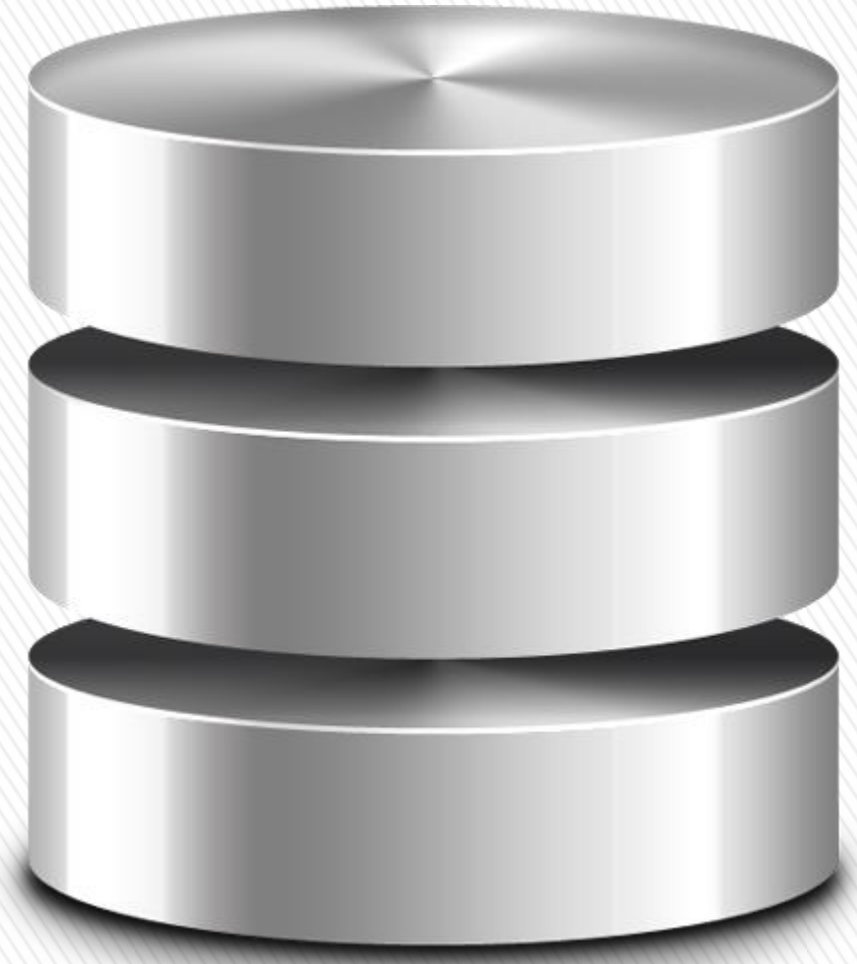
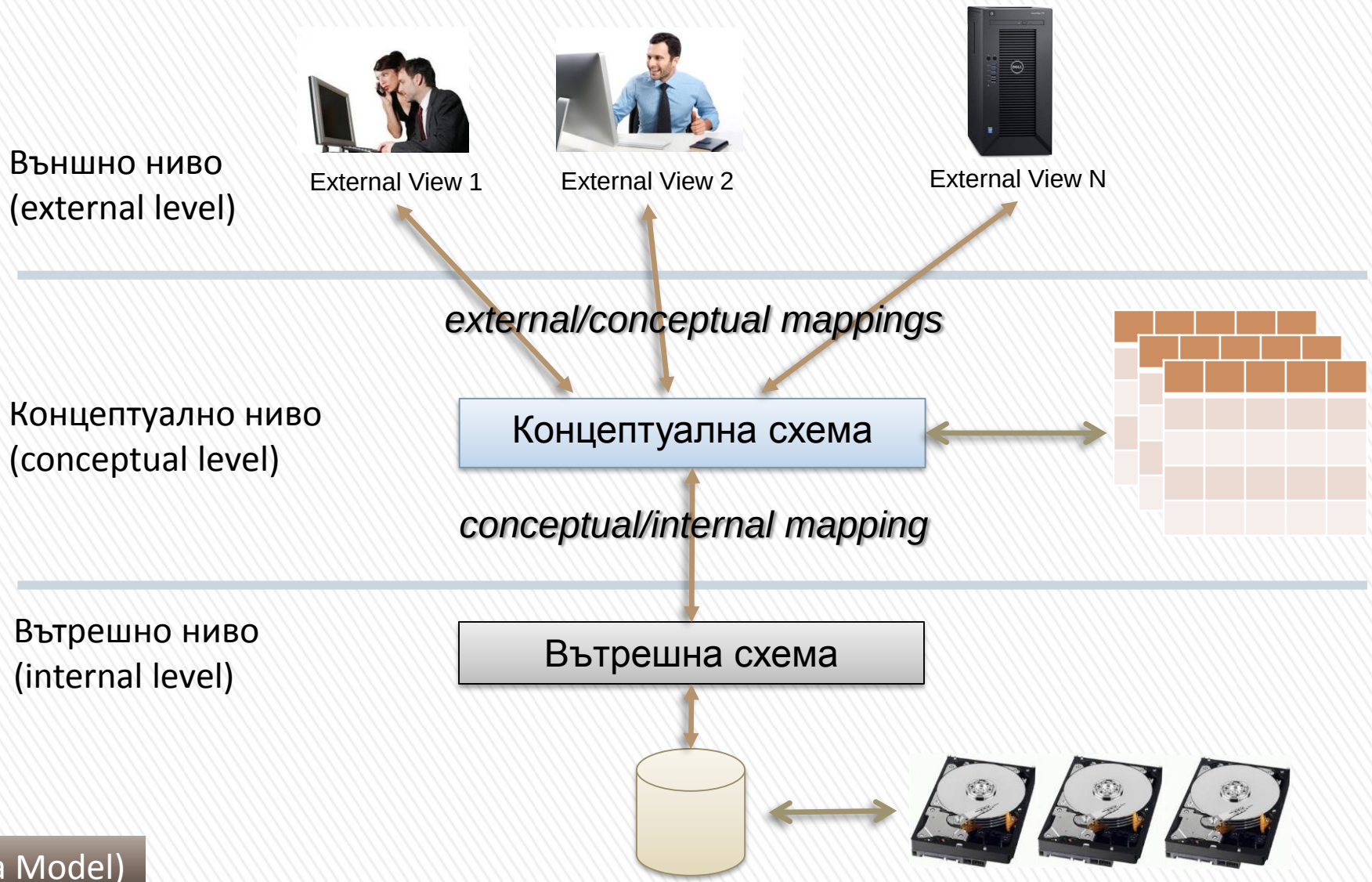


