# Лекция 6. Проектиране на релационни БД

При проектирането на база от данни се решават два основни проблема:

- По какъв начин да се изобразят обекти от предметната област в абстрактни обекти на модели данни, за да може това изображение да не противоречи на семантиката на предметната област и да бъде по възможност най-добро (ефективно, удобно и т.н.)? Често този проблем се нарича проблем на логическото проектиране на бази от данни.
- Как да се осигури ефективността на изпълнението на заявките към базата от данни, т.е. как, имейки предвид особеностите на конкретната СУБД, да се разположат данните във външната памет, създаването на какви допълнителни структури (например, индекси) да се поиска и т.н.? Този проблем се нарича проблем на физическото проектиране на бази от данни.

В случая на релационните бази от данни трудно се представят общи рецепти в частта на физическото проектиране. Твърде много зависи от използваната СУБД. Например, при работа със СУБД *Ingres* може да се избира един от предлаганите способи за физическа организация на отношенията, при работа със *System R* следва преди всичко да се помисли за клъстеризацията на отношенията и необходимия набор индекси и т.н. Затова ще се ограничим с въпросите за логическото проектиране на релационни бази от данни, които са съществени при използването на всяка релационна СУБД.

Освен това, няма да засягаме много важния аспект на проектирането – определянето на ограниченията за цялостност (с изключение на ограниченията за първичния ключ). Работата е там, че при използването на СУБД с развити механизми на ограниченията за цялостност (например, SQL-ориентираните системи) е трудно да се предложи някакъв общи подход към определението на ограниченията за цялостност. Тези ограничения могат да имат много общ вид, и тяхната формулировка засега се отнася по-скоро към областта на изкуството, отколкото на инженерното майсторство. Най-много, което се предлага по този повод в литературата, това е автоматическа проверка за непротиворечивост на набора ограничения за цялостност.

Така че ще смятаме, че проблемът за проектирането на релационна база от данни се състои в обоснованото вземане на решения за това,

- от какви отношения трябва да се състои БД и
- какви атрибути трябва да имат тези отношения.

## 6.1. Проектиране на релационни бази от данни чрез използване на нормализация

Отначало ще бъде разгледан класическия подход, при който целият процес на проектиране се извършва в термините на релационния модел на данни по метода на последователните приближения към удовлетворителен набор от схеми на отношения. Изходната точка е представянето на предметната област във вид на едно или няколко отношения, и на всяка стъпка от проектирането се произвежда някакъв набор от схеми на отношения, притежаващи най-добри свойства. Процесът на проектирането представлява процес на нормализация на схемите на отношенията, при това всяка следваща нормална форма притежава свойства по-добри, отколкото предишната.

На всяка нормална форма съответствува някакъв определен набор от ограничения, и отношението е в някаква нормална форма, ако удовлетворява свойствения на нея набор от ограничения. Пример за набор от ограничения е ограничението на първа нормална

форма - значенията на всички атрибути на отношение да са атомарни. Тъй като изискването на първа нормална форма е базово изискване за класическия релационен модел данни, то ще считаме, че изходният набор от отношения вече съответствува на това изискване.

В теорията на релационните бази от данни обикновено изпъква следната последователност от нормални форми:

- първа нормална форма (***1NF***);
- втора нормална форма (***2NF***);
- трета нормална форма (***3NF***);
- нормална форма на Бойс-Код (***BCNF***);
- четвърта нормална форма (***4NF***);
- пета нормална форма, или нормална форма на проекцията-съединение (***5NF*** или ***PJ/NF***).

Основни свойства на нормалните форми:

- всяка следваща нормална форма в някакъв смисъл е по-добра от предишната;
- при прехода към следваща нормална форма свойствата на предишните нормални свойства се съхраняват.

В основата на процеса на проектирането лежи методът на нормализация, декомпозиция на отношението, намиращо се в предишната нормална форма, в две или повече отношения, удовлетворяващи изискванията на следващата нормална форма.

Най-важните на практика нормални форми на отношения се основават на фундаменталното в теорията на релационните бази от данни понятие ***функционална зависимост***.

**Определение 1.** *Функционална зависимост*

В отношение *R* атрибутът *Y* функционално зависи от атрибута *X* (*X* и *Y* могат да бъдат съставни) тогава и само тогава, когато на всяка стойност на *X* съответствува точно една стойност на *Y*: *R.X* (r) *R.Y*.

**Определение 2.** *Пълна функционална зависимост*

Функционалната зависимост *R.X* (*r*) *R.Y* се нарича пълна, ако атрибутът *Y* не зависи функционално от кое да е точно подмножество на *X*.

**Определение 3.** *Транзитивна функционална зависимост*

Функционалната зависимост *R.X* -> *R.Y* се нарича транзитивна, ако съществува такъв атрибут *Z*, че съществуват функционалните зависимости *R.X* -> *R.Z* и *R.Z* -> *R.Y* и отсъства функционалната зависимост *R.Z* --> *R.X*. (Ако го нямаше последното изискване щяхме да имаме "неинтересни" транзитивни зависимости във всяко отношение, притежаващо няколко ключа.)

