РЕЛАЦИОННИ БАЗИ ОТ ДАННИ. КОНЦЕПЦИЯ ЗА БАЗИ ОТ ЗАННИ В ACCESS

доц. д-р Е. Ангелова

доц. д-р Т. Глушкова

glushkova@uni-plovdiv.bg

Катедра „Компютърни технологии“

Съдържание

Релационни бази от данни

- Въведение
- Системи за управление на БД
- Видове БД
- Релационни БД. Свойства.

Концепция за БД с MS Access

- Проектиране на БД.
- Първични ключове
- Кардиналност
- Нормализация

Въведение

Данни и управление на файлове

- Още от своето създаване компютърните системи са използвали различни механизми за обработка на данни и извличане на информация.
- Те са записвали данните във външни файлове и затова са наречени системи за работа с файлове. Макар, че тези системи имат множество предимства пред ръчната обработка, те имат и доста ограничения:
  - Данните не са свързани помежду си;
  - Данните често се дублират;
  - Програмите, които ги обработват зависят от формата на файла.

Системи за работа с бази от данни

- Целта на технологията на БД е да преодолее съществуващите недостатъци на системите за работа с файлове.
- Системата за управление на базите от данни (СУБД) е програма, която позволява съхранените данни да бъдат интегрирани, намалява дублирането на информация, премахва зависимостта на програмите за обработка от формата на данните във файла и предоставя възможности за обработка на сложни обекти.

Системи за управление на бази от данни

- На практика:
  - системите да работа с файлове осигуряват достъп до файловете с данни;
  - програмите за работа с бази от данни реализират връзка със СУБД, която от своя страна осигурява достъп до съхранените данни.
- Разликата е съществена, защото СУБД се явява междинен слой, което освобождава програмиста от отговорността за физическото съхранение на данните и той трябва да насочи усилията си към контрол на продуктивността на програмата за работа с базите данни.

База от данни

<u>База Данни</u> - **в най-общия смисъл една *база данни (БД)* е компютърна система за регистриране и поддържане на логически свързани помежду си структури от данни**.

- Самата БД може да бъде разглеждана като място за съхраняване на електронни структури от данни.
- Една база данни освен оперативните данни, съдържа и описание на собствената си структура (метаданни, каталог на данните)

Въведение в системите за управление на бази данни

- БД може да се разглежда като колекция от интегрирани записи, т.е. освен данните и мета-данните, тя включва и описание на връзките между отделните записи.
- Тези връзки се съхраняват и се използват по време на работата с БД.
- Характерно за базите данни е, че данните са независими от софтуера. Това ги прави универсални за използване както от различни програми, така и в различни периоди от време.

Въведение в системите за управление на бази данни

- Потребителите на такива системи обикновено разполагат с набор от средства, които им дават възможност да извършват следните операции с данните:
  - добавяне на нови структури в БД
  - добавяне на нови данни в съществуващи структури
  - извличане на данни от съществуващи структури
  - промяна на данни в съществуващи структури
  - отстраняване на структури с вече ненужна информация.

Въведение в системите за управление на бази данни

Съществуващите СУБД работят върху машини от различни класове - от малки персонални компютри до големи мейнфреймове (mainframes).

Възможностите, които предлага всяка отделна система са различни и се лимитират от използваната хардуерна конфигурация.

Въведение в системите за управление на бази данни

Съществуват два режима на работа:

- ✓ **Еднопотребителски** –когато във всеки момент с БД работи само един потребител (***single-user***).
- ✓ **Многопотребителски** –във всеки момент със системата за управление на БД работят повече от един потребител като се конкурират за ресурсите на машината. БД за големи машини са предимно многопотребителски (***multi-user***).

СУБД

- Приложният софтуер, който осигурява възможност за работа с бази от данни, се нарича система за управление на бази от данни (СУБД).
- Тя включва три основни компонента:
  - средства за разработване на приложения;
  - потребителски интерфейс;
  - ядро, което извършва операциите по търсене, сортиране и актуализиране на данните в базата от данни.

СУБД

- Заявките на потребителите за достъп до съдържанието на БД се обработват от СУБД.
- Друга важна функция, която изпълнява СУБД, е защита (екраниране) на потребителите от хардуерните детайли (аналогично на езиците за програмиране),т.е. СУБД предоставя на потребителите поглед върху БД, който е високо над хардуерния слой на системата.
- Това се реализира предимно чрез така наречените потребителски интерфейси.

Свойства на СУБД

- ***Цялостност*** - БД се разглежда като унификация на отделни физически структури, при което тя съдържа излишество от информация, което може да бъде частично елиминирано.

Свойства на СУБД

***Пример за цялостност**:*

- *БД съдържа две структури SEMINAR (име,адрес,факултетен номер,... на студентите в една семинарна група) и KURS (имената на студентите, посещаващи определен курс).*
- *Ако искаме да узнаем факултетния номер на даден участник в курса не е необходимо те да бъдат записани повторно в структурата KURS.*
- *При необходимост факултетния номер ще бъде получен посредством обръщение към файла SEMINAR.*

Свойства на СУБД

- ***Разпределеност*** - отделна единица информация в БД може да бъде разпределена между различни потребители без тя да губи своята цялостност, т.е. всеки може да има достъп до едни и същи данни.
- Исканията за достъп на различните потребители могат да постъпят едновременно в системата (concurrent access).

Свойства на СУБД

Пример за разпределеност:

- *Факултетният номер може едновременно да се иска от личен състав и учебен отдел .*
- *Те могат да използват тази информация за различни цели.*

Следствие от цялостността на БД

- Всеки потребител работи само с определено подмножество на цялата БД.
- Отделните подмножества могат да се препокриват по различни начини. С други думи - една БД се възприема по различен начин от различните потребители.
- Когато двама потребители разделят едно и също подмножество на БД, в зависимост от нивото на детайлизация, ***техните гледни точки за това подмножество могат да се различават значително.***

Данни. Видове

1.Оперативни данни:

- данните, които съдържа БД се наричат *оперативни* или *експлоатационни*.
- В този контекст една БД може да се дефинира като множество от съхранени оперативни данни, които използват (обработват) от приложните системи на различни потребители.
- Голямото многообразие на потребители (от еднолични малки фирми до големи корпорации) обуславя и съществуването на БД с различни характеристики (от малки частни БД до големи разпределени БД).

Данни. Видове

2. Входни данни:

- първична информация, която се въвежда в системата (обикновено еднократно с помощта на клавиатура, скенер, светлинна писалка и т.н.).
- Въвеждането на такива данни може да предизвика необходимост от промяна на някои оперативни данни и самите те могат да станат част от БД (т.е. оперативни);

Данни. Видове

3. Изходни данни:

- резултати от обработката на заявките, които се извеждат на печат или екран.
- Те се получават като резултат от определени операции с оперативните данни
- Те по принцип не могат да станат част от самата БД.

Архитектура на системите за управление на бази данни

Архитектурата на СУБД е разделена на три основни нива:

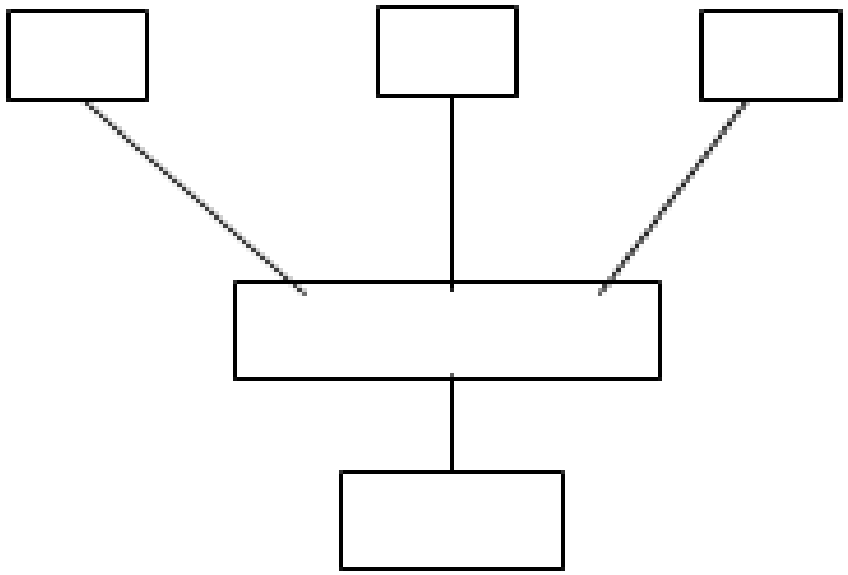
- ***вътрешно ниво*** *(internal level)* - най-близо до физическата памет, т.е. то решава проблемите свързани с начините за физическото съхраняване на данните;
- ***външно ниво*** *(external level)* - най-близо до потребителите, т.е. разглежда възможните начини за избирателното представяне на данните (views) за различните потребители;
- ***концептуално ниво*** *(conceptual level)* - това е нивото за връзка между първите две нива.

Архитектура на системите за управление на бази данни

External level
(individual user views)

Conceptual level
(community user view)

Internal level
(storage view)



Архитектура на СУБД

- ***Външно ниво:***
  - занимава се предимно с индивидуалните гледни точки на потребителите.
- ***Вътрешно ниво:***
  - *представя физическото разположение на данните във вторичната памет.*

Архитектура на СУБД

- ***Концептуално ниво*** - различните гледни точки трябва да се трансформират в единна логическа схема, която е характерна за конкретната СУБД.
- С други думи, ние различаваме много различни външни гледни точки, всяка от които съдържа някакво абстрактно представяне на част от цялата БД, и същевременно съществува само една концептуална гледна точка, която съдържа абстрактно представяне на БД като цяло (да припомним, че повечето потребители не се интересуват от цялата БД,а само от една част от нея).

