

## БАЗИ ОТ ДАННИ

Според определението на Джеймс Мартин БД<sup>1</sup> представляват съвкупност от взаимосвързани и съхранявани съвместно данни при наличието на такова минимално дублиране, което осигурява тяхното оптимално използване за едно или повече потребителски приложения. Данните се съхраняват така, че да са независими от приложенията, които ги използват. За добавяне на нови, за модифициране на съществуващите и за търсене на данни в БД се използва общ управляем способ.

БД е именувана съвкупност от взаимосвързани помежду си данни, отнасящи се към дадена предметна област и използвана от едно или повече приложения. Общото програмно управление на БД се осъществява от програмна среда, която осигурява създаване, поддържане в актуално състояние и организация на достъпа до данните от различните потребители. Такива среди се наричат системи за управление на бази от данни (СУБД). Сред водещите съвременни представители на СУБД са Microsoft SQL Server, IBM DB2, MySQL, Oracle, IBM IMS и др.

Базите от данни се характеризират със следните особености:

- данните се описват и съхраняват отделно от приложенията;
- една база от данни може да обслужва множество приложения;
- непротиворечивост и съгласуваност на данните след извършване на всяка транзакция;
- възможност за безпроблемно разширяване на системата, т.е. добавяне на нови приложения и актуализация на съществуващите.

Основните изисквания към БД са:

- интеграция и свързаност;
- минимално дублиране на данните;
- независимост на данните – поддържат се 2 равнища на независимост - логическа и физическа. Логическата независимост се изразява в независимост на общото логическо описание на БД от приложенията, които я

---

<sup>1</sup> Martin, James, Computer Data-Base organization, Prentice-Hall, 1977

използват, а физическата – в независимост на физическия модел на БД от логическия<sup>2</sup>;

- непротиворечивост и цялостност на данните;
- защита на данните.

В БД могат да се поддържат следните видове отношения между същностите<sup>3</sup> и техните атрибути или между атрибутите: 1:1, 1:M, M:1 и M:M. Отношенията са двупосочни и показват колко екземпляра от едната същност съответстват на един екземпляр от друга същност и обратно. Така например отношение 1:M имаме, когато на един екземпляр от същност А съответстват няколко екземпляра от същност Б, а на един екземпляр от същност Б съответства един екземпляр от същност А, т.е. връзката е 1:M в едната посока, а в обратната – 1:1. Отношение M:M има когато и в двете посоки връзката е 1:M. Такъв тип отношения (M:M) са неопределени и сложни и изискват специално внимание при проектиране на модела на базите от данни.

### **Модели на БД**

Моделът на базите данни определя тяхната структура и начините за съхраняване, организиране и манипулиране с данните. Различаваме три основни модела на данните: йерархичен, мрежови и релационен.

#### **Йерархичен модел на БД**

Йерархичният модел възниква и се прилага широко в края на 60-те години на миналия век. Първата популярна йерархична база данни е IBM Information Management System (IBM IMS), която е разработена първоначално за нуждите на космическата програма Аполо през 1966-1968 г. Способността ѝ да поддържа големи бази данни и да осъществява бързо и безпроблемно интензивни транзакции я правят популярна и днес, предимно в сферата на здравеопазването, банкови и финансови институции. Друга широко използвана йерархична СУБД е Cache на InterSystems.

Йерархичният модел представлява йерархична дървовидна структура (вж. фиг. 12.1). Всяка йерархична структура се състои от върхове (сегменти), в които

---

<sup>2</sup>Физическият и логическият модел се разглеждат по-подробно в тема 13.

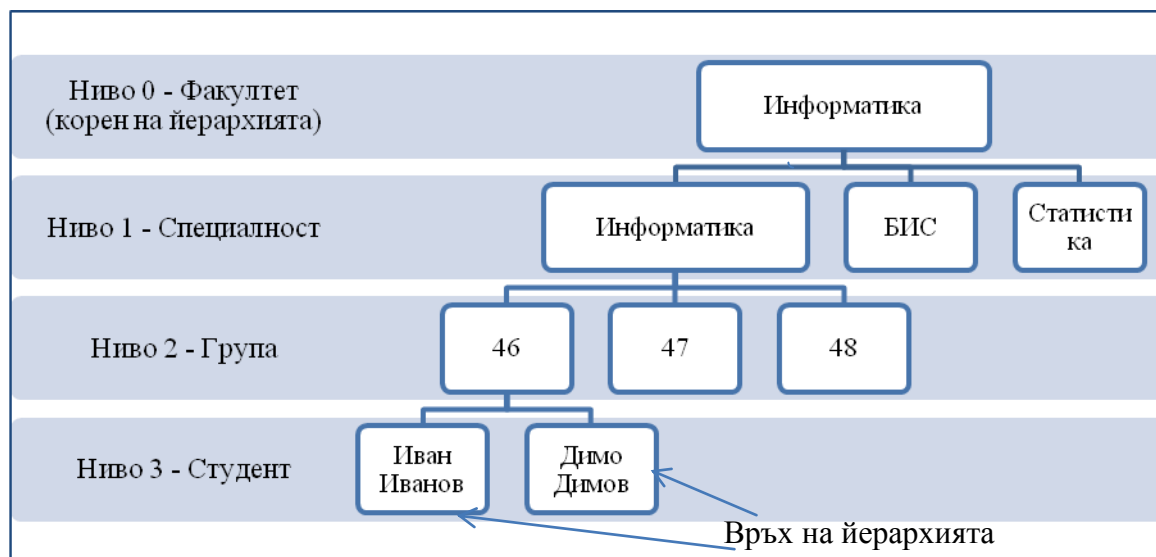
<sup>3</sup>Същност в контекста на базите данни са обекти, явления, концепции, процеси и др. от предметната област, за които се съхраняват данни и които могат да се опишат чрез набор от атрибути

се описват данните и обектите на предметната област, а дъгите дефинират връзките между тях. Върховете биват: породени (зависими), пораждащи (независими) и родителски сегмент по отношение на един или няколко породени. В йерархичната БД има един сегмент (връх), който представлява корен на йерархията и винаги се използва за вход в БД.

Йерархията винаги започва от корена. Всеки връх може да бъде и породен и пораждащ, или само породен или само пораждащ. Добавянето на сегменти може да стане хоризонтално и вертикално. Достъпът до породен връх става от корена пред пораждащия го. Отношенията, които поддържа йерархичната БД, са 1 : М, 1 : 1 и 1 : 0. Задаването на връзките става по подразбиране, като се следва йерархията, т.е. връзките не се именуваат, а се подразбират.

Коренът на йерархичната структура може да има няколко екземпляра. Един екземпляр на кореновидният сегмент заедно с всички екземпляри на сегмента, породен от кореновия, образуват една логическа йерархична БД. Логическата йерархична БД е подмножество на общата БД.

Йерархичните бази от данни, заедно с мрежовите, са навигационни (navigational database), тъй като обръщението към записите се осъществява чрез указатели (pointers) и пътеки (paths) от родителските записи.



**Фиг.1. Йерархичен модел на база данни**

Предимствата на йерархичните база от данни са<sup>4</sup>:

<sup>4</sup> Singh, S., Database Systems: Concepts, Design and Applications, Pearson Education, 2011

- простота на връзките – поддържаните от модела връзки са логически и концептуално опростени;
- ефикасност – йерархичните бази данни са особено ефективни при работа с голям обем данни и брой транзакции;
- поддържа се цялост на данните – не може да се добави породен сегмент, без да съществува пораждащият го;
- опит в използването им – този модел на базата данни се използва още от 60-те години на двадесети век, като за изминалия дълъг период са разработени редица СУБД и приложения;

Наред с предимствата обаче, йерархичните бази от данни притежават и някои недостатъци като:

- недостатъчна гъвкавост. Тъй като достъпът до всеки сегмент се осъществява само чрез пораждащия го, добавянето и премахването на сегменти от йерархията може да бъде значително затруднено. Така например при изтриване на даден сегмент ще се изтрият всички породени от него сегменти;
- не се поддържат връзки М:М – такъв тип връзки са доста често срещани и за да се отразят в йерархичната база от данни е необходимо да се създаде изкуствена йерархия, което усложнява модела. Поради това йерархични бази от данни е препоръчително да се използват само когато между същностите съществува естествена йерархия.
- Липсва физическа независимост на БД – При йерархичните бази данни достъпът до данните се осъществява посредством път (access path), който зависи от физическата структура на БД. При промяна на структурата на базата данни, този път също трябва да се промени, което може да изисква и промени в съответните приложения, използващи базата данни;
- Липсва утвърден стандарт за йерархични бази данни;
- Сложен програмен език за манипулиране с данните – специалистите трябва да познават много добре физическото разпределение на данните, което изисква задълбочени знания и голям опит с комплексна система от указатели.