**Определение 4.** *Неключов атрибут*

Неключов атрибут се нарича всеки атрибут на отношение, невлизащ в състава на първичния ключ (в частност, първичния).

**Определение 5.** *Взаимно независими атрибути*

Два или повече атрибута са взаимно независими, ако нито един от тях не е функционално зависим от другите.

### *6.1.1. Втора нормална форма*

Да разгледаме следния пример на схема на отношение:

СЪТРУДНИЦИ-ОТДЕЛИ-ПРОЕКТИ

(СЪТР_НОМЕР, СЪТР_ЗАПЛ, ОТД_НОМЕР, ПРО_НОМЕР, СЪТР_ЗАДАН)

Първичен ключ:

СЪТР_НОМЕР, ПРО_НОМЕР

Функционални зависимости:

СЪТР_НОМЕР -> СЪТР_ЗАПЛ

СЪТР_НОМЕР -> ОТД_НОМЕР

ОТД_НОМЕР -> СЪТР_ЗАПЛ

СЪТР_НОМЕР, ПРО_НОМЕР -> СЪТР_ЗАДАН

Както се вижда, макар че първичен ключ е съставният атрибут СЪТР_НОМЕР, ПРО_НОМЕР, атрибутите СЪТР_ЗАПЛ и ОТД_НОМЕР функционално зависят от част от първичния ключ, атрибута СЪТР_НОМЕР. В резултат не можем да вмъкнем в отношението СЪТРУДНИЦИ-ОТДЕЛИ-ПРОЕКТИ кортеж, описващ сътрудник, който още не изпълнява никакъв проект (първичният ключ не може да съдържа неопределена стойност). При премахване на кортежа ние не само разрушаваме връзката на дадения сътрудник с дадения проект, но загубваме информацията за това, че той работи в някакъв отдел. При превода на сътрудника в друг отдел ще бъдем принудени да модифицираме всички кортежи, описващи този сътрудник, или ще получим несъгласуван резултат. Такива неприятни явления се наричат аномалии на схемата на отношение. Те се отстраняват чрез нормализация.

**Определение 6.** *Втора нормална форма (в това определение се предполага, че единствения ключ на отношение е първичния ключ)*

Отношение ***R*** е във *втора нормална форма* (***2NF***) тогава и само тогава, когато е в ***1NF***, и всеки неключов атрибут напълно зависи от първичния ключ.

Може да се извърши следната декомпозиция на отношението СЪТРУДНИЦИ-ОТДЕЛИ-ПРОЕКТИ на двете отношения СЪТРУДНИЦИ-ОТДЕЛИ и СЪТРУДНИЦИ-ПРОЕКТИ:

СЪТРУДНИЦИ-ОТДЕЛИ (СЪТР_НОМЕР, СЪТР_ЗАПЛ, ОТД_НОМЕР)

Първичен ключ:

СЪТР_НОМЕР

Функционални зависимости:

СЪТР_НОМЕР -> СЪТР_ЗАПЛ

СЪТР_НОМЕР -> ОТД_НОМЕР

ОТД_НОМЕР -> СЪТР_ЗАПЛ

СЪТРУДНИЦИ-ПРОЕКТИ (СЪТР_НОМЕР, ПРО_НОМЕР, СЪТР_ЗАДАН)

Първичен ключ:

СЪТР_НОМЕР, ПРО_НОМЕР

Функционални зависимости:

СЪТР_НОМЕР, ПРО_НОМЕР -> СЪТР_ЗАДАН

Всяко от тези две отношения е в *2NF*, и в тях са отстранени отбелязаните по-горе аномалии (лесно се проверява, че всички указани операции се изпълняват без проблеми).

Ако се допусне наличие на няколко ключа, то определение 6 ще има следния вид:

**Определение 6~**

Отношение *R* е във втора нормална форма (*2NF*) тогава и само тогава, когато то е в *1NF*, и всеки неключов атрибут напълно зависи от всеки ключ *R*.

### *6.1.2. Трета нормална форма*

Да разгледаме още един път отношението СЪТРУДНИЦИ-ОТДЕЛИ, което е в *2NF*. Функционалната зависимост СЪТР_НОМЕР -> СЪТР_ЗАПЛ е транзитивна; тя е следствие от функционалните зависимости СЪТР_НОМЕР->ОТД_НОМЕР и ОТД_НОМЕР -> СЪТР_ЗАПЛ. С други думи, работната заплата на сътрудника всъщност е характеристика не на сътрудника, а на отдела, в който той работи (това не е добро естествено предположение, но е достатъчно за примера).

