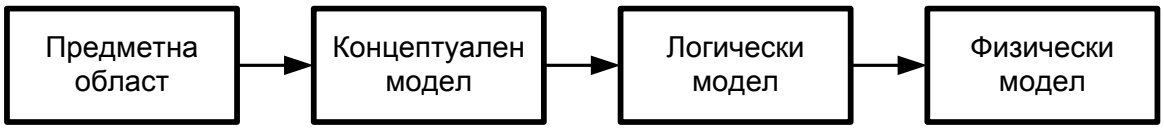
Проектиране на бази данни

1. Процесът на проектиране на БД

Процесът на проектиране на БД е свързан с:

- проектиране на ***концептуален модел*** – представяне на всички обекти и взаимовръзките между тях;
- проектиране на ***логически модел*** **–** представяне на обектите и взаимовръзките с помощта на моделите на данни: мрежов, йерархичен или релационен; представяне на графически модели, изхождайки от модела на данни;
- проектиране на ***физически модел*** – методите за съхраняване и достъп до данните, а също и методите за достъп до ОС;

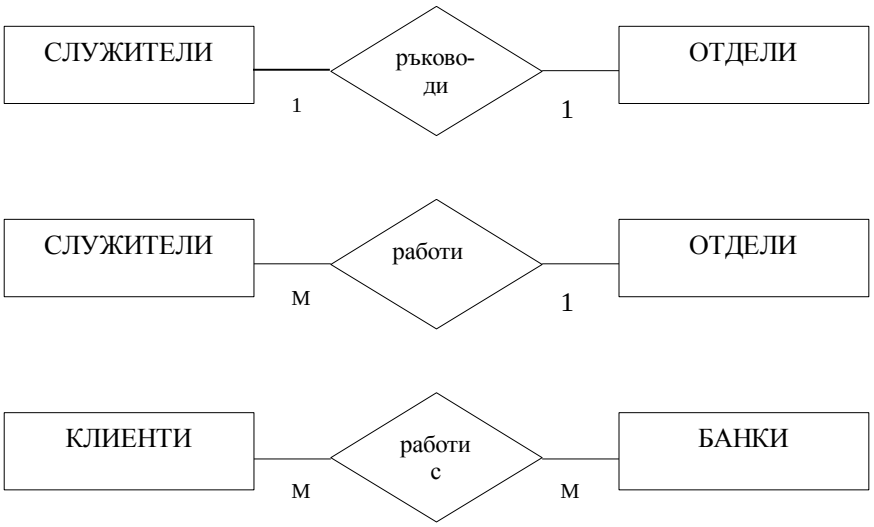


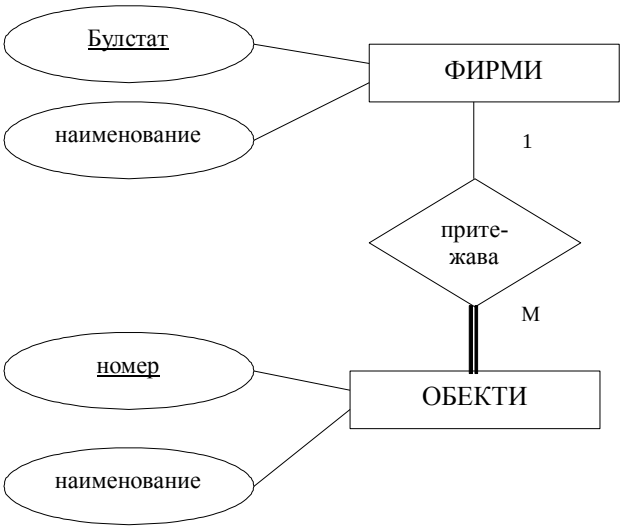
Фиг.1 Процес на проектиране на БД

Концептуална схема ще наричаме универсално представяне на структурата на данните в рамките на определена предметна област, независимо от крайната реализация на базата данни и апаратната платформа.