»Гл. ас. д-р Георги Чолаков

»Бази от данни

Архитектура и модели СУБД ›

Три нива на архитектурата



(ANSI/SPARK Data Model)

Външно ниво

- Разглежда възможните начини за избирателното представяне на данните за различните потребители;
- Включва различни външни схеми (views), като всяка схема представя само определена част от данните, касаеща конкретните потребители, скривайки останалата част от данните;
- На външно ниво:
  - Крайните потребители разполагат с приложения с форми, менюта и т.н. за достъп до данните;
  - Програмистите използват обикновено език за програмиране (C++, Java, C#... – host language), който има подезик за работа с база от данни (DSL – Data Sublanguage).

Концептуално ниво

- Схемата му представя структурата на цялата база данни, скривайки детайлите относно физическото съхраняване на данните, концентрирайки се върху обектите, взаимоотношенията, типовете данни, потребителските операции и ограничения;

- Концептуалната схема трябва да се придържа към изискванията за независимост на данните.

Вътрешно ниво

- Представя на ниско ниво цялата база данни, най-близо е до физическата памет;
- Има вътрешна схема, която описва физическата структура на съхраняване на данните във файловете, представяйки актуалното им разположение;
- Но все пак не се занимава с физическото съхранение на файловете във вторичната памет – сектори, цилиндри и т.н.

Кореспонденции (mappings)

» Кореспонденциите между вътрешното и концептуално ниво (conceptual/internal mapping) определят как данните на концептуално ниво се представят на физическото.

Ако структурата на съхранение на данните бъде променена, то тези кореспонденции също трябва да бъдат променени, за да не се наложи промяна в концептуалната схема. Така ще бъде запазена ***физическата*** *независимост на данните.*

Кореспонденции (mappings)

» Кореспонденциите между външното и концептуално ниво (external/conceptual mapping) определят начина на представяне на данните от потребителска гледна точка.

Това се постига чрез програмните приложения и логическата гледна точка на използвания програмен език и подезик за данните.