В резултат няма да можем да запишем в базата от данни информация, характеризираща работната заплата на отдел, дотогава, докато в този отдел не се появи поне един сътрудник (първичният ключ не може да съдържа неопределена стойност). При изтриване на кортежа, описващ последния сътрудник на даден отдел, ще се лишим от информацията за работната заплата в отдела. За да може съгласувано да се измени работната заплата на отдела, ще сме принудени предварително да намерим всички кортежи, описващи сътрудниците на този отдел. Т.е. в отношението СЪТРУДНИЦИ-ОТДЕЛИ както и преди съществуват аномалии. Тях можем да отстраним чрез по-нататъшна нормализация.

**Определение 7.** *Трета нормална форма. (Отново определението се дава при предположение за съществуване на единствен ключ.)*

Отношение *R* е в *трета нормална форма* (***3NF***) тогава и само тогава, ако е в ***2NF*** и всеки неключов атрибут нетранзитивно зависи от първичния ключ.

Може да се извърши декомпозиция на отношението СЪТРУДНИЦИ-ОТДЕЛИ на двете отношения СЪТРУДНИЦИ и ОТДЕЛИ:

СЪТРУДНИЦИ (СЪТР_НОМЕР, ОТД_НОМЕР)

Първичен ключ:

СЪТР_НОМЕР

Функционални зависимости:

СЪТР_НОМЕР -> ОТД_НОМЕР

ОТДЕЛИ (ОТД_НОМЕР, СЪТР_ЗАПЛ)

Първичен ключ:

ОТД_НОМЕР

Функционални зависимости:

ОТД_НОМЕР -> СЪТР_ЗАПЛ

Всяко от тези две отношения е в ***3NF*** и е свободно от отбелязаните аномалии.

Ако се откажем от ограничението, отношението да има единствен ключ, то определението за ***3NF*** ще приеме следната форма:

**Определение 7~**

Отношение *R* е в трета нормална форма (***3NF***) тогава и само тогава, ако е в *2NF*, и всеки неключов атрибут не е транзитивно зависим от някакъв ключ *R*.

На практика трета нормална форма на схемите на отношения е достатъчна в повечето случаи, и с привеждането към трета нормална форма процесът на проектиране на релационна база от данни обикновено завършва. Но понякога е полезно да се продължи процеса на нормализация.

### *6.1.3. Нормална форма на Бойс-Код*

Да разгледаме следния пример на схема на отношение:

СЪТРУДНИЦИ-ПРОЕКТИ (СЪТР_НОМЕР, СЪТР_ИМЕ, ПРО_НОМЕР, СЪТР_ЗАДАН)

Възможни ключове:

СЪТР_НОМЕР, ПРО_НОМЕР

СЪТР_ИМЕ, ПРО_НОМЕР

Функционални зависимости:

СЪТР_НОМЕР -> СЪТР_ИМЕ

СЪТР_НОМЕР -> ПРО_НОМЕР

СЪТР_ИМЕ -> СЪТР_НОМЕР

СЪТР_ИМЕ -> ПРО_НОМЕР

СЪТР_НОМЕР, ПРО_НОМЕР -> СЪТР_ЗАДАН

СЪТР_ИМЕ, ПРО_НОМЕР -> СЪТР_ЗАДАН

В този пример предполагаме, че личността на сътрудник напълно се определя както от номера му, така и от името (това отново не е добро жизнено предположение, но е достатъчно за примера).

В съответствие с определение7~ отношението СЪТРУДНИЦИ-ПРОЕКТИ е в *3NF*. Но фактът, че има функционални зависимости на атрибутите на отношението от атрибута, явяващ се част от първичния ключ, води до аномалии. Например, за да се измени името на сътрудник с даден номер по съгласуван начин, ще трябва да модифицираме всички кортежи, включващи неговия номер.

**Определение 8.** *Детерминанта*

Детерминанта - произволен атрибут, от който напълно функционално зависи някой друг атрибут.

**Определение 9.** *Нормална форма на Бойс-Код*

Отношение *R* е в *нормална форма на Бойс-Код* (***BCNF***) тогава и само тогава, когато всяка детерминанта е възможен ключ.

Очевидно е, че това изискване не е изпълнено за отношението СЪТРУДНИЦИ-ПРОЕКТИ. Може да се извърши декомпозицията му към отношенията СЪТРУДНИЦИ и СЪТРУДНИЦИ-ПРОЕКТИ:

СЪТРУДНИЦИ (СЪТР_НОМЕР, СЪТР_ИМЕ)

Възможни ключове:

СЪТР_НОМЕР

СЪТР_ИМЕ

Функционални зависимости:

СЪТР_НОМЕР -> СЪТР_ИМЕ

СЪТР_ИМЕ -> СЪТР_НОМЕР

СЪТРУДНИЦИ-ПРОЕКТИ (СЪТР_НОМЕР, ПРО_НОМЕР, СЪТР_ЗАДАН)

Възможен ключ:

СЪТР_НОМЕР, ПРО_НОМЕР

Функционални зависимости:

СЪТР_НОМЕР, ПРО_НОМЕР -> СЪТР_ЗАДАН

Възможна е алтернативна декомпозиция, ако се избере за основа СЪТР_ИМЕ. В двата случая получаваните отношения СЪТРУДНИЦИ и СЪТРУДНИЦИ-ПРОЕКТИ са в *BCNF*, и не са им присъщи отбелязаните аномалии.