### **Мрежови модел на базата от данни**

Мрежовият модел е предложен през 1969г. и формализиран като CODASYL<sup>5</sup>-модел. Този модел е подобен на йерархичния, но за разлика от него една породена същност може да е свързана с няколко пораждаци. При този модел се поддържат следните отношения между данните: М:М, 1:1, 1:М, М:1, М:М. Връзките могат да бъдат разнопосочни. В резултат мрежовата структура като правило е достатъчно сложна и може да съдържа различни връзки между два сегмента, цикли, обръщения на сегмент сам към себе си.

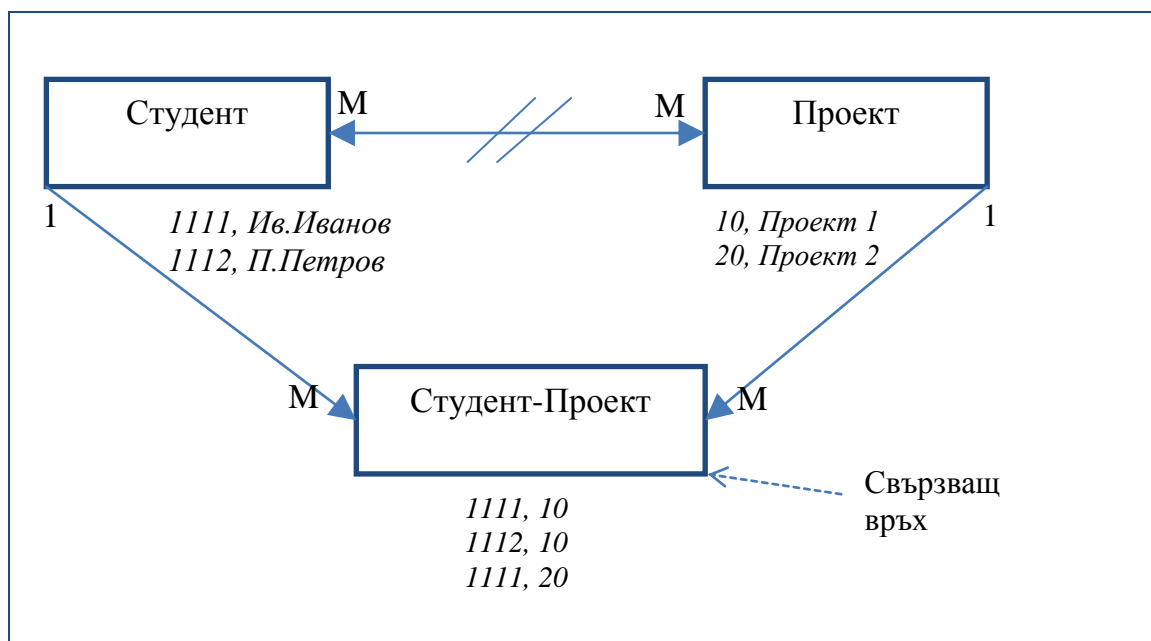
Основните компоненти на мрежовия модел са: тип на записа (record type, същност), атрибути от данни (data items, полета) и връзки (links). Връзките между същностите формират набори (sets). Във всеки набор участват поне два типа записи. Единият се нарича собственик (owner), което съответства на пораждания (родителски) запис при йерархичния модел, а вторият - член (member) на набора, което отговаря на породения тип запис при йерархичния модел. Връзките се именуват, т.е. се задават в явен вид, докато в йерархичната БД се подразбират. Мрежовите бази от данни, подобно на йерархичните, са навигационни, но при тях пътят за достъп до данните се определя от връзките между собственика и членовете на набора. Характерно при мрежовия модел е, че един член може да участва в множество набори, т.е. да притежава повече от един собственик, което позволява поддържането на връзки М:М.

Множествените връзки се преобразуват във връзки от типа 1:М, както е показано на фиг. 12.2. За целта се създава свързващ връх, като се разрушава<sup>6</sup> връзката М:М. Свързващият връх участва в два набора и има два собственика – Студент и Проект. Вход в мрежовата БД може да бъде всеки връх. Няма корен, който да служи само за това. В така създадената структура може да се влезе през върха “Студент” и да се изведе информация за проектите, по които е работил студентът, чрез свързващия елемент „Студент-Проект“.

---

<sup>5</sup> CODASYL – Conference on Data System Languages

<sup>6</sup>Под „разрушава“ се има предвид преобразуването на връзката между същностите от концептуалния модел на данните.



**Фиг. 12.2. Мрежови модел на базата данни**

Предимствата на мрежовия модел на базата данни са:

- Простота на връзките;
- Възможност за поддържане на връзки М:М чрез добавяне на свързващи върхове;
- По-лесен достъп до всеки сегмент от модела в сравнение с йерархичния модел;
- Поддържа се цялост на данните – не може да съществува член на набора без да съществува собственик;
- Има разработени стандарти за този модел като DBTG/CODASYL.

Недостатъци на мрежовия модел:

- Сложност на модела – мрежовият модел, подобно на йерархичния, е сложен за проектиране и реализиране. Изискват се много специфични познания относно механизма за достъп до данните и тяхната вътрешна структура;
- Липсва физическа независимост на данните – при промяна на физическата структура на БД се изисква промяна в подсхемите, за да може приложенията да осъществят безпроблемен достъп до данните. Мрежовият модел, подобно на йерархичния, поддържа само логическа независимост на данните.

## Релационен модел на БД

Релационният модел е представен за пръв път от Е.Код през 1970 г. Съгласно този модел, релационните бази са изградени от “плоски”, двумерни таблици, наричани още релации или отношения. Релацията (Relation) е двумерна таблица, състояща се от редове и колони. В колоните са отразени характеристиките на обектите. В редовете се съдържа наборът от значения, които имат отделните характеристики за всеки екземпляр на обекта. Друга важна особеност на релациите е атомарният характер на атрибутите – всеки атрибут приема едно точно определено значение, т.е. не може да се структурира на податрибути и да има множество значения. Имената на колоните са уникални и в релационната таблица не съществуват редове с едно и също съдържание.

Съвкупност от значенията на един атрибут се нарича домейн (domain). Синоними на това понятие са понятията „колона“ и „поле“. Всеки домейн има име и тип. Типовете полета са числово, текстово (символно), дата и час, логическо, мемо и др. Поддръждането на домейните в релациите е произволно.

Един ред от релационната таблица, в който са отразени значенията на домейните за конкретен екземпляр от обекта се нарича кортеж. Всеки ред от релационната таблица има първичен ключ, чрез който се идентифицира. Първичният ключ (primary key) се състои от едно или няколко полета, които уникално идентифицират всеки ред (запис) от отношението. Когато първичният ключ съдържа само едно поле, говорим за прост първичен ключ, а при две и повече полета – за съставен първичен ключ.

Връзките между два обекта в предметната област се представят чрез външни ключове. Външният ключ (foreign key) е поле в таблицата, което съответства на първичен ключ в друга таблица. Таблицата, в която полето играе ролята на първичен ключ, се нарича главна (master), а тази, съдържаща външния ключ се нарича свързана (related).

В релационните БД се поддържат следните типове връзки (relationships): 1:1; 1:M; M:1. Връзки M:M не могат директно да се поддържат и изискват добавянето на свързваща релация между първоначално участващите в множествената връзка отношения.