Основни видове бази данни

- ***релационни –*** данните се представят като таблици
- ***йерархични -*** данните се представят като множества от дървовидни структури,а предоставените оператори могат да манипулират такива структури
- ***инвертирани списъци*** - в отделни таблици се съхранява информация, подпомагаща достъпа до оперативните данни
- ***мрежови структури*** - данните се представят в мрежови структури
- ***дедуктивни*** - релационна БД с логически компонент
- ***обектно-ориентирани*** - поддържат обектно-ориентирани структури и механизми.

През последните години повечето разработени БД са релационни.

Релационни бази от данни

Една система за управление на БД е релационна, когато:

- Данните се съхраняват в двумерни таблици, наречени релации;
- Всеки запис в релацията е различна от останалите.
- Релациите са свързани помежду си в обща структура.
- Операторите, с които потребителят разполага, генерират нови релационни обекти (релации и домейни), променят и изтриват съществуващите релационни обекти и обработват данните по подходящ начин.

Системите за релационни бази от данни използват математическата теория на множествата, за да организират ефективно данните.

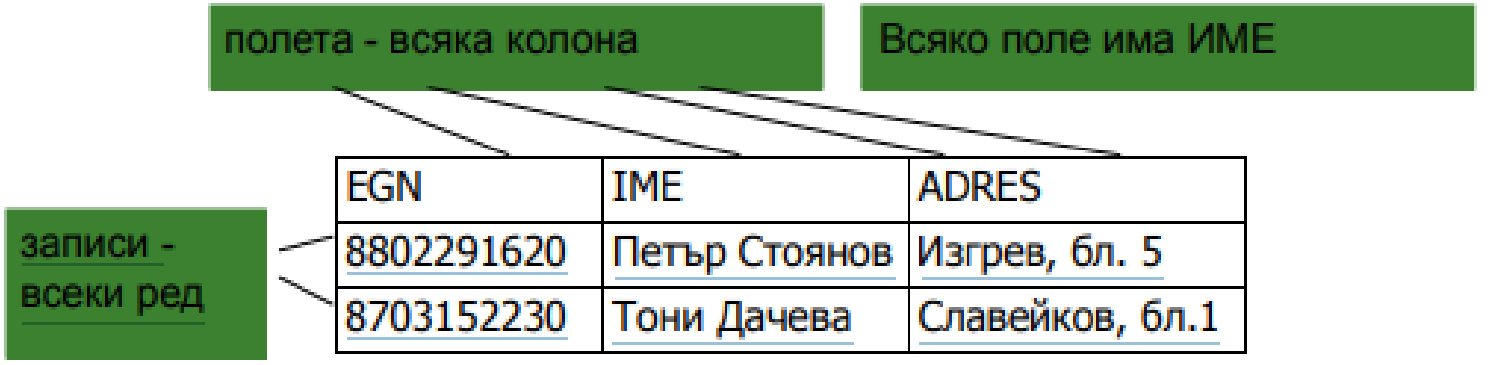
Релационни БД. Определения.

Релационна БД (*неформално определение):* БД, която се възприема от потребителите като множество от таблици (и нищо друго освен таблици).

- ✓ ***релация (relation)*** - съответства на познатото ни понятие *таблица*
- ✓ ***подредено множество (запис)*** - съответства на ред
- ✓ ***атрибут (поле)*** - колона
- ✓ ***кардиналност*** – брой редове
- ✓ ***степен –*** брой колони.

Релационни БД. Определения.

- ✓ Данните се помнят в таблици с уникално име



- ✓ Всяка колона от таблицата се нарича **поле**. То съдържа данни от един и същи тип - низове,числа, дати и др.
- ✓ Всеки ред от таблицата се нарича **запис**.

Примери за таблици

- **Таблици, задаващи категория от обекти:**
  - Таблицата “Клиент”. Отличителните черти, характеризиращи един клиент са например неговото ЕГН, име, адрес, телефон.
  - Таблица «Книги» - каталожен номер, заглавие, автор, количество, цена

- **Таблица, задаваща категория от действия**
  - Таблица “Поръчка” -номер на поръчката; каталожен номер на поръчаната книга; ЕГН на клиента, направил поръчката; дата на поръчката;

Релационни БД. Определения.

- ***първичен ключ (primary key)*** - уникален идентификатор на таблица, т.е. поле (или комбинация от полета), който (която) притежава следното свойство: във всеки момент няма повече от един ред с една и съща стойност в това поле (комбинация)
- ***област (domain)*** - множество от стойности, които могат да бъдат присвоявани на актуалните стойности на атрибутите

Първичен ключ

✓ ***Прост първичен ключ – състои се от едно поле***

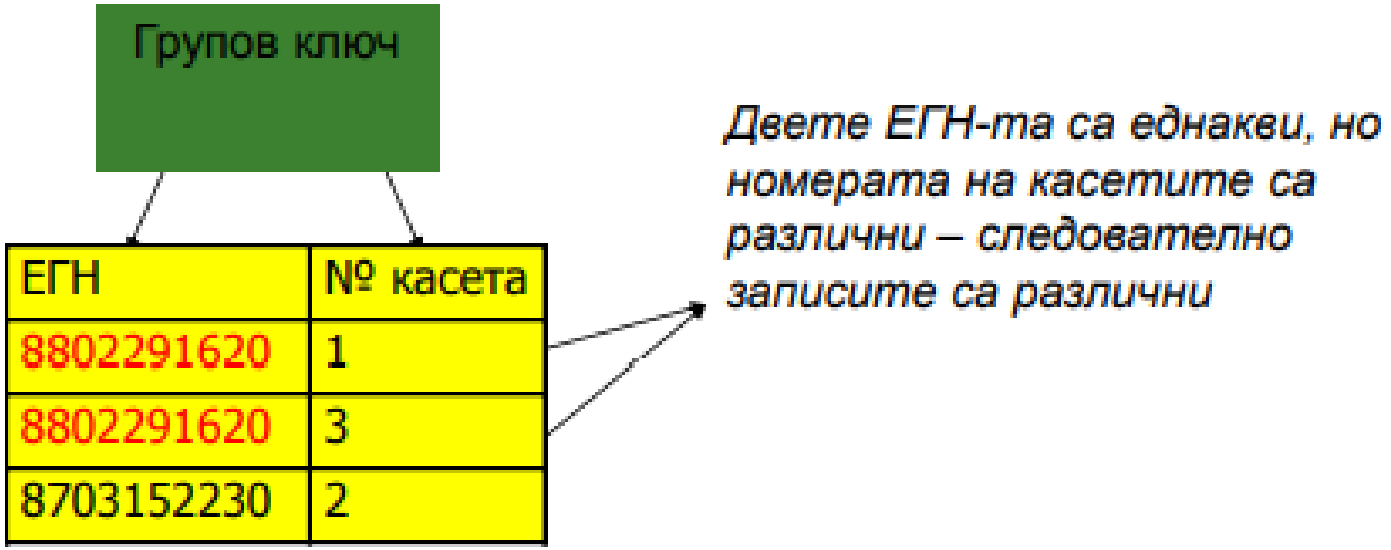
В него не може да има 2 еднакви стойности. Всеки запис е уникален. Всяка таблица трябва да има първичен ключ.

ЕГН	Име	Адрес
8802291620	Петър Стоянов	Изгрев, бл. 5
8703152230	Тони Дачева	Славейков, бл.1

Първичен ключ

- ✓ ***Съставен първичен ключ – състои се от две или повече полета***

В него не може да има 2 еднакви стойности. Всеки запис е уникален.



Релационни БД. Основни свойства

- **всички стойности са атомарни** – таблиците не съдържат *повтарящи се групи* (напр. {Т1,Т2}).
- **цялата информация, съдържаща се в БД е представена с явни стойности** - Не съществуват указатели, свързващи отделните таблици
- когато казваме, че една система е ***релационна***, мислим че тя поддържа *релации* на външно и концептуално ниво
- на ниско ниво (вътрешно) системата е свободна да използва произволни структури, стига да е в състояние на горните две нива да ги представи като релации
- една релационна БД се възприема от потребителите като множество от таблици. Тя не е БД, в която данните се съхраняват като таблици

Свойства на релациите

Релациите имат някои важни свойства. В една дадена релация:

- няма дублирани записи;
- записите са неподредени (top to bottom);
- атрибутите са неподредени (left to right);
- всички стойности на атрибутите са atomic.

Свойства на релациите

- ***Няма дублирани записи*** **-** това следва от факта, че тялото на релацията е множество (в мат. смисъл), т.е. по дефиниция не включва повтарящи се елементи.
- *Важно следствие* - винаги съществува първичен ключ. Понеже записите са уникални, следва че поне комбинацията на всички атрибути на релацията притежава качеството уникалност.
- На практика обаче обикновено не е необходимо да се включват всичките атрибути - обикновено по малки комбинации са от значение.

Свойства на релациите

- ***Записите са неподредени:***
  - ***Т***ова свойство също следва от факта, че тялото е математическо множество.
  - Множествата не са подредени.
  - Записите могат да бъдат разглеждани и в обратен ред - това не променя релацията.

Свойства на релациите

- ***Атрибутите са неподредени:***
- следва от факта, че заглавната част на релацията също е множество.
- Атрибутите с идентифицират посредством имената си и никога посредством позицията си.