### *6.1.4. Четвърта нормална форма*

Да разгледаме следната схема на отношение:

ПРОЕКТИ (ПРО_НОМЕР, ПРО_СЪТР, ПРО_ЗАДАН)

Отношението ПРОЕКТИ съдържа номерата на проектите, за всеки проект – списъкът на сътрудниците, които могат да изпълняват проекта, и списъкът задания, предвидени в проекта. СЪТРУДНИЦИ може да участвуват в няколко проекта, и различни ПРОЕКТИ може да включват еднакви задания.

Всеки кортеж на отношение свързва проект със сътрудник, участващ в този проек, и заданието, което сътрудникът изпълнява в рамките на дадения проект (предполагаме, че всеки сътрудник, участващ в проекта, изпълнява всички задания, предвидени от този проект). По причина на сформулираните по-горе условия единственият възможен ключ на отношението е съставният атрибут ПРО_НОМЕР, ПРО_СОТР, ПРО_ЗАДАН, и няма никакви други детерминанти Следователно, отношението ПРОЕКТИ е в *BCNF*. Но при това то има недостатъци: ако, например, някой сътрудник се присъедини към дадения проект, трябва да се вмъкнат в отношението ПРОЕКТИ толкова кортежи, колкото задания са предвидени в него.

**Определение 10.** *Многозначни зависимости*

В отношението *R* (*A, B, C*) съществува многозначна зависимост *R.A -> -> R.B* тогава и само тогава, когато множеството значения на *B*, съответстващо на двойка стойности на *A* и *C*, зависи само от *A* и не зависи от *C*.

В отношението ПРОЕКТИ съществуват следните две многозначни зависимости:

ПРО_НОМЕР -> -> ПРО_СЪТР

ПРО_НОМЕР -> -> ПРО_ЗАДАН

Леко се доказва, че в общия случай в отношението *R* (*A, B, C*) съществува многозначната зависимост *R.A -> -> R.B* тогава и само тогава, когато съществува многозначната зависимост *R.A -> -> R.C*.

По-нататъшната нормализация на отношенията, подобни на отношението ПРОЕКТИ, се основава на следната теорема:

**Теорема на Фейджин**

Отношението *R* (*A, B, C*) може да се проецира без загуби в отношенията *R1* (*A, B*) и *R2* (*A, C*) тогава и само тогава, когато съществува ***MVD*** *A -> -> B | C*.

Под проециране без загуби се разбира такъв начин на декомпозиция на отношение, при който изходното отношение напълно и без излишност се възстановява чрез естествено съединение на получените отношения.

**Определение 11.** *Четвърта нормална форма*

Отношение *R* е в четвърта нормална форма (***4NF***) тогава и само тогава, ако в случай на съществуване на многозначна зависимост *A -> -> B* всички останали атрибути на *R* функционално зависят от А.

В нашия пример може да се извърши декомпозиция на отношението ПРОЕКТИ на две отношения ПРОЕКТИ-СЪТРУДНИЦИ и ПРОЕКТИ-ЗАДАНИЯ:

ПРОЕКТИ-СЪТРУДНИЦИ (ПРО_НОМЕР, ПРО_СЪТР)

ПРОЕКТИ-ЗАДАНИЯ (ПРО_НОМЕР, ПРО_ЗАДАН)

Тези две отношения са в *4NF* и са свободни от отбелязаните аномалии.

### *6.1.5. Пета нормална форма*

Във всички разгледани до момента нормализации се извършва декомпозиция на едно отношение в две. Понякога това не може да се направи, но е възможна декомпозицията в по-голям брой отношения, всяко от които има по-добри свойства.

Да разгледаме, например, отношението

СЪТРУДНИЦИ-ОТДЕЛИ-ПРОЕКТИ (СЪТР_НОМЕР, ОТД_НОМЕР, ПРО_НОМЕР)

Да предположим, че един и същи сътрудник може да работи в няколко отдела и работи във всеки отдел над няколко проекта. Първичният ключ на това отношение е пълната съвокупност от атрибутите му, отсъстват функционални и многозначни зависимости.

Затова отношението е в *4NF*. Но в него могат да съществуват аномалии, които може да се отстранят чрез декомпозиция на три отношения.

**Определение 12.** *Зависимост за съединение*

Отношение $R$ ($X$, $Y$, ..., $Z$) удовлетворява зависимостта за съединение ***** ($X$, $Y$, ..., $Z$) тогава и само тогава, когато $R$ се възстановява без загуби чрез съединение на своите проекции на $X$, $Y$, ..., $Z$.

**Определение 13.** *Пета нормална форма*

Отношение $R$ е в пета нормална форма (*нормална форма на проекция-съединение* - ***PJ/NF***) тогава и само тогава, когато всяка зависимост на съединение в $R$ следва от съществуването на някакъв възможен ключ в $R$.