Предимства на релационните бази от данни:

- Простота на модела. Релационните бази са по-опростени от йерархичните и мрежовите. Не се изисква проектантите на БД да имат детайлна информация относно физическото съхраняване на данните. Релационните модели са ясни, логични и позволяват с тях да работят по-широк кръг потребители;
- Поддържат логическа и физическа независимост на данните, което улеснява проектирането, внедряването и използването им;
- За релационните бази има разработени гъвкави и мощни средства за извличане на данни посредством структурирания език за заявки (Structured Query Language – SQL). За разлика от йерархичните и мрежови бази, при релационните необичайни и сложни заявки могат да се реализират значително по-бързо и лесно;

Недостатъци на релационните бази данни:

- Изискват се по-мощни изчислителни ресурси за изпълнение на транзакции и заявки. Поради това обикновено релационните бази са по-бавни в сравнение с йерархичните и мрежовите. С усъвършенстването на информационните и компютърни технологии обаче, този недостатък не е толкова съществен, както в началото на използването на релационните бази;
- Лекотата, с която се проектират и внедряват релационните бази, може да доведе до пренебрегване на качеството на логическия модел за сметка на скоростта на изграждане. Разработчици на релационни бази могат да бъдат не само висококвалифицирани експерти, както при йерархичните и мрежовите бази, което може да бъде предпоставка за допускане на грешки в модела. С нарастване на обема на базата от данни лошият модел може да доведе до забавяне на транзакциите и заявките и до загуба на данни.

## **ПРОЕКТИРАНЕ НА РЕЛАЦИОННИ БАЗИ ОТ ДАННИ**

### **1.Етапи на проектирането на БД**

Етапите на проектирането на БД се определят от равнищата на независимост на БД и от архитектурата на БД, предложена от Изследователската



група по системи за управление на бази данни (Study Group on Data Base Management Systems) на Американския институт за стандарти (ANSI/SPARC).

### **Независимост на данните**

През 1985г. в списание „Computerworld“ Е.Код формулира 12 правила<sup>7</sup>, според които една база данни може да се счита за пълно релационна. Две от тях разглеждат въпроса за равнищата на независимост на базите от данни, а именно:

*Правило 8: „Приложните програми са независими от физическото съхраняване и методите за достъп до данните“.* Това правило дефинира физическата независимост на БД, т.е. промените във физическото разположение на файловете не оказват влияние върху начина, по който приложните програми взаимодействат с БД.

*Правило 9: „Промените в логическата схема на БД, като добавяне, разделяне и комбиниране на таблици, не оказват влияние върху приложенията“.* Тук е необходимо да подчертаем, че логическата независимост не се отнася до всички промени в схемата на БД, а само до тези, които не са свързани със загуба на информация. Операции, свързани със загуба на информация, са например премахване на таблици, колони или връзки между таблици. Такива промени изискват и промени в приложенията, ако те използват данни от модифицираните обекти. Промените, които не водят до загуба на информация (information-preserved changes), като добавяне на нови таблици или връзки, не оказват влияние върху вече съществуващите обекти в БД и следователно не изискват промени в приложенията.

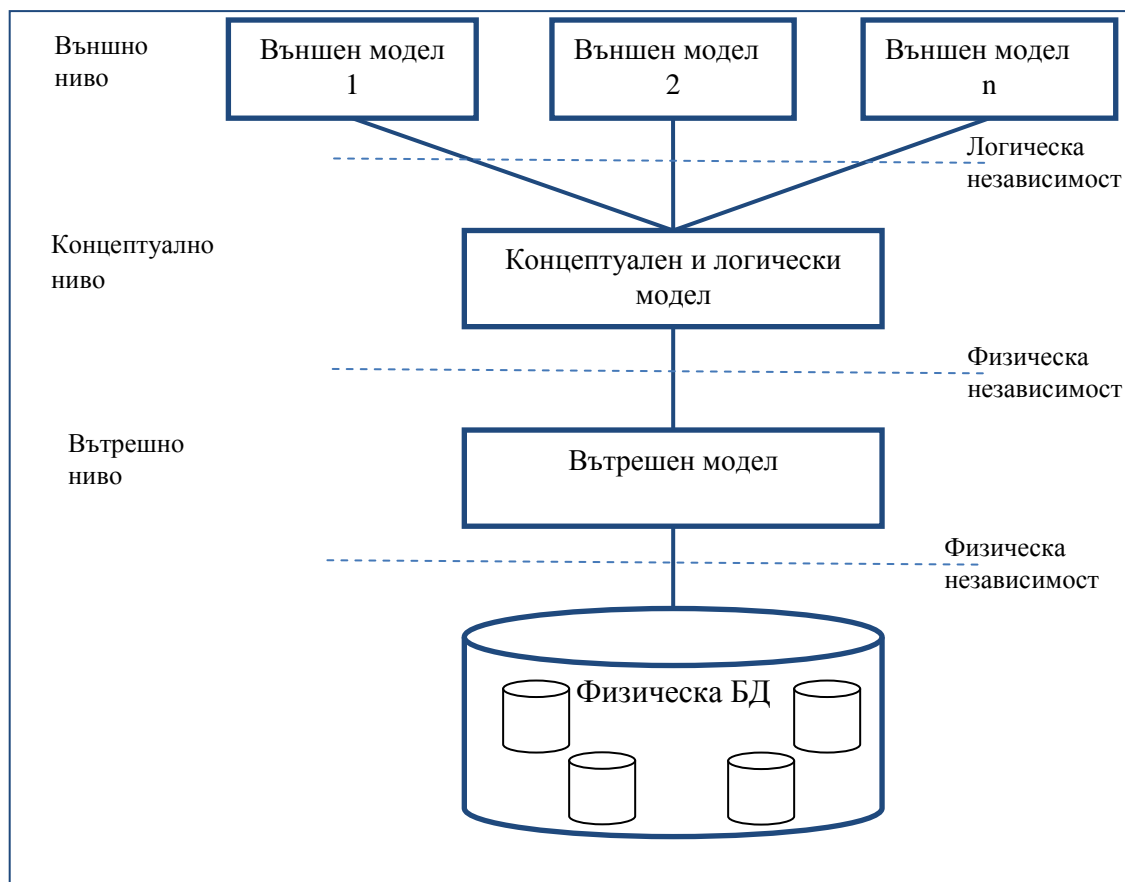
Релационните бази от данни поддържат и двете равнища на независимост – физическа и логическа, докато мрежовите и йерархичните поддържат само логическа независимост. Липсата на физическа независимост при навигационните бази (йерархична и мрежова) се компенсира от бързия достъп до данните и високата скорост на осъществяване на транзакциите.

### **Архитектура на БД**

---

<sup>7</sup> Всъщност правилата са 13, номерирани от 0 до 12, но в литературата са познати като „Дванадесетте правила на Е.Код“ (E.Codd's twelve rules)

Повечето съвременни реляционни СУБД поддържат предложената през 1975 г. ANSI/SPARC архитектура на БД, включваща три нива: външно, концептуално и вътрешно (вж. фиг.13.1).



**Фиг.1. Архитектура на ANSI/SPARC<sup>8</sup>**

**Външно ниво.** На това ниво се изграждат външните модели, които отразяват представите на отделните потребители на базата данни. Външните модели се наричат още потребителски изгледи (users' views) или външни схеми (externalschemas). Външният модел описва БД на най-високо ниво на абстракция и включва само тези същности, атрибути и връзки, които представляват интерес за потребителите. В обхвата на една предметна област може да има няколко външни модела. Всички те се обединяват в един концептуален модел на БД, но са адаптирани към представите на различни потребители и приложения, което обуславя и възможни разлики между външните модели. Разликите най-често се изразяват в: различно представяне на данните; добавяне на изчисляеми полета, които не присъстват в концептуалния модел; комбиниране на няколко таблици и

<sup>8</sup> адапт. по Singh, S., Database Systems: Concepts, Design and Applications, Prentice Hall, 2009

др. От гледната точка на потребителите базата данни се идентифицира с използваните от тях външни модели. Външните схеми се описват чрез външен език за описание на данните (External Data Definition Language).

