Z|Y$.

Литература

[1] Красимир Манев, *Увод в дискретната математика*, Издателство на НБУ, София, 1998.

[2] Jeffrey D. Ullman, *Principles of database and knowledge-base systems. volume 1: Classical database systems*, Computer Science Press.