Ще въведем следните имена на съставните атрибути:

СО = {СЪТР_НОМЕР, ОТД_НОМЕР}

СП = {СЪТР_НОМЕР, ПРО_НОМЕР}

ОП = {ОТД_НОМЕР, ПРО_НОМЕР}

Да предположим, че в отношението СЪТРУДНИЦИ-ОТДЕЛИ-ПРОЕКТИ съществува зависимостта на съединение:

***** (СО, СП, ОП)

В примерите лесно се показва, че при вмъквания и изтривания на кортежи може да възникнат проблеми. Тях можем да отстраним чрез декомпозиция на изходното отношение в три нови отношения:

СЪТРУДНИЦИ-ОТДЕЛИ (СЪТР_НОМЕР, ОТД_НОМЕР)

СЪТРУДНИЦИ-ПРОЕКТИ (СЪТР_НОМЕР, ПРО_НОМЕР)

ОТДЕЛИ-ПРОЕКТИ (ОТД_НОМЕР, ПРО_НОМЕР)

Пета нормална форма е последната нормална форма, която може да бъде получена чрез декомпозиция. Нейните условия са достатъчно нетривиални, и на практика ***5NF*** не се използва. Да отбележим, че зависимостта на съединение е обобщение както на многозначна зависимост, така и на функционална зависимости.

## 6.2. Семантично моделиране на данни, ER-диаграми

Широкото разпространение на релационните СУБД и използването им в най-разнообразни приложения показва, че релационния модел данни е достатъчен за

моделиране на предметни области. Но проектирането на релационна база от данни в термините на отношенията на основата на разгледания механизъм на нормализация често представлява много сложен и неудобен за проектировчика процес.

При това се проявява ограничеността на релационния модел данни в следните аспекти:

- Моделът не предоставя достатъчно средства за представяне смисъла на данните. Семантиката на реалната предметна област трябва да е независима от модела по начина на представяне в главата на проектировчика. В частност, това се отнася до споменатия вече проблем на представянето на ограниченията за цялостност.
- За много приложения е трудно да се моделира предметната област на основата на плоски таблици. В редица случаи в най-начален стадий на проектиране на проектировчика се налага да извърши насилие над себе си, за да опише предметната област във вид на една (възможно, даже ненормализирана) таблица.
- Макар целият процес на проектиране да става на основата на отчитането на зависимости, релационния модел не предоставя средства за представяне на тези зависимости.
- Независимо, че процесът на проектиране започва с отделянето на няколко съществени за приложението обекти от предметната област ("същности") и показване на връзките между тези същности, релационният модел данни не предлага апарат за разделяне на същностите и връзките.

### *6.2.1. Семантични модели данни*

Потребностите на проектировчиците на бази от данни от по-удобни и мощни средства за моделиране на предметна област предизвикват появата на направлението на семантичните модели данни. Макар, че всеки развит семантичен модел данни, както и релационния модел, включва структурна, манипулационна и цялостна част, главното предназначение на семантичните модели е осигуряването на възможност за изразяване на семантиката на данните.

Преди, да разгледаме накратко особеностите на един от най-разпространените семантични модели, ще се спрем на техните възможни приложения.

Най-често на практика семантичното моделиране се използва на първия стадий от проектирането на база от данни. При това в термините на семантичния модел се извежда концептуалната схема на базата от данни, която след това ръчно се преобразува в релационна (или друга) схема. Този процес се изпълнява под управлението на методики, в които достатъчно точно са определени всички етапи на такова преобразуване.

Не толкова често се реализира автоматизирана компилация на концептуалната схема в релационна. При това са известни два подхода: на основата на явно представяне на концептуалната схема като изходна информация за компилатора и построяването на интегрирани системи за проектиране с автоматизирано създаване на концептуална схема на основата на интервюта с експерти по предметната област. И в единия, и в другия случай в резултат се извежда релационна схема на базата от данни в трета нормална форма (по-точно, не са ни известни системи, осигуряващи по-високо ниво на нормализация).

Накрая, третата възможност, която още не е излязла (или сега излиза) извън границите на изследователските и експериментални проекти, - това е работа с базата от данни в семантичния модел, т.е. СУБД, основани на семантични модели данни. При това отново се разглеждат два варианта: осигуряване на потребителски интерфейс на основата на семантичния модел данни с автоматично изобразяване на конструкциите в

релационния модел данни (това е задача примерно от такова ниво сложност, като автоматичната компилация на концептуалната схема на база от данни в релационна схема) и пряка реализация на СУБД, основана на семантичен модел данни. Най-близко до втория подход са съвременните обектно-ориентирани СУБД, моделите данни на които по много параметри са близки до семантичните модели (макар че в някои аспекти те са по-мощни, а в други – по-слаби).