**Концептуално ниво.** На това ниво се създава концептуалният модел на БД. Този модел включва всички същности от предметната област и връзките между тях. Концептуалният модел представя логическата структура на БД и се описва чрез концептуална схема на БД<sup>9</sup> (conceptual schema), която съдържа: всички същности, атрибути и връзки; ограничения върху данните, семантична информация, правила за поддържане на цялостност, интегритет и защита на данните. Концептуалният модел не разглежда физическото съхраняване на данните.

Концептуалното ниво поддържа всички външни модели, т.е. всички данни, с които оперират потребителите трябва да се съдържат или да могат да бъдат изведени от концептуалния модел, затова при концептуалното проектиране трябва да се изхожда от потребителските представи и изисквания. За създаване на концептуалната схема на БД се използва концептуален език за описание на данните (Conceptual Data Definition Language).

**Вътрешно ниво.** На това ниво БД се описва чрез вътрешен модел, който описва физическите структури за съхранение на данните – файлове, дялове, записи, индекси, методите за достъп и организация на данните във физическата среда.

Вътрешната схема на БД показва как описаните в концептуалния модел същности и релации в действителност се съхраняват. Чрез вътрешния модел се дефинират: разпределение на пространството за съхранение на БД; описание на записите, включително и размер на всяко поле; индекси; методи за добавяне на нови записи; техники за компресиране и криптиране на данните. Вътрешната схема се създава с помощта на структуриран език за заявки (Structured Query Language – SQL) или физически език за описание на данните (Physical Data

---

<sup>9</sup> В някои източници вместо „концептуална схема“ се използва понятието „логическа схема“ (logical schema). Считаме, че между тези две понятия трябва да има разграничение и логическият модел следва да се разглежда като преобразуван концептуален модел в зависимост от избрания модел на БД – йерархичен, мрежови или релационен.

Description Language). Вътрешният модел зависи от избраната СУБД за изграждането на физическата база от данни.

Предимства на трислойната архитектура на БД:

- Всеки потребител има достъп до една и съща база данни, но чрез дефиниран специално за неговите потребности изглед. Външният модел, използван от даден потребител или приложение е независим от останалите външни модели;
- Взаимодействието между потребителите и базата данни не зависи от физическото ѝ разположение;
- Вътрешната схема на БД не зависи от физическата организация на съхраняваните данни и обратно, промените в физическата организация не изискват промени във вътрешния модел;
- Концептуалният модел може да се променя без да се променят всички външни модели.

**Основните етапи на проектиране на БД** включват<sup>10</sup>:

1. **Концептуално проектиране на БД** Създаването на концептуален модел започва с анализ на изискванията на потребителите и документиране на тяхната представа за данните. На етапа на концептуалното проектиране се обединяват всички външни модели в единен концептуален модел на данните, който не зависи от избраната за реализация СУБД. Концептуалният модел трябва да включва всички същности, атрибути и връзки, дефинирани в предметната област и да позволява изпълнението на необходимите транзакции с данните.

2. **Логическо проектиране на БД.** След избор на подходяща СУБД концептуалният модел се преобразува в логически, който описва модела на данните в терминологията на избраната СУБД. При избор на релационни СУБД концептуалният модел се преобразува в релационен модел, в който същностите се представят като релационни таблици, атрибутите – като колони, а връзките се реализират чрез механизма „първичен ключ – външен ключ“. Ако в

---

<sup>10</sup> Кашева, М., О.Тулешкова, Ив.Куюмджиев, Бази от данни, Наука и икономика, Варна, 2009 и Connolly, T., C.Begg, R.Holowczak, Business Database Systems, Addison-Wesley, 2008

концептуалния модел съществуват множествени връзки, връзки тип:подтип и множествени атрибути, те трябва да се преобразуват съгласно изискванията на релационните бази данни. На етапа на логическо проектиране се извършва и нормализация на данните. Логическият модел на БД е независим от физическата структура на данните. Той зависи от избрания модел на БД (мрежови, йерархичен или релационен), но не и от конкретната СУБД за реализация

3. **Физическо проектиране на БД.** На този етап се създава физическият модел на БД, който представя начинът на реализиране на логическия модел в средата на конкретната СУБД. При физическото проектиране може да се изгради и модел на трансформация, представящ съответствието между обектите в логическия и физическия модели. С цел подобряване на производителността на БД при необходимост се извършва денормализация на БД. Основните стъпки при физическото проектиране на БД са свързани с<sup>11</sup>: проектиране на физическите таблици, методите за получаване на изчисляеми полета и правилата за интегритет на данните, анализ на транзакциите и избор на организация на файловете и индексите.

## **2.Нормализация на отношенията в БД**

**Нормализацията** е операция по преобразуване и усъвършенстване на релационните структури с оглед създаването на ефективни релационни бази данни (РБД), в които е минимизирано дублирането на данните и няма проблеми при добавяне, обновяване и изтриване на елементи от БД. Нормализацията е мощен инструмент за структуриране на БД в хода на нейното проектиране и се изпълнява на етапа на разработка на логически модел.

### ***Основни понятия, свързани с нормализацията***

**Функционална зависимост.** В отношение  $R(A^*, B)$ <sup>12</sup> атрибут  $B$  функционално зависи от атрибут  $A$ , ако на всяко значение на атрибут  $A$  съответства не повече от едно значение на атрибут  $B$  ( $A$  еднозначно определя  $B$ ). Атрибут  $A$  се третира като ключов атрибут – ключ на отношението.

---

<sup>11</sup>Connolly, T., C.Begg, R.Holowczak, Business Database Systems, Addison-Wesley, 2008

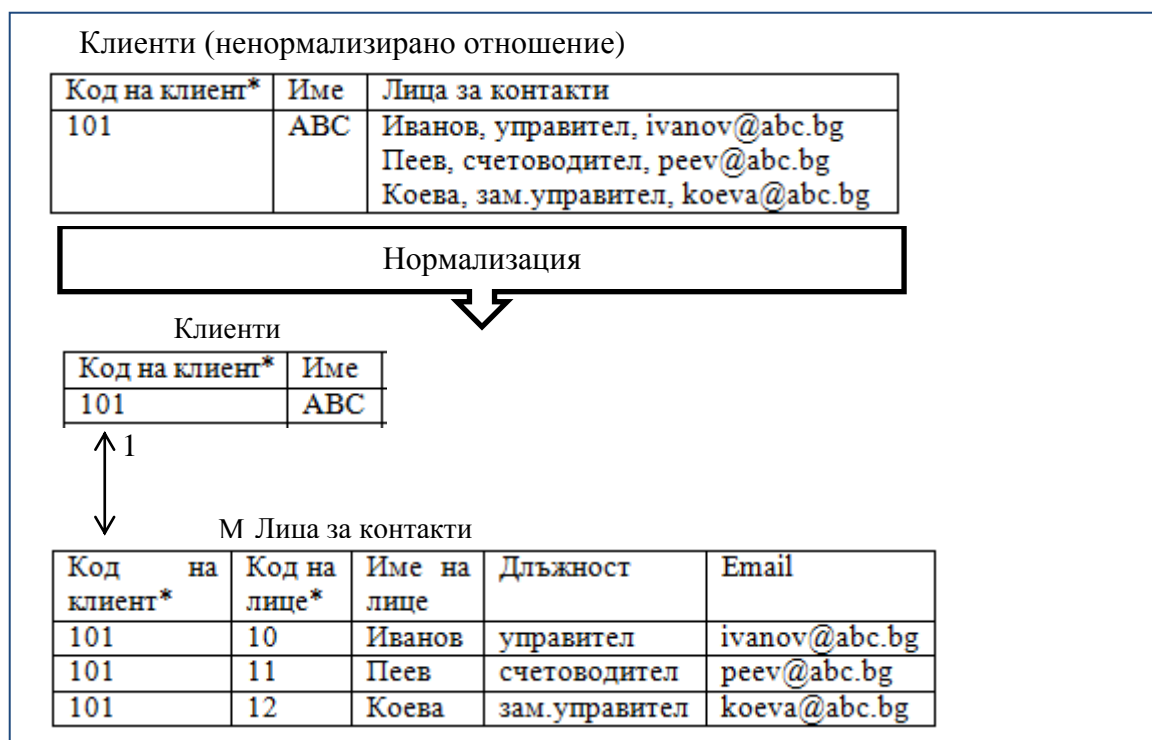
<sup>12</sup> Със символа „\*“ ще обозначаваме ключови атрибути

*Функционална пълна зависимост.* В отношение  $R(A^*, B^*, C)$  атрибут  $C$  се намира в пълна функционална зависимост от множеството  $(A, B)$ , ако във всеки момент от време зависи от цялото множество  $(A, B)$ , но не и от някаква негова съставна част. Пълна функционална зависимост се търси когато първичният ключ на отношението е съставен.