Ако концептуалната схема може да бъде променена без това да доведе до промяна на външната схема казваме, че е налице ***логическа*** *независимост на данните.*

Администратор на БД

Администраторът на БД е човекът (или група от хора), отговорен за контрола върху цялата БД. Сред неговите отговорности са:

- Авторизация на достъпа до базата данни;
- Координиране на използването и мониторинг на производителността ѝ;
- Увеличаване на хардуерните и софтуерни ресурси при нужда;
- Архивиране и възстановяване – определяне на стратегия за архивиране, така че при срив да може да бъде възстановена за възможно най-кратък срок;
- Определя структурата за съхраняване на данните в паметта и стратегията за достъп.

Приложен програмист

Тази роля е свързана с разработването на приложен софтуер, осъществяващ достъп до базата данни. Сред отговорностите са:

- Създаване на спецификация с операциите и обектите от БД, до които ще има достъп разработваното приложение;
- Разработване на софтуера с оглед на описаните по-горе спецификации;
- Реинженеринг/разширение на софтуера с оглед реализация на динамично променящите се изисквания (нови версии);
- Отстраняване на възникнали проблеми;
- Документиране.

Функции на СУБД

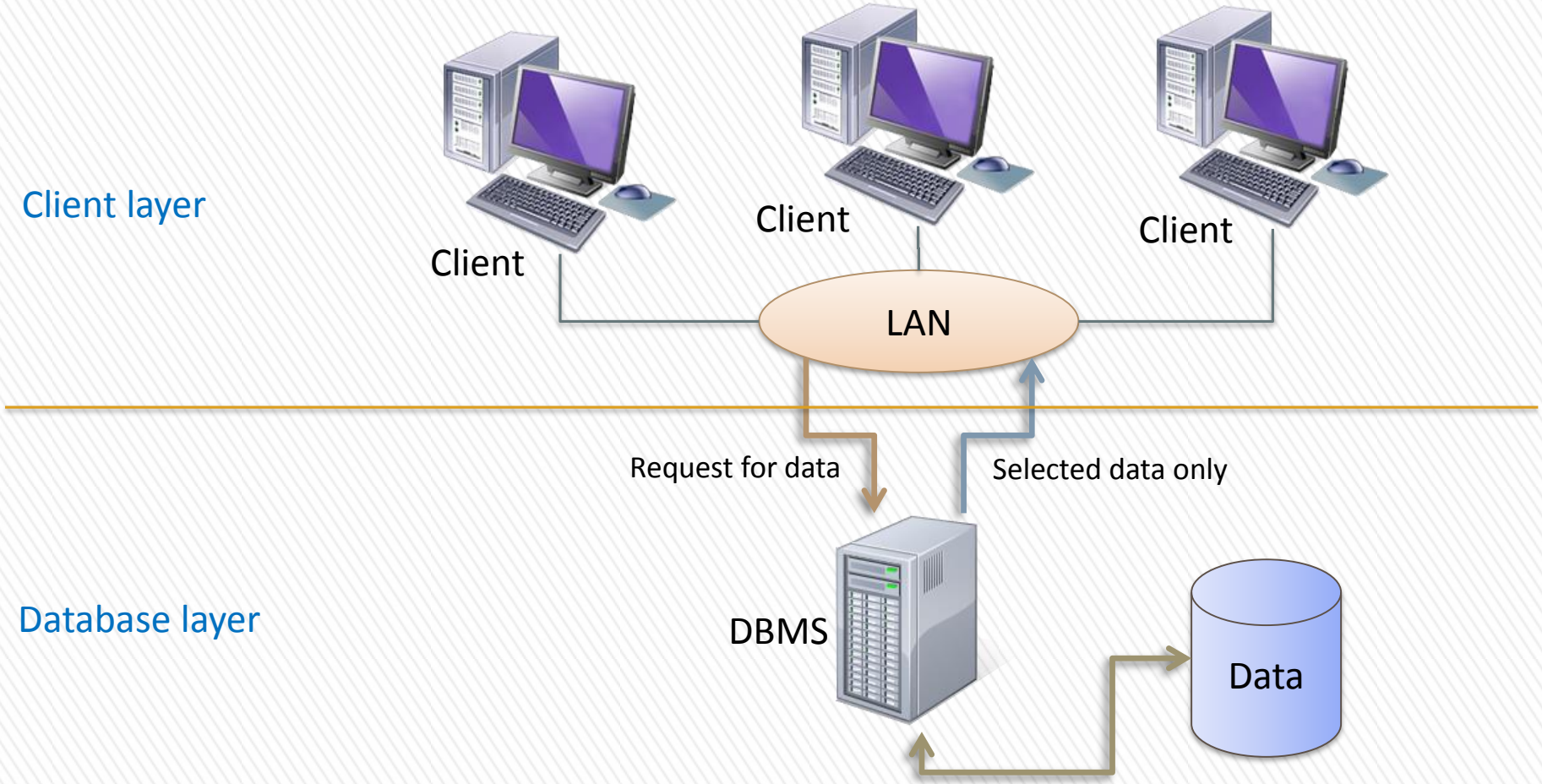
- Дефиниране на структурите за данни – обекти, зависимости, ограничения, изгледи и т.н.;
- Манипулиране на данните – добавяне, променяне, изтриване, извличане;
- Сигурност и интегритет на данните – да следи за евентуални нарушения на правилата за достъп до данните и за правилата за валидността им;
- Управление на конкурентността – когато СУБД позволява едновременен достъп на множество потребители до едни и същи данни тя трябва да осигури възможност потребителите да не си пречат взаимно, конкурирайки се за ресурсите на системата;