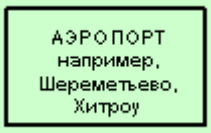
### *6.2.2. Основни понятия на модела Entity - Relationship (Същност - Връзки)*

На използването на разновидности на *ER*-модела се основават повечето съвременни подходи за проектиране на бази от данни (основно релационните). Моделът е бил предложен от **Чен** (***Chen***) през 1976 г. Моделирането на предметната област се базира на използването на графични диаграми, включващи неголям брой разнородни компоненти. Във връзка с нагледността на представянето на концептуалните схеми на бази от данни *ER*-моделите са получили широко разпространение в *CASE* системите, поддържащи автоматизираното проектиране на релационни бази от данни. Сред множеството разновидности на *ER*-моделите един от най-развитите се използва в *CASE* системата на фирмата *ORACLE*. Ще го разгледаме като се съсредоточим върху структурната част на този модел.

Основните понятия на *ER*-модели са **същност**, **връзка** и **атрибут.**

**Същност** - това е реален или представян обект, информацията за който трябва да се съхранява и да бъде достъпна. В диаграмите на *ER*-модела същността се представя като правоъгълник, съдържащ името на същността. При това името на същността е име на тип, а не на някакъв конкретен екземпляр от този тип. За по-голяма изразителност и по-добро разбиране името на същност може да се съпровожда с примери за конкретни обекти от този тип.

По-долу е изобразена същността АЕРОПОРТ с примерни обекти Шереметьево и Хитроу:



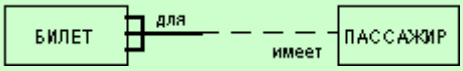

Всеки екземпляр на същност трябва да е отличим от всеки друг екземпляр на тази същност (това изискване е в някаква степен аналогично на изискването за отсъствие на кортежи-дубликати в релационните таблици).

**Връзка** – това е графично изобразена асоциация, установявана между две същности. Тази асоциация винаги е бинарна и може да съществува между две различни същности или между същност и нея самата (рекурсивна връзка). Във всяка връзка изпъкват два края (в съответствие със съществуващата двойка свързани същности), на всеки от които се посочва името на края на връзката, степента на края на връзката (колко екземпляра от дадената същност се свързват), задължителност на връзката (т.е. всеки ли екземпляр от дадената същност трябва да участвува в дадената връзка).

Връзката се представя като линия, свързваща двете същности или водеща от същност към нея самата. При това в мястото на "стиковката" на връзката със същността се използва триточков вход в правоъгълника на същността, ако за тази същност във връзката могат да се използват много (many) екземпляри на същността, и едноточков вход, ако във връзката може да участвува само един екземпляр на същността. Задължителният край на връзката се изобразява с плътна линия, а незадължителния – с прекъсната линия.

Както и същността, връзката е типово понятие, всички екземпляри на двете двойки свързвани същности се подчиняват на правилата за свързване.

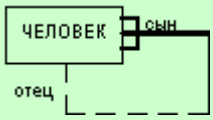
В изобразения по-долу пример връзката между същностите БИЛЕТ и ПЪТНИК свързва билети и пътници. При това края на същността с име "*за*" позволява да се свързват с един пътник повече от един билет, при това всеки билет трябва да е свързан с някакъв пътник. Краят на същността с име "*има*" означава, че всеки билет може да принадлежи само на един пътник, при това пътникът не е длъжен да има поне един билет.




Лаконична устна трактовка на изобразената диаграма е следната:

- Всеки БИЛЕТ е предназначен за един и само един ПЪТНИК;
- Всеки ПЪТНИК може да има един или повече БИЛЕТИ.

В следващия пример е изобразена рекурсивна връзка, свързваща същността ЧОВЕК със самата нея. Краят на връзката с име "*син*" определя факта, че един баща може да има повече от един син. Краят на връзката с име "*баща*" означава, че не всеки човек може да има синове.




Лаконична устна трактовка на изобразената диаграма е следната:

- Всеки ЧОВЕК е син на един и само на един ЧОВЕК;
- Всеки ЧОВЕК може да е баща на един или повече ХОРА ("ЧОВЕЦИ").

Атрибут на същност е всеки детайл, който служи за уточнение, идентификация, класификация, числова характеристика или изразяване на състоянието на същността. Имената на атрибутите се записват в правоъгълника, изобразяващ същността, под името на същността и се изобразяват с малки букви, ако е възможно, с примери.

Уникален идентификатор на същност е атрибут, комбинация от атрибути, комбинация от връзки или комбинация от връзки и атрибути, уникално отличаваща всеки екземпляр на същност от другите екземпляри на същността от този тип.

### *6.2.3. Нормални форми на ER-схеми*

Както и в релационните схеми на бази от данни, в *ER*-схемите се въвежда понятието нормални форми, при това техният смисъл много близко съответствува на смисъла на релационните нормални форми. Формулировките за нормални форми на *ER*-схемите правят по-понятен смисъла на нормализацията на релационните схеми. Ще приведем само много кратки и неформални определения на трите първи нормални форми.