Свойства на релациите

- ***Всички стойности на атрибутите са atomic:***
- Всички прости атрибути са неделими (atomic). Това е следствие от факта, че прилежащите области са неделими, т.е. съдържат само неделими елементи.
- Това свойство можем да изразим и по друг начин - на всяка позиция в таблицата винаги може да стои точно една стойност, а не множество от стойности.
- Една релация, удовлетворяваща това условие се нарича *нормализирана*.

Още за правилата за цялостност

Да обърнем внимание на това, че:

- *във всеки момент, всяка БД съдържа определена конфигурация от стойности на данните и тази конфигурация отразява (е модел) на една реална ситуация*
- *определени конфигурации нямат смисъл, ако те не представляват някакво възможно състояние на реалния свят*

Следователно дефиницията на БД трябва да бъде разширена - тя трябва да включва *правила за цялостност*. Тяхната цел е да информират СУБД за някои ограничения в реалния свят (напр., теглото не може да бъде отрицателно число). Така могат да се избегнат някои конфигурации, които нямат смисъл.

БД са субект на многобройни правила за цялостност.

Още за правилата за цялостност

- Някои правила за цялостност са специфични за точно определена БД.
- Освен специфични правила за цялостност релационният модел включва две общовалидни, които са изпълнени за всяка релационна БД:
  - ***концепцията за първични ключове***
  - ***концепцията за външни ключове.***
- Правилата са общи в смисъл, че те са приложими за всички БД, които претендират, че се съобразяват с изискванията на релационния модел.
- Наистина, всяка отделна БД може да има свои собствени специфични правила, които допълват двете основни правила.

Ключове кандидати и алтернативни ключове

- ***Ключове-кандидати***
  - първичните ключове са специален случай на по-общата конструкция, наречена ключове-кандидати (candidate keys).
  - Един *ключ-кандидат* е един *уникален идентификатор*.
  - По дефиниция всяка една релация има най-малко един ключ-кандидат (на практика,обаче, повечето релации имат точно един, но има такива с по два и повече ключа).
  - За дадена релация ние избираме един от ключовете-кандидати за ***първичен*** и всички останали стават (ако съществуват) ***алтернативни*** ключове.

Ключове кандидати и алтернативни ключове

Забележка:

- Всяка релация има най-малко един ключ-кандидат, тъй като релациите не съдържат дублирани n-торки.
- Понеже n-торките са уникални, следва че най-малко комбинацията от всички атрибути на релацията притежава свойството *уникалност*.

Ключове кандидати и алтернативни ключове

- **Пример:** релацията ELEMENTS (таблицата на химическите елементи). Всеки хим. елемент има уникален:
  - име
  - символ
  - атомарен номер.
- Ние дефинираме ключовете-кандидати като множества от атрибути. Един ключ-кандидат, който се състои от повече от един атрибут се нарича ***съставен***. С един атрибут – ***прост*.**

Още за първичните ключове

Както видяхме, възможно е една базова релация да има повече от един ключ-кандидат. В такъв случай релационният модел изисква един от тях да бъде обявен (избран) за *първичен ключ* (ПК). Останалите се наричат *алтернативни ключове* (АК).

- *Например* за релацията ELEMENTS избираме за ПК - SYMBOL, а за АК - NAME, NUMBER. Кой ключ ще бъде избран зависи от прагматични причини.

Първичният ключ идентифицира еднозначно записите в релациите.

Външни ключове (Foreign keys)

Външен ключ - нека R е базова релация. Един външен ключ (ВК) в R е едно подмножество на множеството на атрибутите така че:

- съществува една базова релация R1 с първичен ключ
- по всяко време стойностите на ВК в R са равни на стойности на ПК в някои запис на R1.

Пример: да разгледаме следната отдел-служител БД

- DEPT (DEPT_ID, DNAME, MGR_ID, BUDGET)
- EMP (EMP_ID, ENAME, DEPT_ID, SALARY)

Атрибутът EMP.DEPT_ID е външен ключ в релацията EMP (равен на първичния ключ DEPT.DEPT_ID на релацията DEPT), обаче той не е част от първичния ключ EMP.EMP_ID. Аналогично, атрибутът DEPT.MGR_ID е външен ключ в релацията DEPT.

Свойства на външните ключове

По тази дефиниция можем да характеризираме външните ключове както следва:

- ВК са също множество от атрибути
- Всяка стойност на ВК трябва да се появява като стойност на съответния първичен ключ. Обратното не се изисква, т.е. ПК може да има стойност, която не се появява във ВК
- един ВК ще бъде съставен, само ако кореспондиращият ПК е също съставен. Ако ВК е прост, ПК - също е прост
- един ВК представлява връзка (референция) към реда, съдържащ съответстващата стойност на ПК.

Свойства на външните ключове

- ***Референциална цялостност*** - проблемът за осигуряване, че БД не съдържа невалидни стойности на ВК.
- ***Референциално ограничение*** - стойностите на един ВК трябва да съответстват на стойностите на съответния ПК.
- *Релацията е:*
  - ***реферираща*** – ако съдържа ВК.
  - ***референцирана (целева)*** - релацията на ПК.
- ***Правило за референциалната цялостност*** - БД не трябва да съдържа стойности на ВК, които нямат съответствие, т.е. няма съществуваща стойност за значещия ПК в целевата релация.

Възможностите

1. Системата да отхвърли всяка операция, която води до некоректно състояние, ако бъде изпълнена;
2. Системата извършва операцията, а след това извършва допълнителни компенсиращи операции.

Възможностите

За всеки ВК съществуват два основни въпроса, на които трябва да се отговори:

1. *Какво ще се случи при опит да се изтрие целевата на ВК референция?* Напр., опит да се изтрие идентификатор на продукт, който участва в поръчките. Съществуват два подхода за решение:

- RESTRICTED - операцията изтриване е ограничена само за случая, когато продуктът не участва в заявките
- CASCADES - операцията се каскадира (разширява) и изтрива също така и заявките, в която този продукт е включен.

2. Какво ще стане при опит да се промени един ПК, който е целева референция на ВК? Напр., опит да се промени идентификатор на продукт, който участва в поръчки. Същите два подхода:

- RESTRICTED
- CASCADES.

Нулеви стойности

- ***Нулева стойност*** - липсва информация.
- В реалния свят има много такива ситуации.
- В релационния модел е приет един специален маркер за означаване на липсваща информация - **NULL.**
- Един атрибут може да съдържа или може да му бъде забранено да съдържа нулеви стойности.

Нулеви стойности

Не се разрешава на нито един от компонентите на ПК да има нулева стойности, защото*:*

- базовите релации са автономни именувани релации. Те кореспондират с обекти от реалния свят. *Напр. базовата релация Доставчици кореспондира на група доставчици в реалния свят*
- по дефиниция обектите (същностите) в реалния свят са различими - т.е. те трябва да бъдат идентифицирани по някакъв начин
- първичните ключове изпълняват функцията на *уникална идентификация* в релационния модел

Извод **-** в една релационна БД ние не трябва да записваме информация, която не можем да идентифицираме.

Свързване на таблиците в обща БД

- За нормалното функциониране на една база от данни е необходимо да се определят правилно и да се изградят връзките между съществуващите таблици
- За да създадем релация между 2 таблици - те трябва да имат **общо/еднакво/** поле!

Кардиналност на връзките

- В теорията за релационните БД можем да разгледаме три типа връзки със следните кардиналности:
  - *Едно към едно*
  - *Едно към много*
  - *Много към много*

Едно към едно

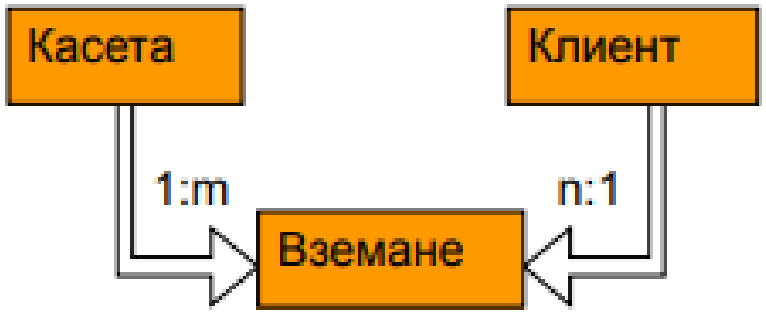
- Този тип връзка се използва, когато **един** обект от едната таблица се свързва с **един** обект от другата таблица.
- Например:
  - Един клас – един класен ръководител
  - Една фирма- един началник
  - Едно училище- един директор

Едно към много

- Този тип връзка се използва, когато **един** обект от едната таблица се свързва с **много** обекти от другата таблица.
- Например:
  - Един клас – много ученици
  - Една фирма- много работници
  - Едно училище- много учители
  - Един университет – много специалности

Много към много

- Този тип връзка се използва, когато **един** обект от едната таблица се свързва с **много** обекти от другата таблица и **един** обект от втората таблица се свързва с **много** обекти от първата.
- Например: Една видеокасета се взема от много клиенти, а един клиент заема много видеокасети