Функции на СУБД

- Архивиране и възстановяване – да предоставя възможност за реализиране на стратегия за архивиране и съответно възстановяване от архивни копия;
- Езици за достъп до СУБД, API и комуникационни интерфейси – език за достъп (SQL), библиотеки за програмни езици (ODBC, JDBC, …)

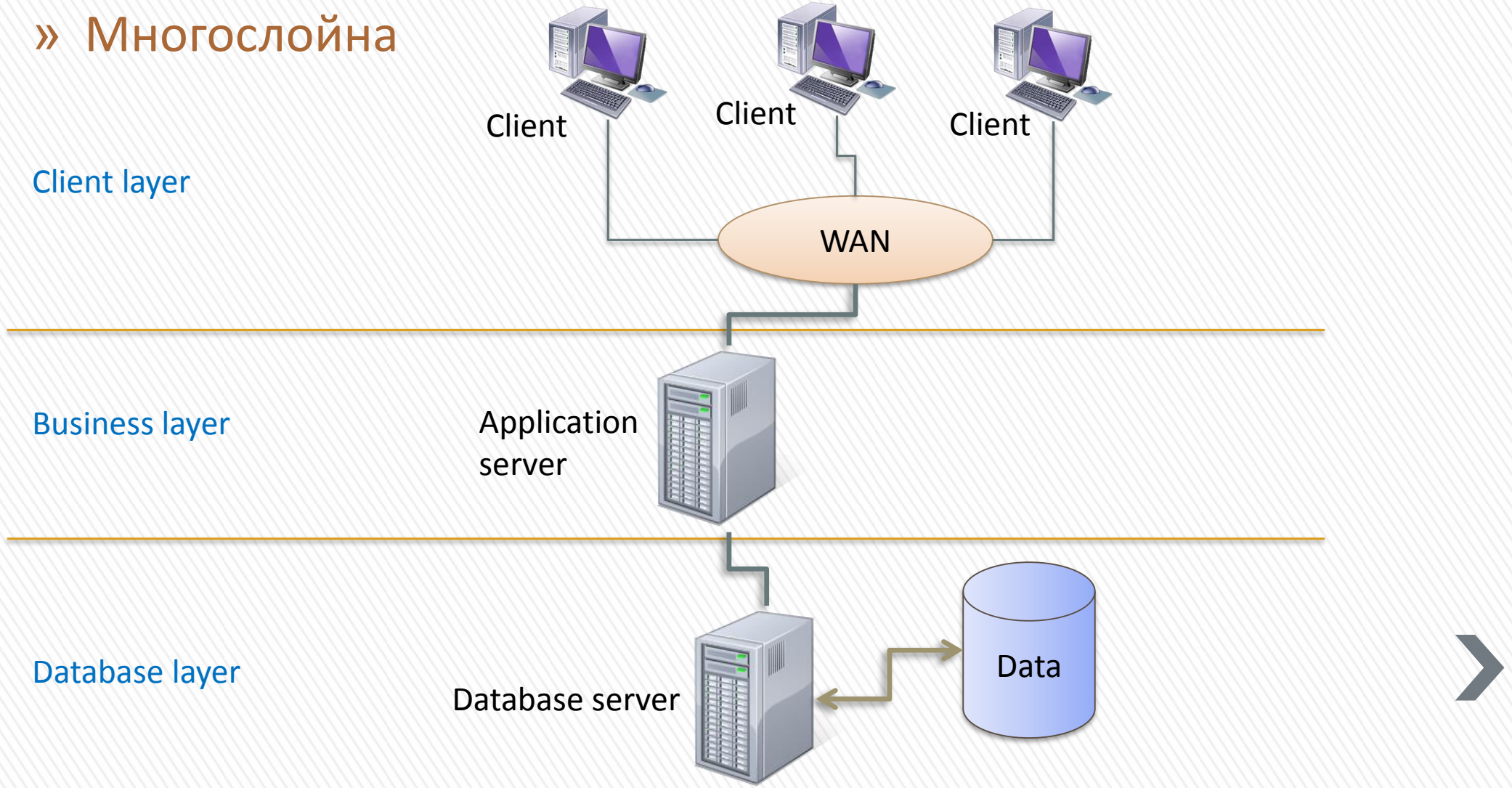
Системна архитектура

» Двуслойна



Системна архитектура

» Многослойна



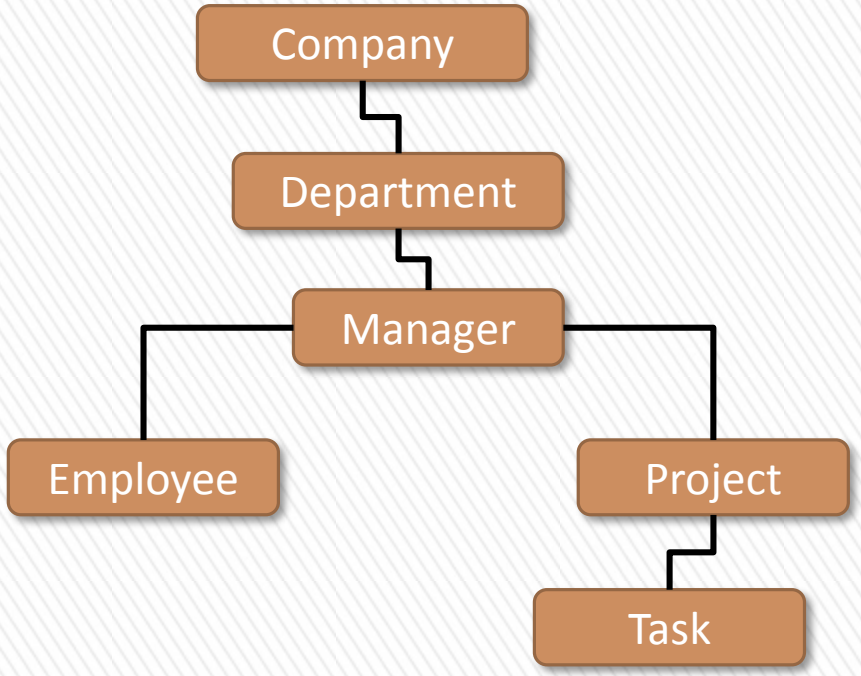
Модели СУБД - еволюция

- Файлови системи (60-70‘s)
  - Първоначално решение, което трудно може да се нарече база от данни (flat files);
  - Съдържа неструктурирани данни обикновено;
  - Ако файлът има определена структура, съдържа разделители (CSV), то вече не е съвсем flat, но все още е далеч от БД;
  - Всяко търсене/извличане на данни от такива файлове трябва да бъде програмирано, а в БД обикновено това е стандартизирано и не е нужно всяка заявка да се програмира;
  - Ако данните се съхраняват в повече от един файл комбинирането им трябва да бъде също прецизно кодирано.

Модели СУБД - еволюция

» Йерархичен модел (70‘s)

Данните се представят като множества от дървовидни структури, като всяка йерархия представя определен брой свързани записи.



Всеки елемент с данни има един родител, а всеки родител може да има няколко наследника, които могат да съществуват само ако родителят съществува. Резултатът е, че този модел поддържа 1:N взаимоотношения.