В *първата нормална форма* на *ER*-схема се отстраняват повтарящите се атрибути или групи атрибути, т.е. изваждат се на показ неявните същности, "маскирани" като атрибути.

Във *втора нормална форма* се остраняват атрибутите, зависещи само от част на уникалния идентификатор. Тази част на уникалния идентификатор определя отделната същност.

В *трета нормална форма* се отстраняват атрибутите, зависещи от атрибути, невлизащи в уникалния идентификатор. Тези атрибути са основата на отделната същност.

### *6.2.4. По- сложни елементи на ER-модела*

Спряхме се само на най-основните и най-очевидни понятия на *ER*-модела на данни. Към по-сложните елементи на модела се отнасят следните:

- *Подтипове* и *супертипове* на същности. Както в езиците за програмиране с развити типови системи (например, в езиците за обектно-ориентирано програмиране), се въвежда възможност за наследяване на типа на същност, изхождайки от един или няколко супертипа. Интересни нюанси са свързани с необходимостта от графично изображение на този механизъм.
- Връзки "*many-to-many*". Понякога е необходимо да се свържат същности по такъв начин, че от двата края на връзката могат да присъстват няколко екземпляра на същност (например, всички членове на кооператив съвместно владеят имуществото на кооперативa). За това се въвежда разновидност на връзка "много-към-много".
- *Уточняеми степени на връзка*. Понякога е по-полезно да се определи възможния брой екземпляри на същност, участващи в дадена връзка (например, на служител се разрешава да участвува в не повече, от три проекта едновременно). За изразяване на това семантично ограничение се разрешава да се укаже на края на връзка нейната максимална или задължителна степен.
- *Каскадни изтривания на екземпляри на същности*. Някои връзки са толкова силни (разбира се, в случай на връзка "един-към-много"), че при изтриването на опорен екземпляр на същността (съответстващ на края на връзката "един") трябва да се премахнат и всички екземпляри на същността, съответстващи на края на връзката "много". Съответното изискване за "каскадно изтриване" може да се формулира при определяне на същност.
- *Домейни*. Както и при релационния модел данни е полезна възможността за определяне на потенциално допустимото множество значения на атрибут на същност (домейна).

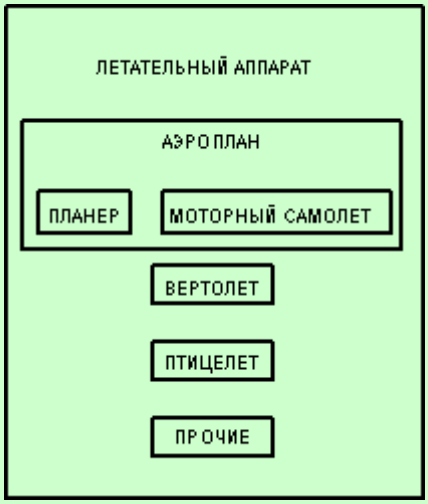
Тези и други по-сложни елементи на модела данни "Същност-Връзки" го правят съществено по-мощен, но одновременно в някаква степен усложняват използването му. Разбира се, при реално използване на *ER*-диаграмите за проектиране на бази от данни е необходимо да се запознаем с всички възможности.

Тук ще се спрем по-подробно само на един от споменатите елементи - подтип на същност.

Същност може да бъде разцепена на два или повече взаимно изключващи се подтипа, всеки от които включва общи атрибути и/или връзки. Тези общи атрибути и/или връзки явно се определят един път на по-високо ниво. В подтиповете може да се определят собствени атрибути и/или връзки. По принцип подтипизацията може да продължава на по-ниски нива, но опитът показва, че в повечето случаи са достатъчни две-три нива.

Същност, на основата на която се определят подтипове, се нарича *супертип*. Подтиповете трябва да образуват пълно множество, т.е. всеки екземпляр на супертипа трябва да се съотнася към някакъв подтип. Понякога за пълнота се налага да се определя допълнителен подтип ДРУГИ.

Пример: Супертип ЛЕТАТЕЛЕН АПАРАТ




Как би трябвало това да се чете? От супертипа: ЛЕТАТЕЛЕН АПАРАТ, който трябва да е АЕРОПЛАН, ВЪРТОЛЕТ, ПТИЦЕЛЕТ или ДРУГ ЛЕТАТЕЛЕН АПАРАТ. От подтипа: ВЪРТОЛЕТ, който се отнася към типа ЛЕТАТЕЛЕН АПАРАТ. От подтипа, който е едновременно супертип: АЕРОПЛАН, който се отнася към типа ЛЕТАТЕЛЕН АПАРАТ и трябва да е ПЛАНЕР или МОТОРЕН САМОЛЕТ.

Понякога е удобно да има две или повече различни разбивания на същност на подтипове. Например, същността ЧОВЕК може да бъде разбита на подтипове по професионален признак (ПРОГРАМИСТ, ДОЯРКА и т.н.), а може - по полов признак (МЪЖ, ЖЕНА).