Един от най-популярните концептуални модели на данни е **моделът „Същност-Връзка”**, накратко наричан ER-модел (Entity-Relationship). Моделът е предложен от Питър Чен (Chen) през 1976 г. Моделирането на предметната област се базира на използване на графични диаграми, в които обектите и връзките между тях се представят с графични символи. Моделът ER е доразвит от множество разработчици и днес съществуват редица негови разновидности: модел на ORACLE; модел на Vantage Team Builder; модел на Баркер; метод IDEF1, разработен от Т. Ремей (T.Ramey), и неговата доразвита версия IDEF1X (използва се в ERwin, Design/IDEF и др.); метод IE (Information Engineering) и т. н. Разликите между отделните модели се състоят в графичното изобразяване на обектите, атрибутите и връзките, както и в степента на отразяването на особеностите на обектите, атрибутите и връзките.

В диаграма ER (същности – връзки) обектите се представят като именувани правоъгълници, връзките (отношенията) - чрез именован ромб, който е свързан с класовете обекти чрез линии, а атрибутите (свойствата) – чрез именувана елипса [4] (фиг. 2).





Фиг. 2 Примери за представяне на различни видове връзки между обекти чрез диаграма „същности-връзки”

Между две същности е възможно да съществуват следните видове връзки:

един към един (1:1) - На 1 екземпляр от едната същност съответства точно 1 екземпляр от другата същност и обратно. Примери: съпруг и съпруга (при моногамно семейство); клас и класен ръководител; директор и училище

един към много (1:M) На 1 екземпляр от едната същност съответстват много (повече от 1) екземпляри от другата същност. На 1 екземпляр от втората същност съответства 1 екземпляр от първата същност.

Примери: ученици и класни ръководители (един ученик има само един класен ръководител, но един класен ръководител ръководи много ученици); книги и издателства (определена книга се издава от точно едно издателство, но едно издателство издава много книги)

много към много (N:M) На 1 екземпляр от едната същност съответстват много (повече от 1) екземпляри от другата същност и обратно. Това е най-често срещания вид връзка. Примери: ученици – учители (1 ученик се обучава от няколко учители и 1 учител обучава няколко ученици); книги- автори; артисти – филми.

За преход от концептуален модел към логически модел се използват определени формални правила. Например, за преход от ER модел към релационна схема се използват следните правила:

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 таблицата.

Стъпка 2. Всеки атрибут става възможна колона със същото име; може да се избере и по-точен формат. Стълбовете, съответстващи на незадължителните атрибути, могат да съдържат неопределени стойности, а стълбовете, съответстващи на задължителните атрибути – не могат.

Стъпка 3. Компонентите на уникалния идентификатор на същността се превръщат в първичен ключ на таблицата. Ако има няколко възможни уникални идентификатора, избира се най-използвания.

Стъпка 4. Връзките много към много (и един към много) стават външни ключове, като се правят копия на уникалния идентификатор от страната „един” и съответните колони изграждат външния ключ. Незадължителните връзки съответстват на колони, допускащи неопределени стойности, а задължителните връзки – на колони, недопускащи неопределени стойности.

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

2. Етапи при проектиране на релационни БД

Проектирането на една релационна БД включва следните етапи:

1) Определяне на целите, на които ще служи БД.

2) Определяне на таблиците, които са нужни.

3) Определяне на полетата във всяка таблица.

4) Определяне на първичните и външните ключове.
5) Определяне на връзките между таблиците

2.1 Определяне на целите

Изграждането на една БД зависи от целите, които си поставяме.

Пример 1: Разработва се БД за стоките, които продава малка дистрибуторска фирма. Всички данни се съхраняват на един от компютрите в офиса. Достъпът до данните се осъществява от няколко служители чрез локална мрежа.

Тук акцентът ще се постави върху бързия достъп до данните и възможността за едновременна работа на няколко потребители. Надеждната защита на данните от неоторизиран достъп не е от съществено значение.

Пример 2: Разработва се БД за голяма фирма с офиси по целия свят.

В този случай трябва да се осигурят: надеждна защита на данните и различни права на достъп за различните категории служители; дублиране на критичната информация на няколко места в мрежата, бърз достъп до данните и др.

2.2 Определяне на таблиците

Обикновено в една таблица се записват данните, които се отнасят за един клас от обекти. Колко таблици ще има в една РБД, зависи от броя на описваните класове от обекти. Например, в БД за работата на библиотека ще има таблици за: книгите, авторите и читателите; в БД за дейността на едно училище ще има таблици за: учениците, класовете, учителите, предметите, оценките и др.