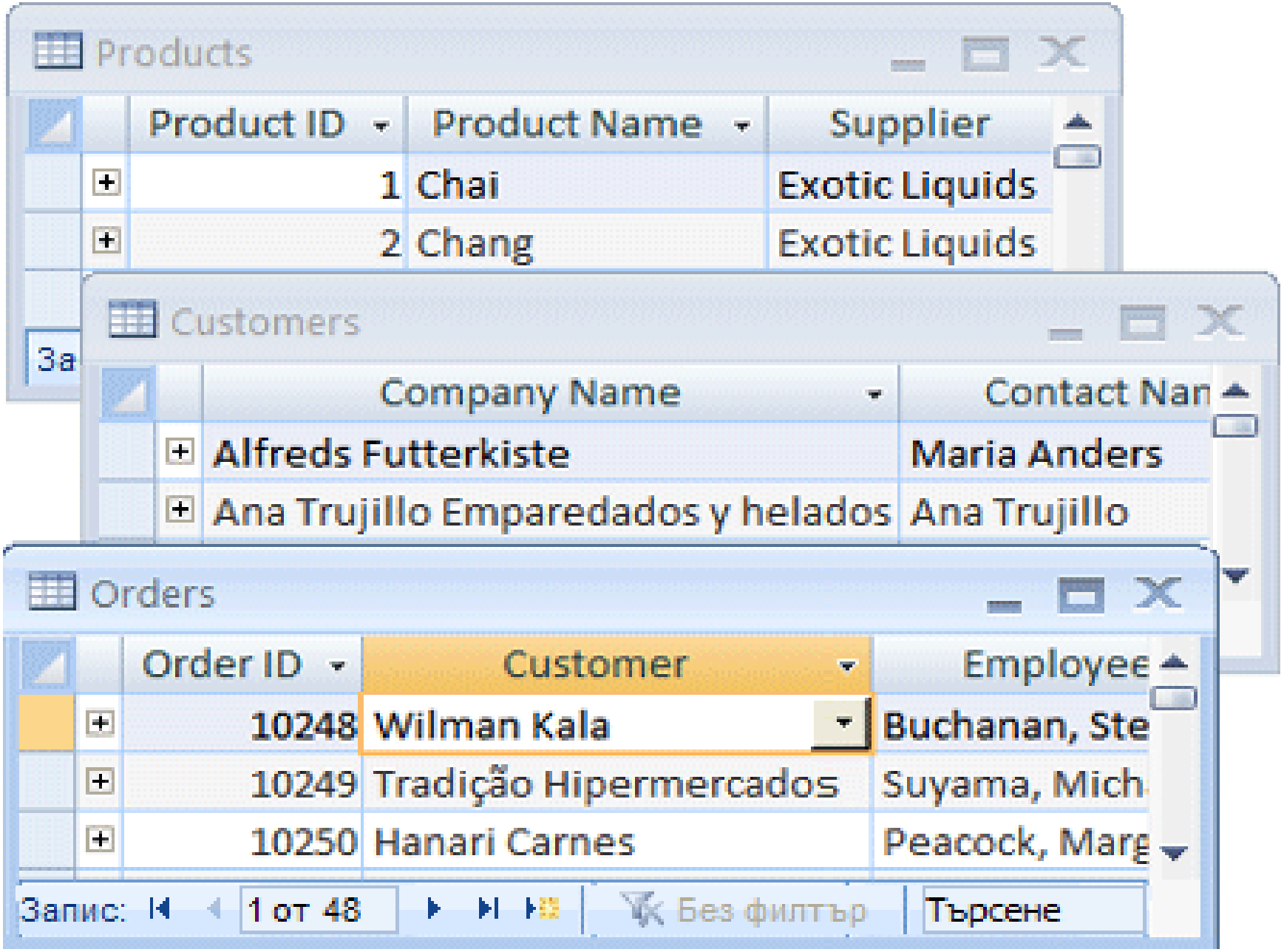
Връзки от тип много към много не се поддържат от СУБД.
Те трябва да се разбият на 2 връзки 1:n чрез добавяне на нова таблица

Много към много

MS Access

- Microsoft Office Access е СУБД от релационен тип.
- Информацията е организирана в двумерни **таблици,** свързани помежду си в цялостна структура.
- Една проста база данни може да съдържа само една таблица и тогава обработката на данни е сходна на Excel с изискването за уникалност на всеки запис.
- В повечето бази данни са необходими повече от една таблици.

Например може да има една таблица за съхранение на информация за продуктите, друга таблица за съхранение на информация за поръчките и трета с информация за клиентите.


Products
Product ID
Product Name
Supplier
1
Chai
Exotic Liquids
2
Chang
Exotic Liquids
За
Customers
Company Name
Contact Nan
Alfreds Futterkiste
Maria Anders
Ana Trujillo Emparedados y helados
Ana Trujillo
Orders
Order ID
Customer
Employee
10248
Wilman Kala
Buchanan, Ste
10249
Tradição Hipermercados
Suyama, Mich
10250
Hanari Carnes
Peacock, Marg
Запис:
1 от 48
Без филтър
Търсене

Структура на система за управление на релационни бази от данни- Access

- **Ядро на БД** – данни, организирани в релации (таблици)
  - Всеки ред на таблицата (запис) представлява един обект
  - колоните на таблицата, наречени още полета, представляват характеристиките на обекта
- **Функционална обвивка**
  - форми, заявки, отчети, макроси и модули.
  - Предназначени са да извършват операции с данни и да автоматизират обслужването на базата.
  - Създават се за конкретни нужди на потребителя.

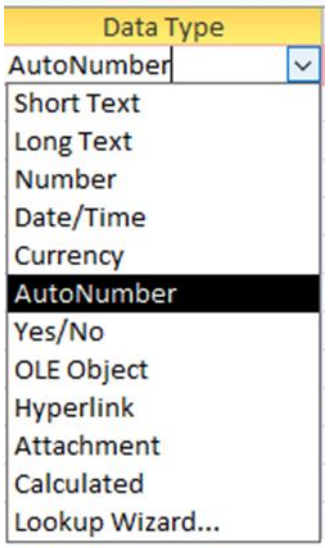
Функционална обвивка в MS Access

- А) ФОРМУЛЯРИ/ФОРМИ(Forms)
  - предназначени за въвеждане и
  - обработване на данни в таблиците
- Б) ЗАЯВКИ(Queries)
  - Търсене и обновяване на данни;
  - Представяне на резултати от търсене и
  - анализ на данните;

Функционална обвивка в MS Access

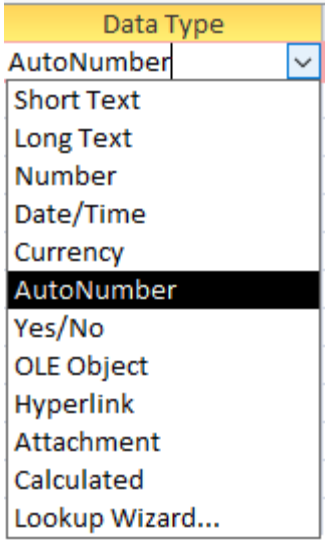
- В) СПРАВКИ/ОТЧЕТИ(Reports)
- подготовка и печат на данните, които се съдържат в таблиците или заявките, като изходните данни са оформени по желан от потребителя начин
- Г) МАКРОСИ (Macros)–
  - Готови процедури за автоматизиране на действия
- Д) МОДУЛИ(Modules) –
  - съдържат описание на процедури за описание на данните на езика за програмиране Visual Basic

Типове данни в Access



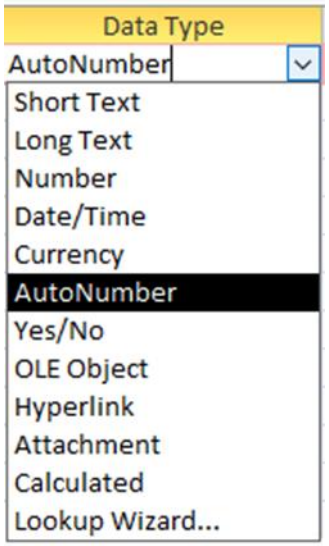
- **Число** (Number) - числови стойности, които в зависимост от установения формат, могат да бъдат цели, дробни или отрицателни
  - byte - [0,255];
  - integer -[-32768,32767];
  - long integer;
  - single - приближено, 7 значещи цифри;
  - double - дълго приближено, 15 значещи цифри;

Типове данни в Access



- Б) **Short Text** - текст с дължина не по-голяма от 255 символа;
- В) **Long Text/Memo** - бележка, подобно на текст, но с дължина приблизително 65000 знака.
- Г) **Date/Time** - за календарни дати или времеви стойности или и на двете едновременно;

Типове данни в Access



- Д) **Currency** - парична стойност, мащабирано число с 15 десетични цифри в цялата част и 4 цифри в дробната;
- Е) **Yes/No** - логическа стойност или алтернативни значения;
- Ж) **Auto Number** - автоматично определяно от СУБД цяло число – брояч. Стойността се увеличава автоматично при въвеждането на нов запис в таблицата.
- **Нестандартни типове** като OLE Object; Hyperlink; Attachment; Calculated и др.

Моделиране ядрото на база данни

- Определяне на целите, на които ще служи БД.
- Проектиране на БД – проучване на данните и определяне на полетата в таблиците и типа на техните данни; определяне на ключови полета; определяне на свързаността на данните и разделянето им в няколко таблици и т.н.
- Обмисляне на справките, които бихме искали да правим в базата;

Проектиране на база данни

- Процесът на проектиране на базите данни се ръководи от известни принципи.
- **Първият принцип** е, че дублирането на информация (наричано още излишни данни) е лошо, защото се заема излишно място и се увеличава вероятността за грешки и несъответствия.
- **Вторият принцип** е, че точността и пълнотата на информацията са важни. Ако базата данни съдържа неточна информация, всички отчети, базирани на тази информация също ще са неточни. В резултат на това всички решения, които се вземат въз основа на тези отчети, ще са погрешни.

Проектиране на БД

Следователно една добра база данни е тази, която:

- Разделя информацията в предметно базирани таблици, за да се намалят излишните данни.
- Предоставя на Access необходимата информация за обединяване на информацията в таблиците, ако е необходимо.
- Помага за поддържането и осигурява точността и целостта на информацията.
- Приспособена е за нуждите на обработката на данни и съставяне на отчети.