### *6.2.5. Получаване на релационна схема от ER-схема*

**Стъпка 1.** Всяка проста същност се превръща в таблица. Простата същност е същност, която не е подтип и няма подтипове. Името на същността става име на таблица.

**Стъпка 2.** Всеки атрибут става възможен стълб със същото име; може да се избира по-точен формат. Стълбовете, съответстващи на незадължителни атрибути, могат да съдържат неопределени значения; стълбовете, съответстващи на задължителни атрибути - не могат.

**Стъпка 3.** Компонентите на уникалния идентификатор на същността се превращат в първичен ключ на таблицата. Ако има няколко възможни уникални идентификатора, се избира най-използвания. Ако в състава на уникалния идентификатор влизат връзки, към стълбовете на първичния ключ се добавя копие на уникалния идентификатор на същността, който е в отдалечения край на връзката (този процес може да продължава рекурсивно). За именуване на тези стълбове се използват имената на краищата на връзките и/или имената на същностите.

**Стъпка 4.** Връзки много-към-един (и един-към-един) стават външни ключове. Т.е. прави се копие на уникалния идентификатор от края на връзката "един", и съответните стълбове съставят външния ключ. Незадължителните връзки съответстват на стълбове, допускащи неопределени значения; задължителните връзки – на стълбове, недопускащи неопределени значения.

**Стъпка 5.** Индекси се създават за първичния ключ (уникален индекс), за външните ключове и за тези атрибути, на които се предполага, че основно ще се базират заявките.

**Стъпка 6.** Ако в концептуалната схема присъстват подтипове, то са възможни два способа:

- всички подтипове в една таблица (*а*)
- за всеки подтип - отделна таблица (*б*)

При използване на способ (*а*) таблица се създава за най-външния супертип, а за подтиповете може да се създават представяния. В таблицата се добавя поне един стълб, съдържащ кода на ТИПА; той става част от първичния ключ.

При използване на метод (б) за всеки подтип от първо ниво (за по-ниските - представяния) супертип се създава с помощта на представянето ***UNION*** (от всички таблици на подтипове се избират общите столбове – столбовете на супертипа).

| **Всички в една таблица** | **Таблица - на подтип** |
|---|---|
| *Предимства* | |
| Всичко се съхранява заедно<br>Лесен достъп до супертип и подтипове<br>Изискват се по-малко таблици | По-ясни правила за подтипове<br>Програмите работят само с нужните таблици |
| *Недостатъци* | |
| Твърде общо решение<br>Изисква се допълнителна логика за работа с различните набори стълбци и различни ограничения<br>Потенциално тясно място (свързано с блокировки)<br>Стълбците на подтипове трябва да са незадължителни<br>В някои СУБД за съхранение на неопределени стойности е необходима допълнителна памет | Твърде много таблици<br>Смущаващи стълбци в представяне UNION<br>Потенциална загуба на производителност при работа през UNION<br>Над супертип са невъзможни модификации |

**Стъпка 7.** Има два начина за работа при наличие на изключващи връзки:

- общи домейн (*а*)
- явни външни ключове (*б*)

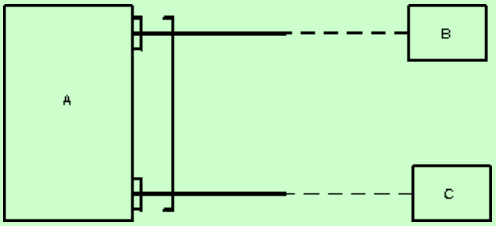
Ако оставащите външни ключи са всички в един домейн, т.е. имат общ формат (способ (*а*)), то се създават два стълба: идентификатор на връзката и идентификатор на същността. Стълбът на идентификатора на връзката се използва за различаване на връзките, покривани от дъгата на изключението. Стълбът на идентификатора на същността се използва за съхраняване на стойностите на уникалния идентификатор на същността на далечния край на съответната връзка.

Ако резултиращите външни ключове не се отнасят към един домейн, то за всяка връзка, покривана от дъгата на изключение, се създават явни стълбци на външни ключове; всички тези стълбци могат да съдържат неопределени значения.

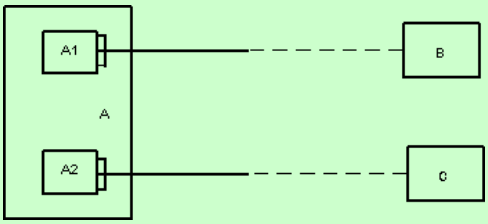
| **Общи домейн** | **Явни външни ключове** |
|---|---|
| *Предимства* | |
| Нужни са само два стълба | Условия за съединения - явни |
| *Недостатъци* | |
| Двата допълнителни атрибута трябва да се използват в съдинения | Твърде много стълбове |

*Алтернативни модели на същности:*

Вариант 1 (лош)




**Вариант 2** (съществено по-добър, ако подтиповете действително съществуват)




**Вариант 3** (приложим при наличие на осмислен супертип D).