При неправилно определяне на таблиците е възможно възникването на следните проблеми, наречени ***аномалии***:

- ***Аномалия на излишеството*** – едни и същи данни се повтарят многократно, изразходвайки излишни ресурси за съхраняването си;
- ***Аномалия на обновяването*** – ако се промени някой от атрибутите, който е дублиран, то е необходимо да се обновяват всички записи, съдържащи този атрибут, за да се осигури непротиворечивост на данните;
- ***Аномалия на включването*** – нов запис може да бъде включен, само ако всички стойности на атрибутите му са дефинирани;
- ***Аномалия на изключването*** – изтриването на едни данни води до изтриването на други (напр., ако се изключат всички стоки на даден доставчик, данните за доставчика се изтриват);

За да изясним горните проблеми, ще разгледаме следния пример. В една училищна информационна система, за всеки ученик трябва да има информация в кой клас е и кой е класният му ръководител.

код ученик	Име ученик	Клас	Кл. рък
1	Асен Стоянов	10а	Руска Василева
2	Борис Велев	10а	Руска Василева
3	Васил Ганев	10а	Руска Василева
4	Ангел Балев	10б	Лиляна Минчева
5	Георги Славов	10б	Лиляна Минчева
6	Станчо Моллов	10б	Лиляна Минчева

Ако включим двете полета в записа на ученик, то идентификаторът на класа и името на учителя ще се среща толкова пъти в таблицата, колкото са учениците в класа (аномалия на излишеството). Ако учителят си смени името, то трябва да бъде променено във всички записи, които го съдържат, а ако пропуснем промяната в някой запис, ще се получи противоречие (аномалия на обновяването). Ако не знаем името на учителя, не можем да включим записите на учениците, на които е класен (аномалия на включването). Ако се изтрият записите на всички ученици, на които е класен ръководител, данните за учителя се загубват (аномалия на изключването).

За да се премахнат аномалиите, се използва т. нар. нормализация на релационната схема.

Нормализацията *е преобразуване на релационната схема, при което върху нея се налагат известни ограничения, водещи до отстраняване на някои нежелателни свойства.*

В теорията на релационните модели са известни 4 ***нормални форми***, които се различават според ограниченията върху типа на функционалните зависимости между атрибутите. За нормализация на релациите в четирите нормални форми се използват два основни метода: ***синтез*** и ***декомпозиция*** (разбиване на таблицата на няколко свързани таблици).

За да изясним метода на декомпозицията, ще разгледаме отново горния пример. За да се избегнат аномалиите, разгледаната таблица може да се разбие (декомпозира) на 2 таблици: таблица на учениците, съдържаща кода и името, и таблица на класовете, съдържаща класа и класния ръководител.

код ученик	Име ученик	Код клас
1	Асен Стоянов	100
2	Борис Велев	100
3	Васил Ганев	100
4	Ангел Балев	200
5	Георги Славов	200
6	Станчо Моллов	200

Код клас	Клас	Кл. рък
100	10а	Руска Василева
200	10б	Лиляна Минчева

2.3 Определяне на полетата във всяка таблица.

При определяне на полетата се задава:

1. Име на полето
2. Тип на данните
3.Размер на полето – посочва се минимално необходимия
4. Формат на данните.

Например, датата 24.05.2010 г. може да се представи в следните формати: 24.5.2010; 24 май 2010 и др.

5. Стойност по подразбиране (Default Value)
6. Условие за допустима стойност (validation Rule)

2.4 Определяне на ключовете във всяка таблица.

За да бъде осигурен достъпът до точно определен запис, всяка таблица трябва да разполага със специална колона (поле), наречена ***първичен ключ*** (или просто ключ).

Първичният ключ е специална колона (поле) или група колони (полета), която се използва за идентифициране на записите в таблицата. Стойностите в тази колона (или група колони) са **уникални** (не могат да се повтарят).

Както единните граждански номера са уникални и ни позволяват да отличим двама души безпогрешно, така и ключовете ни позволяват да разграничим даден запис от всеки друг в таблицата. Когато един ключ се състои от повече от едно полета, той се нарича ***съставен (комбиниран) ключ***.

Една релация може да има повече от един ключ. Тогава единият се избира за първичен, а останалите са **вторични** ключове.