При нормализирането на базите данни се използват т.нар. нормални форми. Известни са 9 нормални форми, всяка следваща от които включва предходната. Обикновено една релационна база се счита за нормализирана, ако отговаря на изискванията за трета нормална форма, т.е. при нея са спазени едновременно изискванията на първа, втора и трета нормални форми. Нормализирането на отношение се постига чрез разделянето му на няколко релации, всяка от които отговаря на изискванията за съответната форма.

**Първа нормална форма:** Едно отношение е релационно, ако се намира в първа НФ, т.е. всички неключови атрибути зависят **функционално** от първичния ключ и домейните имат атомарен характер, т.е. не са множествени и нямат структура.

*Пример.* Фирма  $X$  съхранява данни за своите клиенти в отношение Клиент (Код на клиент\*, Име на клиент, Лица за контакти) (вж.фиг.13.2). Атрибутът „Лица за контакти“ е множествен, тъй като на едно значение на ключовия атрибут „Код на клиент“ съответстват повече от едно значения на „Лица за контакти“. Атрибутът „Лица за контакти“ има и структура, която се състои от името на лицета, длъжността и имейл адрес. Нормализацията на отношението „Клиенти“ съгласно изискванията на първа нормална форма изисква множественият атрибут, заедно с ПК, да се отделят в отделна релация, в която да се разкрие структурата му, както е показано на фи.12.2. Между релациите „Клиенти“ и „Лица за контакти“, получени в резултат на нормализацията, се създава връзка от тип 1:M.



Фиг.2. Нормализация в първа нормална форма

**Втора нормална форма.** Едно отношение се намира във втора нормална форма, ако е било в първа и всички неключови атрибути **функционално пълно** зависят от първичния ключ. Ако в отношението присъства атрибут, който зависи от отделно подмножество на съставния първичен ключ, то той, заедно с подмножеството се отделят в самостоятелно отношение, като първичен ключ на новото отношение става подмножеството на съставния ПК на изходното отношение.

*Пример:* Фирма X съхранява данни за доставените стоки в отношение Доставки (Номер на доставка\*, Дата на доставка, Номер на заявка, Код на доставчик, Код на стока\*, Количество, Ед.цена) Доставките се изпълняват от доставчици на база предварително изпратени към тях заявки. Тъй като с една доставка могат да се доставят няколко стоки, първичният ключ на отношението „Доставки“ е съставен от атрибутите „Номер на доставка“ и „Код на стока“. Отношението отговаря на изискванията на първа нормална форма, но има съставен първичен ключ и трябва да се изследва дали отговаря на изискванията на втора нормална форма. Полетата „Дата на доставка“, „Номер на заявка“ и

„Код на доставчик“ зависят само от „Номер на доставка“, но не и от целия съставен първичен ключ. Непълните функционални зависимости между тези полета и първичният ключ водят до дублиране на едни и същи данни за всеки ред от релационната таблица „Доставки“, отнасящ се до една и съща доставка (вж.фиг. 13.3). За да се нормализира отношението, то трябва да се раздели на две релации, както е показано на фиг.13.3. В едната релация „Доставки“ се включват само полетата, които зависят от „Номер на доставка“, а в другата релация „Доставени стоки“ със съставен първичен ключ „Номер на доставка“ + „Код на стока“ се включват полетата „Количество“ и „Ед.цена“, които зависят функционално пълно от първичния ключ.



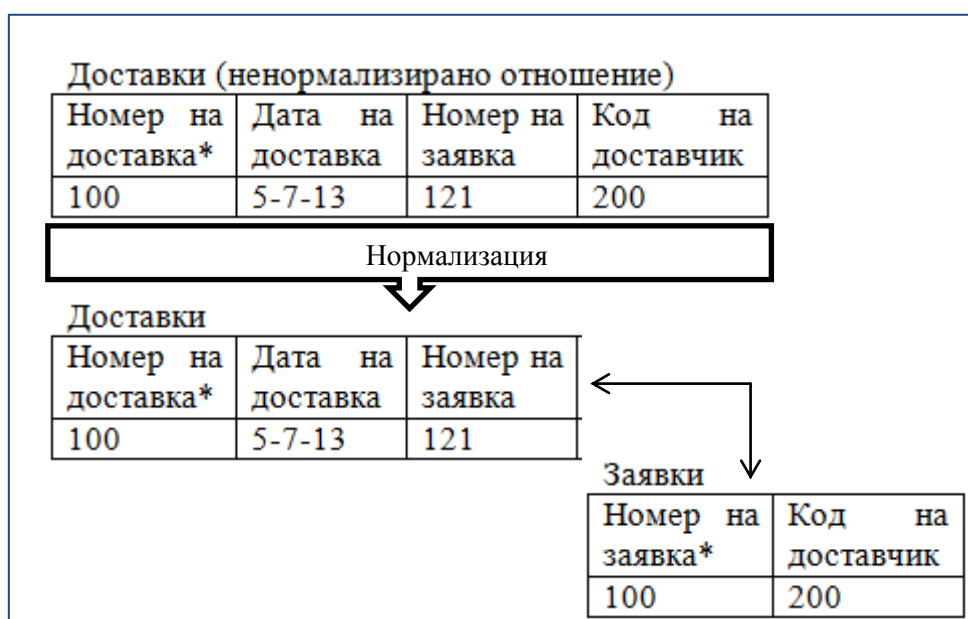
Фиг. 3. Нормализация във втора нормална форма

**Трета нормална форма.** В отношение  $R(A^*, B, C)$  ако  $B$  зависи от  $A$  и  $C$  зависи от  $B$ , се счита, че и  $C$  зависи от  $A$ . Ако обаче  $A$  не зависи от  $B$  и  $B$  не зависи от  $C$ , се казва, че  $C$  транзитивно зависи от  $A$ . Едно отношение е в трета НФ, ако е било във втора НФ и от него са отстранени транзитивните зависимости. Друга формулировка на трета нормална форма гласи: „едно отношение е в трета нормална форма тогава и само тогава, когато всички неключови атрибути зависят само и единствено от целия първичен ключ“.

*Пример:* Нека разгледаме отношението Доставки (Номер на доставка\*, Дата на доставка, Номер на заявка, Код на доставчик), получено в резултат на



нормализацията във втора нормална форма на ненормализираното отношение „Доставки“ (вж. фиг. 13.4). Отношението е във втора нормална форма, но в него има транзитивна зависимост, която се открива като се изследват зависимостите между неключовите атрибути „Номер на заявка“ и „Код на доставчик“. От една страна „Номер на заявка“ и „Код на доставчик“ зависят функционално от първичния ключ „Номер на доставка“, тъй като на едно значение на ПК съответства не повече от едно значение на неключовите атрибути. От друга страна обаче „Номер на заявка“ не зависи от „Код на доставчик“, тъй като един доставчик може да изпълнява множество заявки. „Номер на доставка“ също не зависи от „Номер на заявка“, защото една заявка може да се изпълни с много доставки и следователно „Код на доставчик“ транзитивно зависи от ПК „Номер



на доставка“.