Процес на проектиране на БД

Процесът на проектиране се състои от следните стъпки:

1. **Определяне на предназначението на базата данни** - Това помага за подготовката на останалите стъпки.
2. **Намиране и организиране на необходимата информация** - Събира се информация от всякакъв вид, която ще се запише в базата данни, като например име на продукт и номер на поръчката.

Процес на проектиране на БД

Процесът на проектиране се състои от следните стъпки:

3. **Разделяне на информацията в таблици** -Елементите на информацията се разделят на главни единици или обекти като "Продукти", „Потребители“ или "Поръчки". След това всеки обект става на таблица.
4. **Превръщане на информационните елементи в колони** -Решава се каква информация ще се съхранява във всяка таблица. Всеки елемент става поле или колона в таблицата. Например, една таблица за служителите може да включва полета като „Име„, „Фамилия“ и "Дата на постъпване".

Процес на проектиране на БД

5. **Задаване на първични ключове** -Избира се първичен ключ за всяка таблица. Първичният ключ е колона, която еднозначно идентифицира всеки запис в нея. Пример за това може да е ИД на продукт или ИД на поръчка.
6. **Свързване на таблиците в обща структура** - Разглежда се всяка таблица и се решава как да се свържат данните от една таблица с данните от другите таблици. Добавят се полета в таблиците или се създават нови таблици за построяване на връзките, ако е необходимо.

Процес на проектиране на БД

7. **Прецизиране на проекта** -Проектът се анализира за грешки. Създават се таблиците и се добавят няколко записа с примерни данни. Проверява се дали от таблиците могат да се получат желаните резултати. Правят се корекции на проекта, ако е нужно.
8. **Прилагане на правилата за нормализация** - Прилагат се правилата за нормализация на данните, за да се види дали таблиците са структурирани правилно. Правят се корекции в таблиците, ако е нужно.

Определяне предназначението на БД

- Добре е да се запише бъдещото използване на базата данни на хартия – предназначението ѝ, очакваното ѝ използване и кой ще е я използва.
- Ако базата данни е по-сложна или ще се използва от много хора, както често се случва за фирмено използване, предназначението може лесно да представлява един абзац или повече, и трябва да включва кога и как всеки от тях ще използва базата данни.
- Намерението е да има добре развито изложение на целта, към което да се правят обръщения по време на целия процес на проектиране. Наличието на такова изложение помага да се съсредоточим върху целите си при взимане на решения.

Определяне на информацията

- Започва се със съществуващата информация.

***Например**, информация за клиенти във формуляри на хартия. Съберете тези документи и направете списък за всеки тип информация в тях (например за всяко поле, попълнено във формуляра).*

- Определяме всеки от тези елементи. Например: име на клиента, адрес, град, държава, пощенски код и телефонен номер.
- Всеки от тези елементи представлява потенциална колона в една таблица.

Определяне на информацията

- След това разглеждаме видовете отчети или съобщения, които бихме желали да получим.
- Например при отчет за продажба на продукти може да са важни продажбите по райони, по населени места...
- Може също така да имаме нужда от създаването на формулярни писма до клиентите със съобщения за разпродажби или предложения за промоции и награди.
- Мислено си представяме отчета.

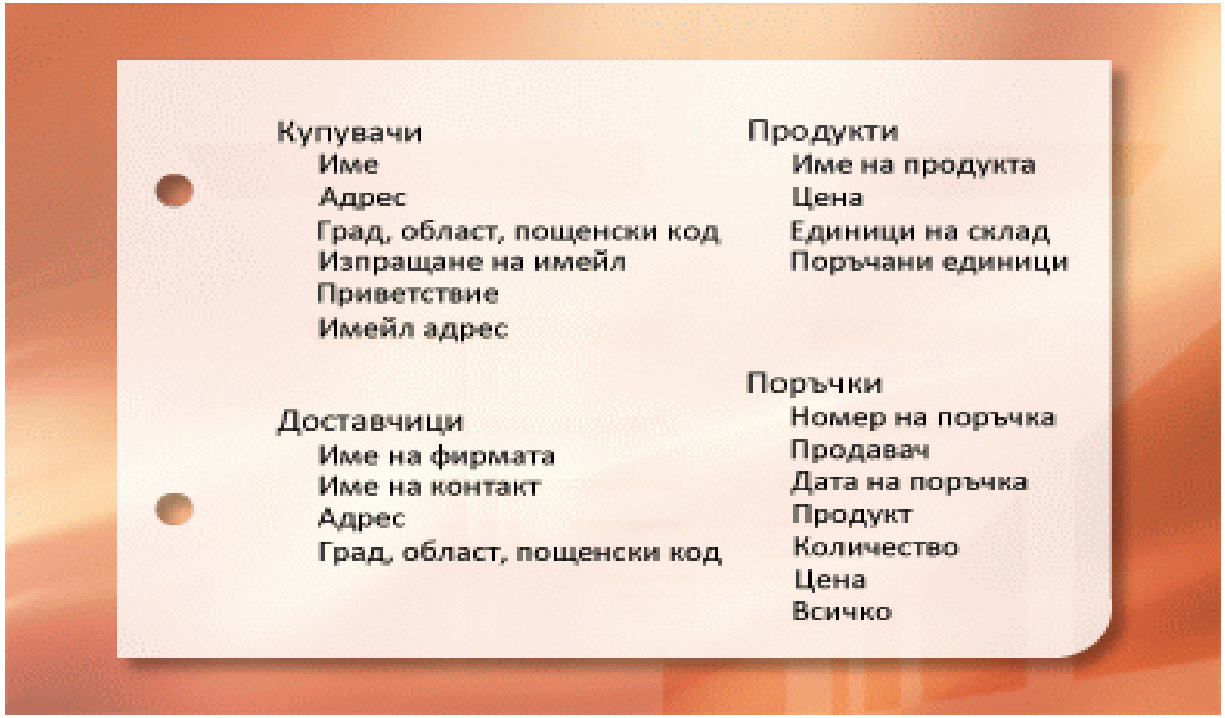
Определяне на информацията

- Важно е всеки обем информация да се разделя на най-малките си полезни части. Когато се касае за име, за да бъде фамилията лесно достъпна, трябва името да се раздели на собствено и фамилно име.
- Важни са въпросите, на които трябва да отговори базата данни.
- Предвиждането на тези въпроси ни помага да се подготвим за запис на допълнителни елементи.

Например, *какъв брой продажби на актуалния продукт са приключени през миналия месец? Къде живеят най-добрите ви клиенти? Кой е доставчикът на най-продавания ви продукт?*

Разделяне на информацията в таблици

- За да разделим информацията в таблици, определяме главните единици или обекти.
- Например, след намирането на информацията за базата данни за продажба на продукти, предварителният списък може да изглежда така:

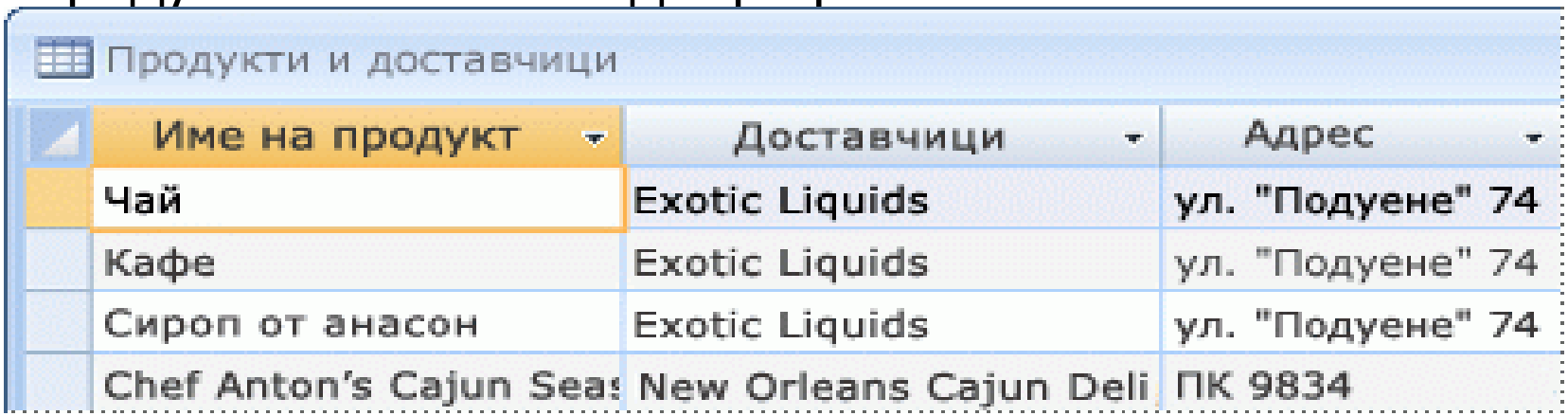


Разделяне на информацията в таблици

- Главните единици, представляват продуктите, доставчиците, клиентите и поръчките.
- Поради това има смисъл да се започне с тези четири таблици, съдържащи факти съответно за: продуктите, доставчиците, клиентите и поръчките.
- Когато най-напред разглеждаме предварителния списък с елементи, може да се изкушим да ги поставим заедно в една таблица, вместо в тези четири таблици.

Защо не запишем всичко в една таблица?

- Всеки ред съдържа информация и за продукта, и за доставчика му.
- Тъй като може да имаме много продукти от един и същ доставчик, информацията за името и адреса на доставчика трябва да бъде повторена много пъти.
- Това заема излишно място на диска. Еднократното записване на информацията за доставчика в отделна таблица за доставчиците и свързването ѝ с таблицата "Продукти" е много по-добро решение.