Добрият кандидат за първичен ключ има няколко характеристики. Първо, той идентифицира еднозначно всеки ред. Второ, той никога не е празен или Null (стойност NULL означава неизвестна или неприложима стойност на атрибута в съответния ред) – той винаги съдържа стойност. Трето, той рядко се променя (в идеалния случай не се променя никога). Access използва полетата за първичен ключ, за да намира и съпоставя бързо данни от няколко таблици. Ако не се сещате какво поле или набор от полета биха били добър първичен ключ, помислете за използването на колона, която има данни от тип автономериране.

Релационната база данни съхранява информация в много таблици, свързани с ключови полета. За установяване връзка между две релационни таблици, освен първичен ключ се използва и ***външен ключ***.

Външният ключ на една релация (таблица) е такъв неин атрибут или списък от атрибути, който съответства на първичен ключ на друга релация.

Чрез свързване на първичния ключ от едната таблица с външния ключ на другата таблица между двете таблици се установява връзка. В общия случай външният ключ не е първичен ключ на релацията (той е също и първичен само във връзките 1:1), с други думи стойностите в полето, което е външен ключ, могат да се повтарят. За разлика от първичния ключ, външният може да приема и стойност NULL. Ограничението

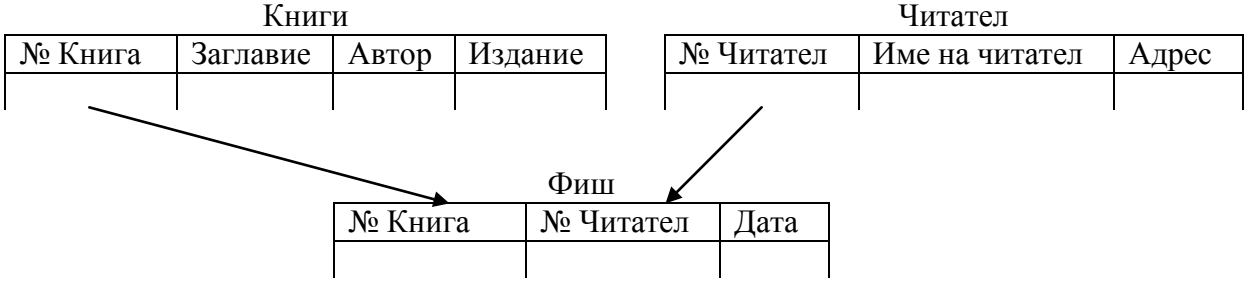
за външен ключ в релационна база данни е, че той може да приема или стойност NULL, или стойност равна на стойността в съответстващия му първичен ключ.

2.5 Определяне на връзките между таблиците

Между две релационни таблици е възможно да съществуват връзки от типа:

- ***един към един (1:1)*** – рядко срещана връзка; осъществява се чрез свързване на двете таблици по техните първични ключове;
- ***един към много (1:M)*** – осъществява се чрез свързване на двете таблици по първичния ключ на страната "един" и външния ключ на страната "много";
- ***много към много (N:M)*** – посредник между двете таблици е допълнителна свързваща таблица, съдържаща като външни ключове първичните ключове на двете таблици. По този начин връзката (N:M) се представя като две връзки "един към много".

Пример: В Базата данни "Библиотека" между читатели и книги съществува връзка много към много (един читател може да заема много книги, но и една книга може да бъде заемана от много читатели). Връзката между читатели и книги се осъществява чрез свързваща таблица "Фиш", която съдържа като външни ключове "№ книга" – първичният ключ от таблицата "Книги" и "№ читател" – първичният ключ от таблицата "Читатели". В таблицата "Фиш" може да се създаде съставен ключ от полетата "№ книга" и "№ читател", ако се предположи, че един читател не чете повторно дадена книга. Ако обаче има читатели, които обичат да препрочитат любими книги, тогава съставният ключ трябва да включва и полето „дата". Вместо със съставни ключове, често е по-удобно е да се работи със създаден от програмиста ключ от тип AutoNumber. В случая, в таблицата Фиш е подходящо да се създаде поле „№ на заемане".



3. Правила за организиране на таблиците

За разполагане на данните в таблиците не съществуват строги правила. Въпреки това, могат да бъдат посочени някои най-общи правила за ефективни БД:

1. Определяне на предназначението на всяка таблица и проверка за това всички данни в определена таблица да съответстват на предназначението й.