**Фиг.4. Нормализация в трета нормална форма**

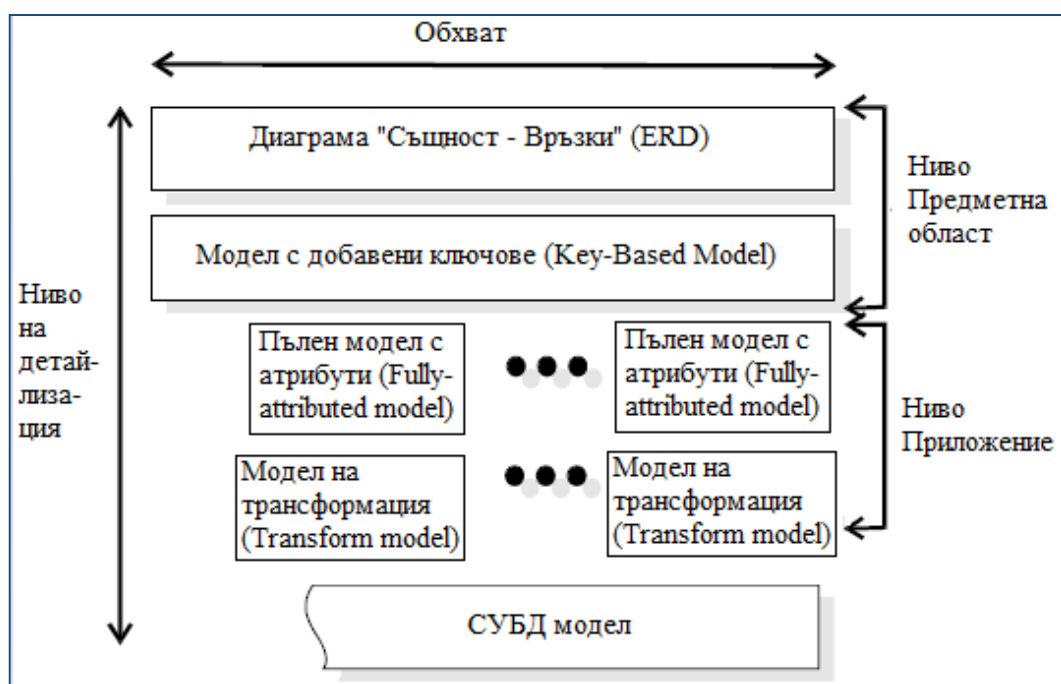
### Проектиране на БД с ERwinDataModeler

Най-широко използваният модел при концептуалното и логическо проектиране на БД е моделът „Същност – Връзки“, който се описва чрез диаграми, представящи в графичен вид обектите и връзките между тях (Entity Relationship Diagram - ERD). Понякога разработката на модел на БД е трудоемка работа, която изисква синхронизиране на усилията на екип от проектанти с цел

обединяване на всички външни потребителски модели в единно концептуално описание. Използването на автоматизирана среда за разработка на модел на базата данни значително улеснява процеса на проектиране, като подпомага създаването на качествени и пълни модели.

ERWIN Data Modeler<sup>13</sup> на компанията ComputerAssociates осигурява платформа за създаване и управление на графични модели на бази данни и складове от данни. Продуктът поддържа методологиите IDEF1X<sup>14</sup> и IE (Information Engineering), базирани на модела «Същност-Връзки».

Моделирането на данните в ERwinDataModeler представлява процес на създаване на модели на различно ниво на детайлизация, при спазване на принцип на проектиране «отгоре-надолу». Моделите се изграждат на пет нива, както е показано на фиг.13.5, като първите три модела са логически (EntityRelationshipDiagram, Key-BasedModel, FullyAttributedModel), а останалите два (TransformationModel, DBMSModel) – физически, т.е. зависят от избраната СУБД.



**Фиг. 5. Модели на данните в ERwin**

<sup>13</sup> <http://erwin.com/products/data-modeler> (25.06.2013 г.)

<sup>14</sup> <http://www.edef.com/IDEF1x.htm> (25.06.2013 г.)

Моделът „Същност- връзки“ (**Entity-Relationship Diagram /ERD/**) е първият модел, който се създава и е с най-високо ниво на обобщаване. В модела се представят основните същности и техните връзки. Възможно е да не присъстват всички същности, а само най-важните в предметната област. Всяка същност се представя с наименование. Връзките между същностите се определят по тип и кардиналност и се именуват. Кардиналността на връзката между две същности А и Б се дефинира в зависимост от това колко екземпляра от същност Б съответстват на един екземпляр от А, и колко екземпляра от същност А съответстват на един екземпляр от същност Б. В зависимост от кардиналността си, връзките могат да бъдат: 1:0; 1:0,1; 1:0,1,M; 1:1; 1:1,M, M:M. Въпреки, че множествени връзки M:M не се поддържат в релационните бази, те могат да се отразяват в първоначалната диаграма «Същност-Връзки», но задължително при следващите модели трябва да се трансформират.

Наименованието представлява обикновено израз, включващ глагол, и може да се зададе за двете посоки на връзката поотделно. Например «купува» и «се купува» могат да се зададат като име на връзката между същностите «Клиент» и «Стока» съответно в посока от «Клиент» към «Стока» и от «Стока» към «Клиент». Наименованието и кардиналността на връзката позволяват «прочитането» на връзката и по-лесното ѝ осмисляне. Във връзките 1:1, 1:M, същността, която е от страната «1» на връзката се нарича родителска, независима или пораждаща, а тази, която е от страната «M» на връзката – породена или зависима.

Вторият модел, **Модел с добавени ключове (Key-BasedModel /KBM/)**, включва всички същности, първични, външни и алтернативни ключове. Обхватът на KBM е същият като на ERD, но включва повече детайли като: свързващи същности, първични, външни и алтернативни ключове. Този модел е основа за разработката на по-детайлните логически и физически модели на данните, използвани от различните приложения. Ако в ERD са били включени връзки от тип M:M или Supertype:Subtype, те трябва да се преобразуват.

**Пълният атрибутивен модел (FullyAttributedModel)** е нормализиран релационен модел, прилагащ изискванията поне на трета нормална форма и

включващ всички същности, атрибути (ключови и неключови) и връзки. Обхватът на този модел съвпада с обхвата на информационната система.

Целта на **Модела на трансформация (TransformationModel)** е да поддържа преобразуването на логическият пълен атрибутивен модел във физически модел съгласно избраната СУБД. Моделът позволява оценка на степента на изпълнение на изискванията към информационната база от страна на физическите структури на БД и идентифициране на възможностите за разширяване и ограниченията на БД.

**СУБД модел (DBMS Model)** е физически модел, реализиран в конкретна СУБД. ERwin позволява изграждане на този модел чрез генериране на физическа БД за най-използваните СУБД (SQLServer, MySQL, Oracle, DB2, ODBS, Sybase). Формират се индекси, прилагат се механизми за реализиране на кардиналността на връзките и др. Обхватът на този модел може да бъде едно или няколко приложения.

#### **Последователност на работа при създаване на логически модел на БД**

Логическите модели се разработват в следната последователност:

1. Диаграма „Същност-Връзки“ (ERD)
2. Модел с ключове (Key-Based Model)
3. Пълен атрибутивен модел (Fully Attributed Model)

#### **Разработване на диаграма „Същност-Връзки“ (ERD)**

След създаването на нов модел се добавят основните същности, идентифицирани в предметната област и се представят връзките между тях.

В зависимост от връзките, в които участват, същностите биват:

- Независими същности (Independent Entities). Такива същности участват във връзки, в които играят роля на родител (Parent, пораждащи, главни същности). Изобразяват се чрез правоъгълник.
- Зависими по отношение на идентифициране на екземплярите (Identification Dependent Entities). За идентифициране на екземплярите от зависимата същност се използват първичните ключове на независимата същност, т.е. ПК на независимата същност, като външни ключове в зависимата, са част от

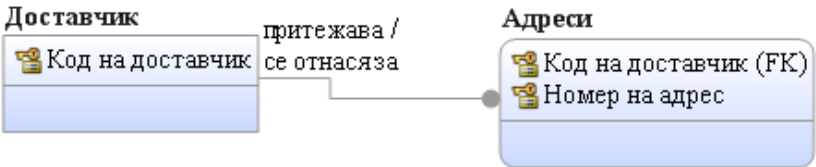
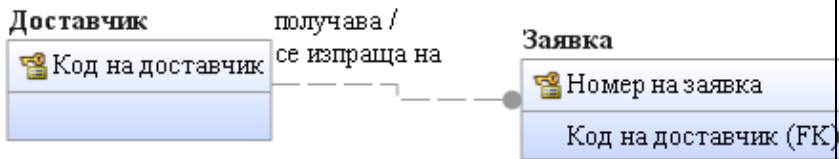
ПК на зависимата същност. Изобразяват се чрез правоъгълник със заоблени върхове.