Продукти и доставчици

Име на продукт	Доставчици	Адрес
Чай	Exotic Liquids	ул. "Подуене" 74
Кафе	Exotic Liquids	ул. "Подуене" 74
Сироп от анасон	Exotic Liquids	ул. "Подуене" 74
Chef Anton's Cajun Sea:	New Orleans Cajun Deli	ПК 9834

Освен това:

- Ако трябва да се промени информацията за доставчика (напр. адреса), трябва да направим тази промяна на всички редове. Записването на адреса на доставчика само на едно място в таблица „Доставчици“ разрешава този проблем.
- Принцип: При проектирането на базата данни се стараем да запишем един факт само веднъж. Ако установим, че повтаряме една и съща информация (напр. адреса на конкретен доставчик) повече от един път, поставяме я в отделна таблица.

Освен това:

- Да предположим, че има само един продукт, доставен от Coho Winery и искаме да го изтрием, но да запазим информацията за доставчика. Това е невъзможно, тъй като всеки запис съдържа факти както за продукт, така и за доставчика му и не може да изтрием едното, без да изтрием другото.
- След като вече е избран обектът за представяне в една таблица, колоните в нея трябва да съхраняват факти само за този обект. Например таблицата за продукти трябва да съхранява факти за продукти.
- Тъй като адресът на доставчика е факт за доставчика, а не за продукта, той ще принадлежи на таблицата за доставчици.

Превръщане на информационните елементи в колони

- За да определим колоните в една таблица, трябва да решим каква информация е необходимо да се проследява за обекта, записан в таблицата.

Например *един добър начален списък на колони за таблицата "Клиенти" съдържа информация за име, адрес, град-държава-код, имейл, обръщение и др.*

- След като сме определили първоначалния набор от колони за всяка таблица, можем да продължим с по-нататъшното прецизиране на колоните.

Например*, има смисъл името на клиента да се съхранява като две отделни колони – собствено и фамилно име; адресът всъщност се състои от пет отделни компонента – адрес, град, област, пощенски код и държава/регион.*

Правила за определяне на колоните

- **Не включваме изчислителни данни** -В повечето случаи резултатите от изчисления не трябва да се съхраняват в таблици. Вместо това Access ще извърши изчисленията, когато искаме да видим резултата.
- **Информацията се съхранява разделена на най-малките си логически части** -Когато обединяваме повече от един тип информация в едно поле, по-късно ще бъде трудно да се извлекат отделните факти.

Купувачи
- Име
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Изпращане на имейл
- Приветствие
- Имейл адрес

Доставчици
- Име на фирма
- Име на контакт
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Телефон

Продукти
- Име на продукта
- Единична цена
- Единици на склад
- Поръчани единици
- Количество на единица

Поръчки
- Номер на поръчка
- Продавач
- Дата на поръчка
- Продукт
- Количество
- Цена

Задаване на първични ключове

- Всяка таблица трябва да включва колона или набор от колони, които **идентифицират еднозначно всеки ред в таблицата**. Това често е уникален идентификационен номер като ИД на служител или сериен номер.
- В терминологията на базите данни тази информация се нарича **първичен ключ** на таблицата.
- Ако имаме уникален идентификатор за дадена таблица, като например номер на продукт, който идентифицира еднозначно всеки продукт в каталога, можем да използвате този идентификатор като първичен ключ на таблицата – но само ако стойностите в тази колона ще са различни за всеки запис.
- **В един първичен ключ не може да има дублиращи се стойности.**

Задаване на първични ключове

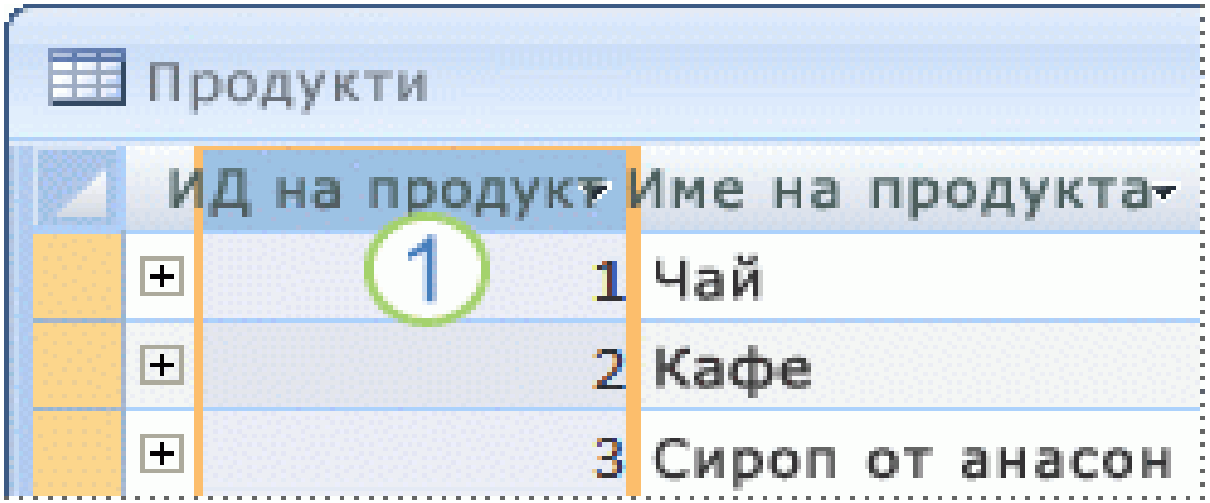
Един първичен ключ трябва винаги да има стойност.

- В една база данни, използваща повече от една таблица, първичният ключ на таблицата може да се използва като препратка към други таблици. Ако първичният ключ се промени, промяната трябва да се приложи навсякъде, където е посочен първичният ключ.
- Често за първичен ключ се използва произволен уникален номер или автоматично увеличаващ се номер.

Например, *може да присвоите на всяка поръчка уникален номер, на всеки човек- неговото ЕГН, на всеки студент- неговия факултетен номер.*

Задаване на първични ключове

- Ако не се сещаме каква колона или набор от колони биха били добър първичен ключ, можем да ползваме данни от типа на ***автономериране.***
- Access автоматично номерира с типа данни AutoNumber



Задаване на първични ключове

- В някои случаи може да искаме две или повече полета заедно да изпълняват ролята на първичен ключ на таблицата.
  - Например за таблицата "Подробни данни за поръчките", съхраняваща елементи от редове за поръчки, за първичния ключ биха се използвали две колони: ИД на поръчка и ИД на продукт.
- Когато един първичен ключ се състои от повече колони, той се нарича **комбиниран (съставен) първичен ключ**.

Купувачи
- ИД на купувач
- Име
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Изпращане на имейл
- Приветствие
- Имейл адрес

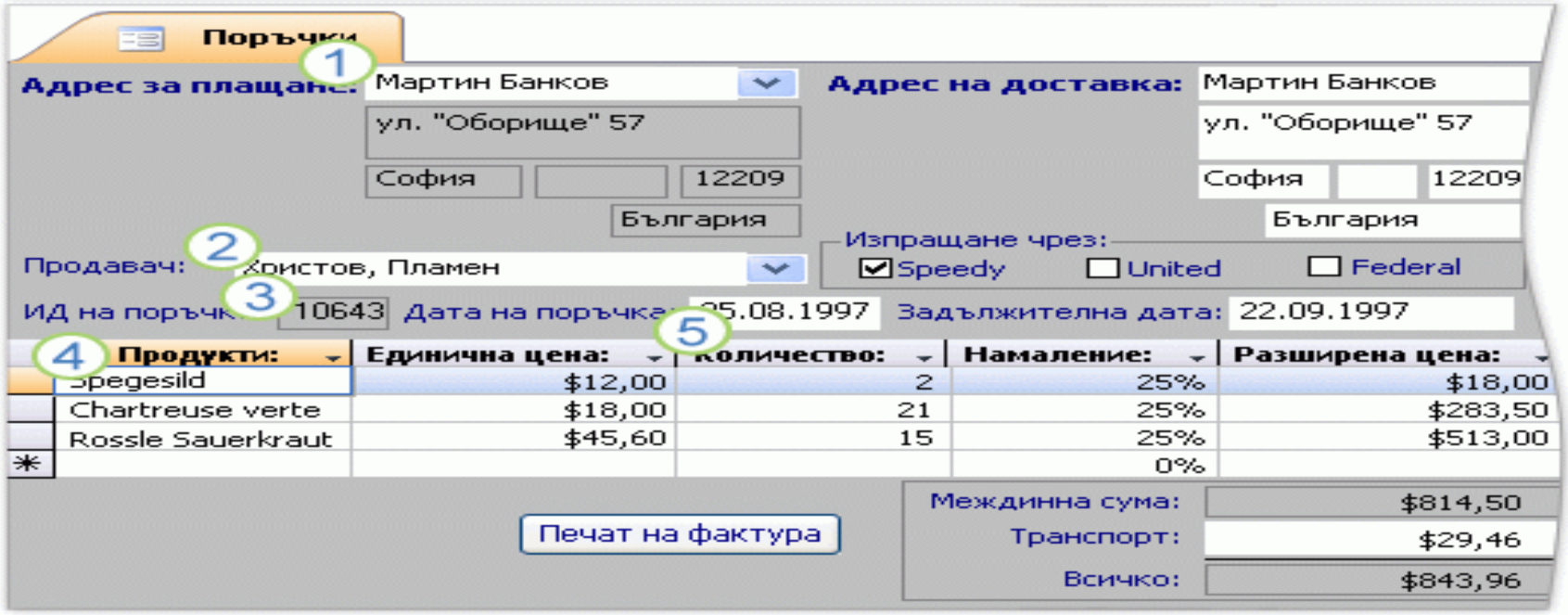
Доставчици
- ИД на доставчик
- Име на фирма
- Име на контакт
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Телефон