2. Ако част от полетата в много записи в една таблица остават празни, тя трябва да се разбие на две свързани.

3. Ако определена информация се повтаря в много записи, необходимо е тя да се премести в отделна таблица и да се установят отношения между таблиците.

4. Наличието на повтарящи се полета, говори за необходимост от създаване на подчинена таблица. Ако например, в таблицата на доставчиците присъстват полета стока1, стока2, стока3, стока4 и т. н., това означава, че информацията за стоките трябва да се премести в подчинена таблица и да се свърже с родителската.

Име доставчик	Стока 1	Стока 2	Стока 3	Стока 4

Ученик	Хоби 1	Хоби 2	Хоби 3	Хоби 4

5. За намаляване на обема на данните и повишаване на точността на входната информация е добре да се използват таблици за преглед, наричани още справочни таблици (lookup table). Например, вместо да се въвежда информация за населеното място, общината и областта, може да се създаде таблица, съдържаща пощенския код и тези данни. Съответното населено място ще се избира от справочната таблица.

6. Не следва да се съхранява в таблица информацията, която може да се изчисли въз основа на данните от други таблици.

4. Съображения за отклонения от правилата

Както беше вече посочено, горните правила не са строги. Една от най-честите причини за отклоняване от тях е увеличаване на производителността.

Например, ако се налага изчисляване на определена характеристика за десетки хиляди записи, може да се окаже по-приемливо да се въведе допълнително поле, съдържащо резултата. Това поле трябва да се обновява всеки път, когато се променя определен запис, като за това поема грижата програмистът.

Пример: датата на раждане и полът може да се изчислят въз основа на ЕГН. Първите 6 цифри показват годината, месеца и датата на раждане, а предпоследната цифра – пола (ако е четна – мъжки, ако е нечетна - женски). Ако се съхранява информация за населението на един голям град, може да е по-удачно да съхраняваме данните за датата на раждане и пола в отделни полета, вместо да ги изчисляваме всеки път при извеждане на справки.

Друга причина за отклонение от правилата е стремежът да не се отварят едновременно голям брой файлове. Тъй като всяка отворена таблица задейства указател на файл и заема място в паметта, значителния брой отворени таблици може да забави работата на програмата.

5. Реализация на приложение за релационна БД.

Реализацията на един проект за БД преминава през следните стъпки:

1. Създаване на БД

а) създаване на таблиците – полета, тип на данните, правила за достоверност на данните;
б) създаване на индекси и ключове;
в) създаване на отношенията (връзките) между таблиците;
г) създаване на заявки;

2. Попълване и редактиране на БД

3. Изменение на структурата на БД

а) промяна на таблици – добавяне, премахване и промяна на полета;
б) добавяне и премахване на таблици;
в) добавяне, премахване и промяна на индекси;
г) отстраняване и добавяне на отношения;
д) добавяне, премахване и промяна на заявки;

4. Изграждане и програмиране на потребителския интерфейс – формуляри, менюта, ленти с инструменти и др.

5. Създаване и редактиране на отчети

6. Конфигуриране на системата за безопасност - задаване на права на достъп

7. Тестване

8. Комплектоване на приложението за разпространение.

Процесът на създаване на едно приложение за БД е итеративен (повтарящ се). Някои от тези стъпки се изпълняват многократно и не е задължително да следват точно тази или някаква друга предварително определена последователност.

Задачи за самостоятелна работа:

1. Да се проектира БД "Видеотека" за обмена на касети във видеотека. Да се съхраняват данни за касетите, клиентите и видеообмена.
2. Да се проектира БД "Болница". Да се съхраняват данни за лекарите, пациентите, заболяванията и престоя на пациентите.
3. Да се проектира БД "Склад". Да се съхраняват данни за стоките, доставчиците и клиентите.
4. Да се проектира БД "Ресторант". Да се съхраняват данни за продуктите, рецептите и менюто.
5. Да се проектира БД "Дневник на класа". Да се съхраняват данни за учениците, предметите, оценките и отсъствията.
6. Да се проектира БД "Мебелна фирма". Да се съхраняват данни за материалите и готовата продукция.