- Зависими по отношение на съществуване на екземплярите (Existence Dependent Entities). Екземплярите от зависимата същност може да съществуват само ако съществуват съответните екземпляри от независимата същност. Външните ключове в зависимата същност са неключови атрибути. Зависимите по отношение на съществуване на екземплярите в тях същност се изобразяват се чрез правоъгълник.


След като на диаграмата се разположат основните същности, се представят връзките между тях. За всяка връзка между същностите се определят: нейният тип, кардиналност и наименование. Типовете връзки са представени в таблица 13.1.

**Таблица 1**

**Типове връзки между същностите в ERwinDataModeler**

Тип връзка	Графично означение
Идентифицираща връзка (Identifying Relationship)	 <p>Родителската същност е независима, а породената същност е зависима по отношение на идентифициране. В зависимата същност външният ключ е част от първичния ключ.</p>
Неидентифицираща задължителна връзка (Non Identifying Mandatory Relationship)	 <p>Не може да съществува екземпляр от зависимата същност без да съществува съответен екземпляр от независимата същност, т.е. за всяка заявка трябва да се избере доставчик. Външният ключ на зависимата същност не може да приема неопределени стойности (в</p>

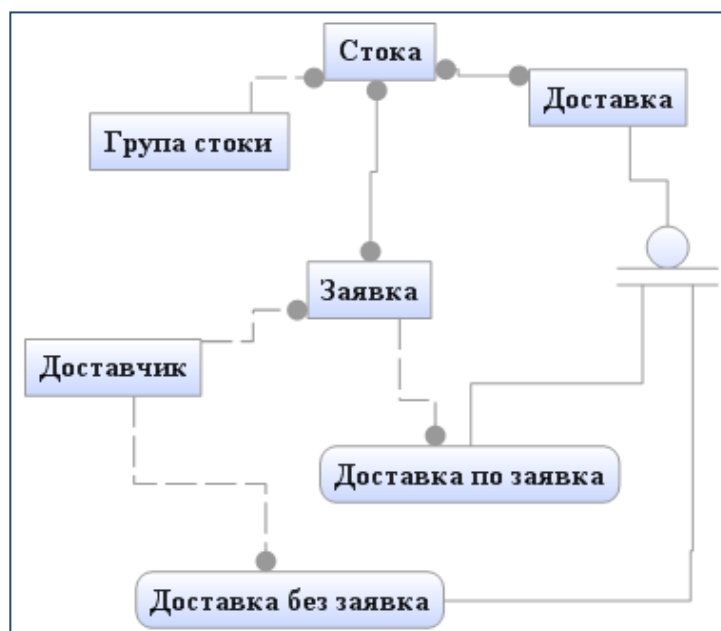
	<p>характеристиките на връзката се избира опция Nulls Not Allowed)</p>
<p>Неидентифицираща незадължителна връзка (Non Identifying Optional Relationship)</p>	<div style="text-align: center;"> <p>притежава / се намира в</p> </div> <p>Може да съществуват екземпляри от зависимата същност без да съществува съответен екземпляр от родителската същност, т.е. може да има доставчик, за който да не е посочена държава. Външният ключ на зависимата същност може да приема неопределени стойности (в характеристиките на връзката се избира опция Nulls Allowed)</p>
<p>Множествена връзка M:M (Many to Many Relationship)</p>	<div style="text-align: center;"> </div> <p>Такива връзки са неопределени и не се поддържат от релационните СУБД. Връзките M:M трябва задължително да се трансформират чрез добавяне на нова свързваща същност. Преобразуването се поддържа от ERwin чрез функцията “Resolve Many-to-Many Relationship”</p>

<p>Връзка Тип:подтип (Supertype:Subtype Relationship)</p>	 <p>Използва се за свързване на същности, които споделят общи атрибути, но имат и специфични за подтипа атрибути. Общите за същностите атрибути се отделят в същност супертип (Supertype), в примера – същност „Доставка“<sup>15</sup>. Специфичните за отделните същности атрибути се включват в подтиповете. В супертипа се добавя и атрибут дискриминатор (discriminator – в примера „Тип на доставка“), чиято стойност указва за кой подтип се отнася екземпляра. Връзките от тип Supertype:Subtype трябва да се преобразуват.</p>
---	--

Кардиналността и наименованието на връзката се задава чрез команда Properties от контекстното меню на избраната връзка.

Последователността на работа по създаване на логически модел ще илюстрираме с примерна опростена предметна област „Доставки на стоки“. Приемаме, че доставката на стоки може да се осъществява на база предварителни заявки, изпратени към доставчиците. В някои случаи обаче доставки могат да се получават и без предварителна заявка. Стоките се класифицират в групи. С една заявка и доставка се заявяват, съответно доставят, множество стоки. На фиг.13.6 е представена първоначална диаграма „Същност-Връзки“, в която се идентифицирани основните същности и връзки. В

<sup>15</sup> Показаната тук връзка отразява следното бизнес правило: “Една доставка може да бъде или по предварителна заявка, или без заявка“.



**Фиг.6. Диаграма „Същност – Връзки“ в ERwinDataModeler<sup>16</sup>**

### Разработка на модел с първични, външни и алтернативни ключове (Key-Based Model)

Следващият етап в моделирането на данните е създаването на модел с ключове. За целта първоначалната диаграма „Същност-Връзки“ се детайлизира като:

- се преобразуват множествени връзки и връзки тип:подтип;
- се добавят първичните ключове в независимите същности и в зависимите по отношение на съществуване на екземплярите;
- се добавят алтернативни ключове.

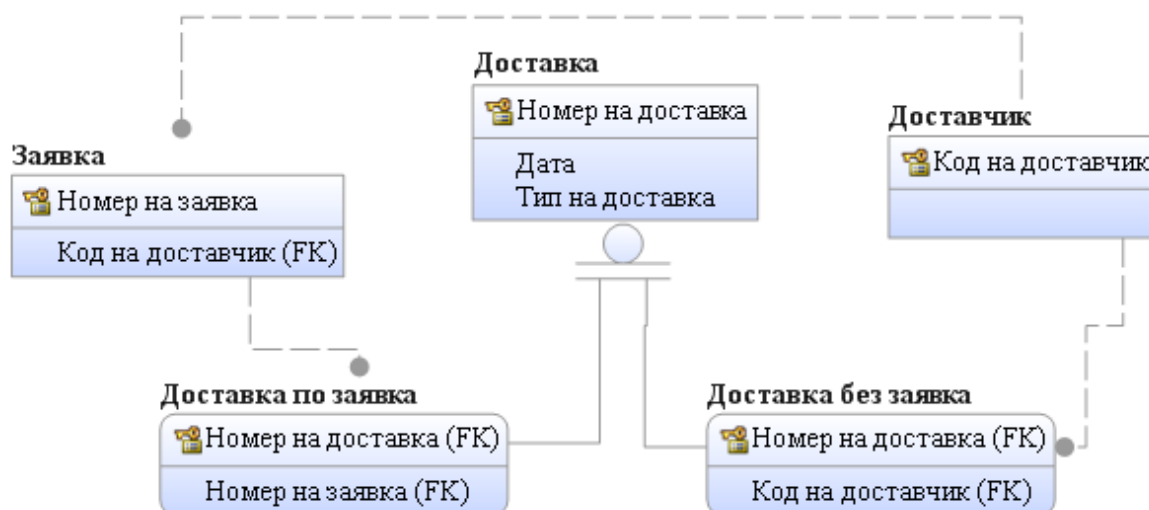
При трансформиране на множествени връзки проектантите могат да използват функцията „Resolve Many-to-Many Relationship” от меню Actions - >подменю Transformations. При преобразуване на М:М връзки се премахва множествената връзка и се добавя нова зависима същност, която се свързва с идентифициращи връзки със същностите, участващи в първоначалната връзка М:М.