Продукти
- ИД на продукта
- Име на продукта
- Единична цена
- Единици на склад
- Поръчани единици
- Количество на единица

Поръчки
- ИД на поръчка
- Продавач
- Дата на поръчка
- Продукт
- Количество
- Цена

Създаване на релации на таблици

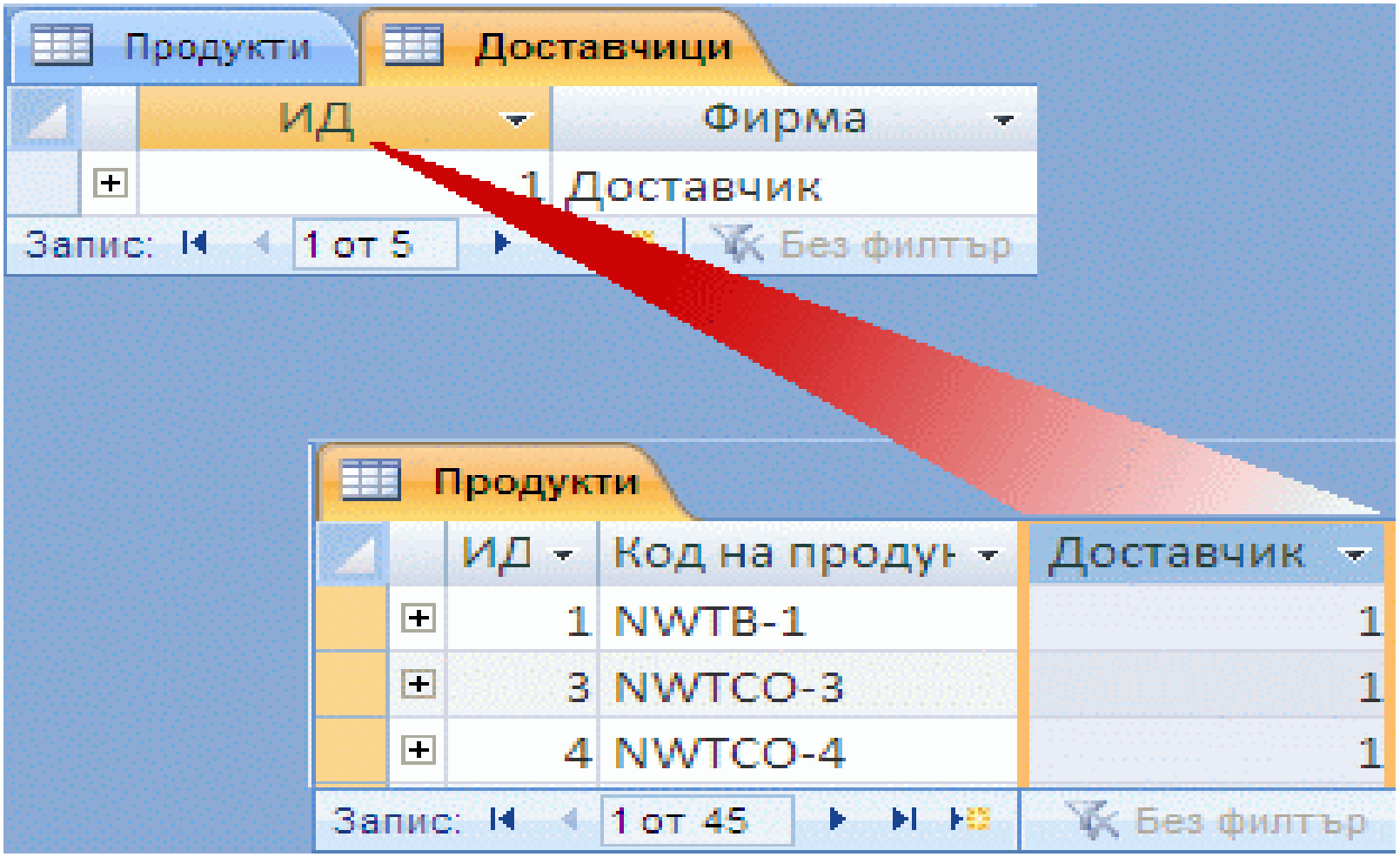
- След като разделихме информацията в таблици, имаме нужда от начин, по който да обединим информацията отново по смислен начин.
- Например следващият формуляр съдържа информация от няколко таблици.



Създаване на връзка "едно към много"

- Да разгледаме таблиците "Доставчици" и "Продукти" в базата данни за поръчка на продукти. Знае се, че един доставчик може да снабдява с всякакъв брой продукти.
- Следователно за всеки доставчик представен в таблицата "Доставчици" може да има много продукти, представени в таблицата "Продукти". Поради това връзката между таблицата "Доставчици" и таблицата "Продукти" е от типа "едно към много".

Връзка „едно към много“



Създаване на връзка „едно към много“. Външен ключ.

- За да представим връзката "едно към много", вземаме първичния ключ от страната "един" на връзката и го добавяме като допълнителна колона или колони към таблицата от страната "много" на връзката. В този случай, например, добавяме колоната за ИД на доставчик от таблицата "Доставчици" към таблицата "Продукти".
- След това Access може да използва ИД номера на доставчика в таблицата "Продукти", за да намери кой е доставчика за всеки продукт.
- Колоната с ИД на доставчика в таблицата "Продукти" се нарича **външен ключ**.
- Един външен ключ осигурява връзка с първичен ключ на друга таблица.

Купувачи
- ИД на купувач
- Име
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Изпращане на имейл
- Приветствие
- Имейл адрес

Доставчици
- ИД на доставчик
- Име на фирма
- Име на контакт
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Телефон

Продукти
- ИД на продукта
- Име на продукта
- Единична цена
- Единици на склад
- Поръчани единици
- Количество на единица
- ИД на доставчик

Поръчки
- ИД на поръчка
- Продавач
- Дата на поръчка
- Продукт
- Количество
- Цена

Създаване на връзка "много към много"

- Да разгледаме връзката между таблиците "Продукти" и "Поръчки".
- Една поръчка може да включва повече от един продукт.
- От друга страна един продукт може да се появи в много поръчки.
- Следователно за всеки запис в таблицата "Поръчки" може да има много записи в таблицата "Продукти". И за всеки запис в таблицата "Продукти" може да има много записи в таблицата "Поръчки".
- Този тип отношение се нарича връзка **"много към много„**

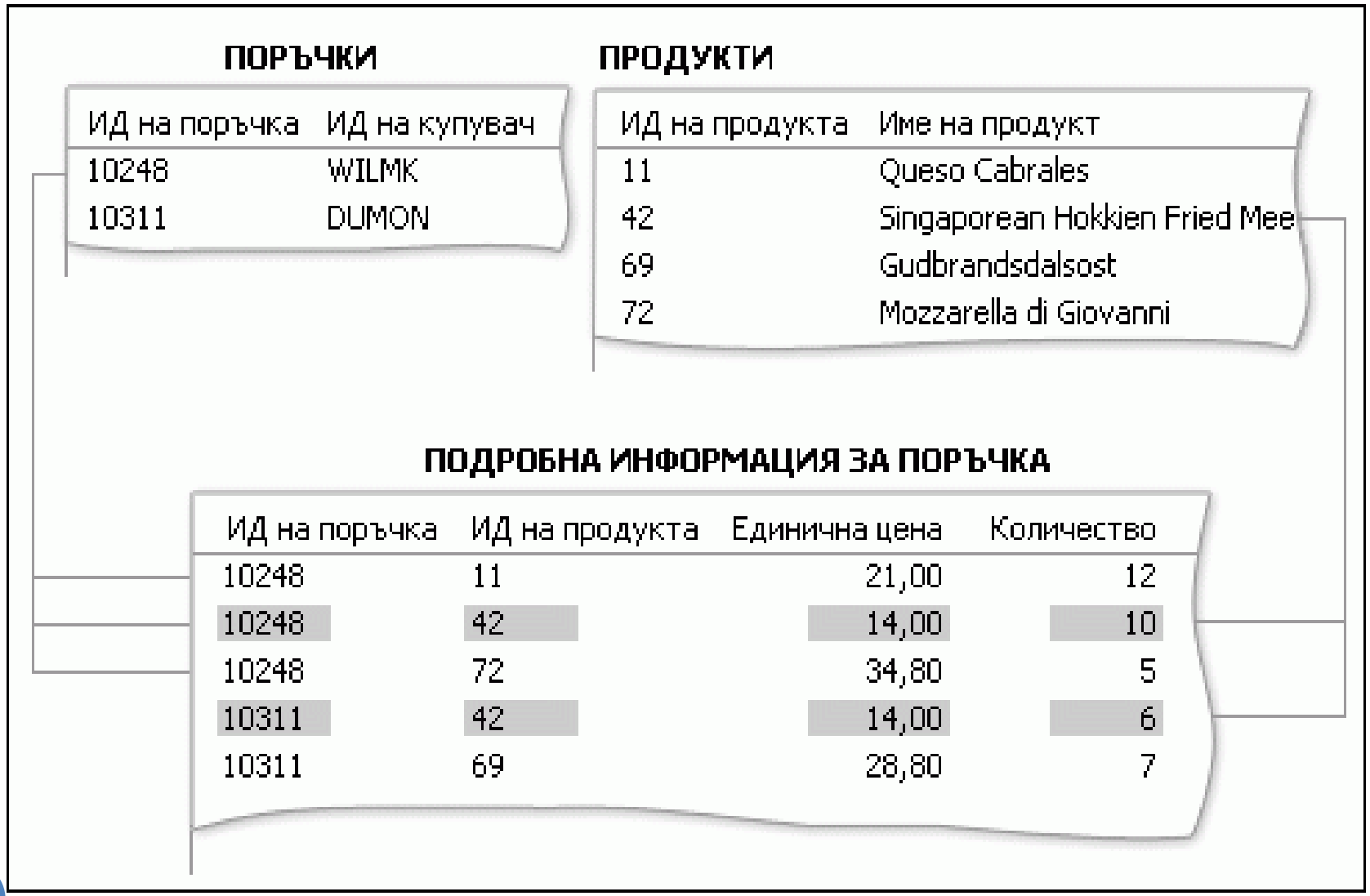
Създаване на връзка "много към много"