<sup>16</sup> Примерите в темата са разработени в средата на безплатната версия Erwin Data Modeler r9.1 Community Edition

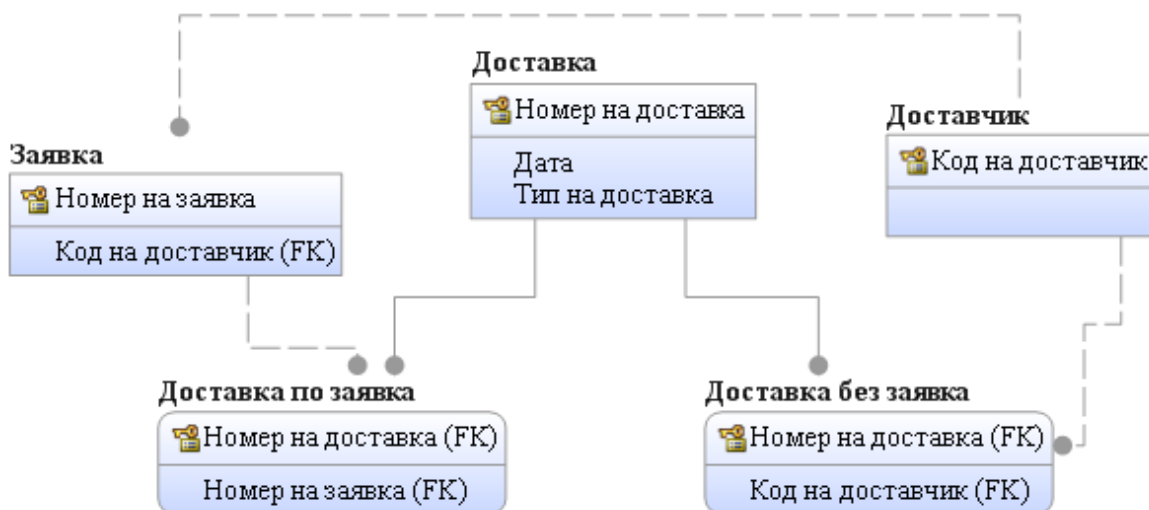


Преобразуването на връзки от вида тип:подтип се изпълняват от меню Actions ->подменю Transformations. Възможните трансформации на този тип връзки са:

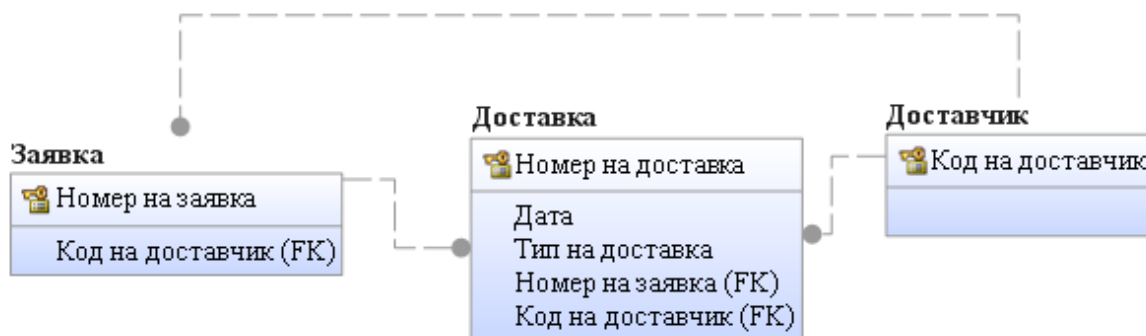
- Resolveto Supertype-Subtype Identity – връзката тип:подтип се замества от идентифициращи връзки между супертипа и подтиповете (вж.фиг.13.8)
- Supertype-SubtypeRollUp – подтиповете се обединяват в една същност със супертипа (вж.фиг.13.9)
- Supertype-Subtype Roll Down – подтиповете наследяват всички атрибути на супертипа, а супертипът се премахва (вж.фиг 10)



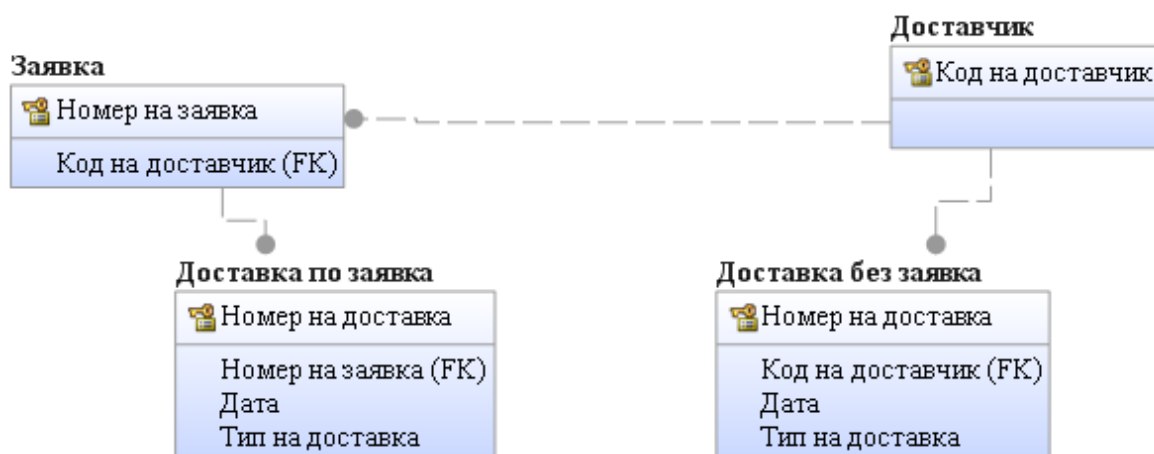
**Фиг. 13.7. Връзка Supertype:Subtype преди преобразуването**



**Фиг.8. Supertype Subtype Identity transformation**



**Фиг. 13.9. Supertype-Subtype Rollup transformation**



**Фиг. 10. Supertype-Subtype Rolldown transformation**

След като се трансформират връзките M:M и Supertype:Subtype се определят първичните ключове на включените в модела същности. Започва се от определяне на ПК на независимите същности. В зависимост от типа на връзките, в които участват независимите същности, ERwin мигрира първичните им ключове като външни ключове в зависимите същности по следния начин:

- Ако независимата същност участва в идентифицираща връзка, ПК мигрира като външен ключ (ВК), част от ПК на зависимата същност.
- Ако независимата същност участва в неидентифицираща връзка, ПК мигрира като ВК, който не е част от ПК на зависимата същност.

След като се дефинират ПК на независимите същности, се определят ПК на всички останали същности и алтернативните ключове.

### **Създаване на пълен атрибутивен модел (Fully Attributed Model)**



Erwin Data Modeler поддържа нормализация на данните в известна степен, но не и цялостен алгоритъм за нормализиране на данните. В зависимост от настройките за модела продуктът идентифицира или забранява дублирането на същности или атрибути. В Erwin Data Modeler не се позволява повторно мигриране на ВК в зависими същности с една и съща роля.

Въпреки, че не се поддържат атрибути от тип „списък“, атомарността на атрибутите, т.е. привеждането на отношенията в първа нормална форма, е грижа на създателите на модела.

Някои от грешките на втора и трета НФ могат да се избегнат, но пълната функционална зависимост между неключовите атрибути и ПК трябва да се съблюдава от проектантите. ERwin Data Modeler не позволява добавяне на атрибут в породената същност, идентичен с такъв в пораждащата.

Преобразуването чрез функция Resolve Many-to-Many Relationship на връзките от тип М:М е средство за поддържане на втора нормална форма, тъй като голяма част от ненормализирани в тази форма отношения възникват при връзки от този тип между същностите. Въпреки, че не е задължително да се преминава винаги през дефиниране на връзки М:М и последващо преобразуване, това дава възможност да се избегнат някои от грешките при нормализация на БД. Тази последователност на работа е особено полезна при проектантите с по-малък опит.