- Връзката “Много към много” не може да се създаде като „едно към много“ .
- Ако се опитаме да създадем връзката между двете таблици чрез добавяне на полето "ИД на продукт" към таблицата "Поръчки„, ще ни е необходим повече от един запис на поръчка в таблицата "Поръчки".
- Ще установим същия проблем, ако поставите полето "ИД на поръчка" в таблицата "Продукти" – ще ни е необходим повече от един запис в таблицата "Продукти" за всеки продукт.

Как се решава този проблем?

Чрез създаването на трета съединителна таблица, която разбива връзката "много към много" на две връзки "един към много".

Създаване на връзка „много към много“



ПОРЪЧКИ

ИД на поръчка	ИД на купувач
10248	WILMK
10311	DUMON

ПРОДУКТИ

ИД на продукта	Име на продукт
11	Queso Cabrales
42	Singaporean Hokkien Fried Mee
69	Gudbrandsdalsost
72	Mozzarella di Giovanni

ПОДРОБНА ИНФОРМАЦИЯ ЗА ПОРЪЧКА

ИД на поръчка	ИД на продукта	Единична цена	Количество
10248	11	21,00	12
10248	42	14,00	10
10248	72	34,80	5
10311	42	14,00	6
10311	69	28,80	7

Създаване на връзка „много към много“

- Всеки запис в таблицата "Подробни данни за поръчките" представлява един елемент на ред от поръчката.
- Първичният ключ на таблицата "Подробни данни за поръчките" се състои от две полета – външните ключове от таблиците за поръчки и продукти.
- В базата данни за продажба на продуктите таблицата "Поръчки" и таблицата "Продукти" не са свързани пряко една с друга. Вместо това те са свързани непряко чрез таблицата "Подробни данни за поръчките".
- Отношението "много към много" между поръчките и продуктите е представено в базата данни чрез използване на две връзки "един към много"

Купувачи
- ИД на купувач
- Име
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Изпращане на имейл
- Приветствие
- Имейл адрес

Доставчици
- ИД на доставчик
- Име на фирма
- Име на контакт
- Адрес
- Град
- Регион
- Пощенски код
- Страна
- Телефон

Продукти
- ИД на продукта
- Име на продукта
- Единична цена
- Единици на склад
- Поръчани единици
- Количество на единица
- ИД на доставчик

Поръчки
- ИД на поръчка
- Продавач
- Дата на поръчка
- ИД на купувач
- Име на транспорта
- Адрес за доставка
- Град на доставка
- Регион на доставка
- Пощенски код на доставка
- Страна на доставка

Подробна информация за поръчка
- ИД на поръчка
- ИД на продукта
- Единична цена
- Количество

Създаване на връзка "едно към едно"

- Друг вид отношение е връзката "едно към едно". Използва се когато искаме да запишем някаква специална допълнителна информация за продукта, която ще е необходима рядко.
- Връзката "един към един", можем да осигурим по няколко начина:
  - Ако двете таблици имат един и същ обект, задаваме връзката като използваме един и същ първичен ключ за двете таблици.
  - Ако двете таблици имат различни първични ключове, избираме една от таблиците (без значение коя) и вмъкваме първичния й ключ в другата таблица като външен ключ.

ИЗВОДИ:

1. ***Когато съществува връзка от типа "едно към едно" или "едно към много", участващите таблици трябва да споделят обща колона или колони.***
2. ***Когато съществува връзка от типа "много към много", за представяне на връзката е необходима трета таблица, която съдържа външни ключове към двете таблици.***

Прецизиране на проекта

- След като създадем БД, трябва да въведем в таблиците примерни данни и да опитаме да работим с тях.
- Това ни помага да открием потенциални проблеми. Необходимо е да проверим:
  - Нужда от нови колони- ако липсва нужната информация трябва да създадем друга таблица.
  - Излишни колони, които могат да се изчислят от наличните.
  - Многократно дублирана информация в някоя от таблиците? Тогава ще трябва да я разделим на две таблици, с връзка"един към много".

Прецизиране на проекта

Освен това проверяваме:

- Дали има ли таблици с малко записи и много празни полета в отделните записи?
- Дали всеки информационен елемент е разбит на най-малките си полезни части?
- Дали всяка колона съдържа факт за предмета на таблицата? Ако една колона не съдържа информация за предмета на таблицата, то тя принадлежи на друга таблица.
- Представени ли са всички връзки между таблиците, било чрез общи полета или трета таблица?

Прилагане на правилата за нормализация

- Нормализацията е процес на подреждане на данните в таблиците.
- Целта е да се представят обектите в БД чрез релации, които да съдържан необходимите за конструирането на обекта данни.
- Трябва да могат лесно да бъдат добавяни, изтривани и модифицирани записите без това да води до нарушаване консистентността на записаните данни.

Прилагане на правилата за нормализация

- Правилата за нормализация са следващата стъпка при проектирането.
- Процесът на прилагане на правилата към проекта на базата данни се нарича нормализиране на базата данни или просто **нормализация.**
- Нормализацията се прилага, за да се убедим, че сме разделили информационните елементи в подходящите таблици.

Прилагане на правилата за нормализация

- Нормализацията се извършва на степени, като всяка следваща степен отговаря на по-високо ниво на нормализация.
- Колкото е по-висока е степента на нормализация, толкова таблиците стават повече, което прави БД по-сложна за управление и намалява нейната производителност.
- Правилата се прилагат последователно, показвайки на всяка стъпка, че проектът е достигнал до една от така наречените "нормални форми".
- Широко възприети са пет нормални форми – от първата до петата нормална форма.
- Ще разгледаме по-подробно първите три, защото те са всичко, което е необходимо за повечето проекти на бази данни.

Първа нормална форма

- **Първата нормална форма:**
- Релацията е в първа нормална форма (1NF) тогава и само тогава, когато всички стойности на данните са неразложими.Това означава, че по дефиниция всички релации са в първа нормална форма, защото прилежащите домейни могат да се дефинират само върху прости типове.
- Т.е. всяко пресичане на ред и колона съществува една единствена стойност и никога – списък от стойности.
- Например не може да имаме поле "Цена", в което да поставим повече от една цена.
- Ако си представяме всяко пресичане на редове и колони като клетка, във всяка клетка може да има само една стойност. С други думи, стойностите в БД са неделими (atomic).

Втора нормална форма

- ***Втора нормална форма*** - релацията е в втора нормална форма (2NF) тогава и само тогава, когато всички неключови атрибути зависят само и изцяло от първичния ключ, т.е. нормализиращата процедура включва разделянето на началната релация на няколко релации, като процесът трябва да е обратим и да не води до загуба на данни.

Втора нормална форма

- **Втората нормална форма** изисква всяка колона без ключ да бъде напълно зависима от целия първичен ключ, а не само от негова част. Това правило се отнася за случаите, когато първичният ключ се състои от повече от една колона.
- Нека например първичният ключ е съставен от ИД на поръчката и ИД на продукта и таблицата има полета:
  - ИД на поръчка (първичен ключ)
  - ИД на продукт (първичен ключ)
  - Име на продукт
- Този проект нарушава втората нормална форма, защото името на продукта зависи от ИД на продукта, но не зависи от ИД на поръчката, така че не зависи от целия първичен ключ. Името на продукта трябва да се премахне от таблицата. То принадлежи на друга таблица (продукти).

Трета нормална форма

- **Третата нормална форма** изисква не само всяка колона без ключ да зависи от целия първичен ключ, но и колоните без ключ да са независими една от друга, т.е. Те трябва да са зависими единствено от първичния ключ.
- Например, нека таблица, съдържа следните колони:
  - ИД на продукт (първичен ключ)
  - Име
  - цена
  - Отстъпка

Трета нормална форма

- Да приемем, че отстъпката зависи от предложената цена на дребно (цена).
- Тази таблица нарушава третата нормална форма, защото една колона без ключ (отстъпка) зависи от друга колона без ключ (цена).
- Независимостта на колоните означава, че трябва да имаме възможност да променяме всяка колона без ключ, без това да се отразява на никоя от другите колони.
- Ако променим една стойност в полето «цена», ще се промени и отстъпката, като по този начин се нарушава това правило.
- В този случай колоната за отстъпката трябва да се премести в някоя друга таблица с ключ.

Заключение и източници

Учебните материали и работните файлове можете да откриете на:

http://sou-brezovo.com

- UN: student-it
- PASS: Plovdiv2018

Използвани източници:

- https://support.office.com/bg-BG/
- https://microsoft.com/bg-BG/download
- http://office.microsoft.com/bg-bg/access/HA012242471026.aspx#Terms