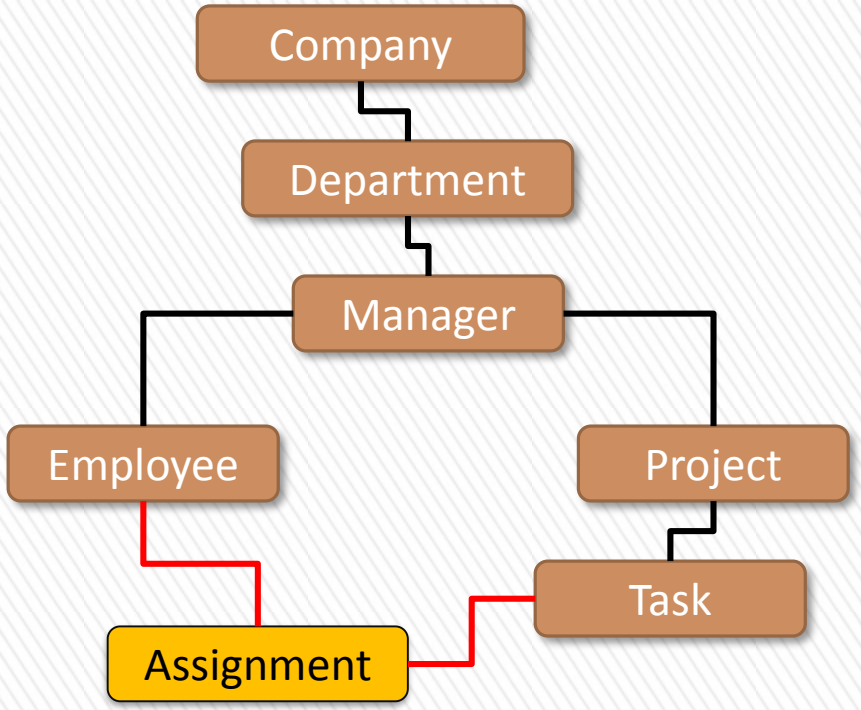
Всяка задача е част от проект, който си има мениджър, който е част от отдел, който е част от фирма.
Недостатъците на този модел са, че всеки достъп трябва да започне от корена, т.е. за да намерим служител трябва да намерим неговата фирма, отдела и мениджъра му.

[IMS, ADABAS, IDS-II]

Модели СУБД - еволюция

» Мрежов модел (70-90‘s)

Усъвършенстване на йерархичния. Позволява наследниците да имат повече от 1 родител, позволявайки M:N взаимоотношения.



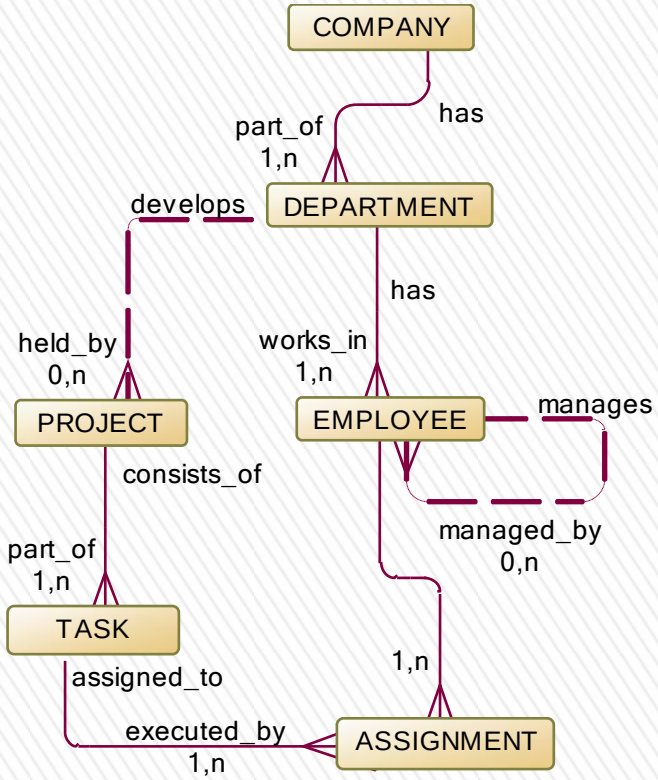
На схемата са показани M:N взаимоотношения между служителите и задачите – на служител могат да бъдат назначени няколко задачи, от друга страна една задача може да бъде изпълнявана от няколко служители.

[IMS, ADABAS, IDS-II]

Модели СУБД - еволюция

» Релационен модел (80‘s)

Представя базата данни като колекция от таблици.

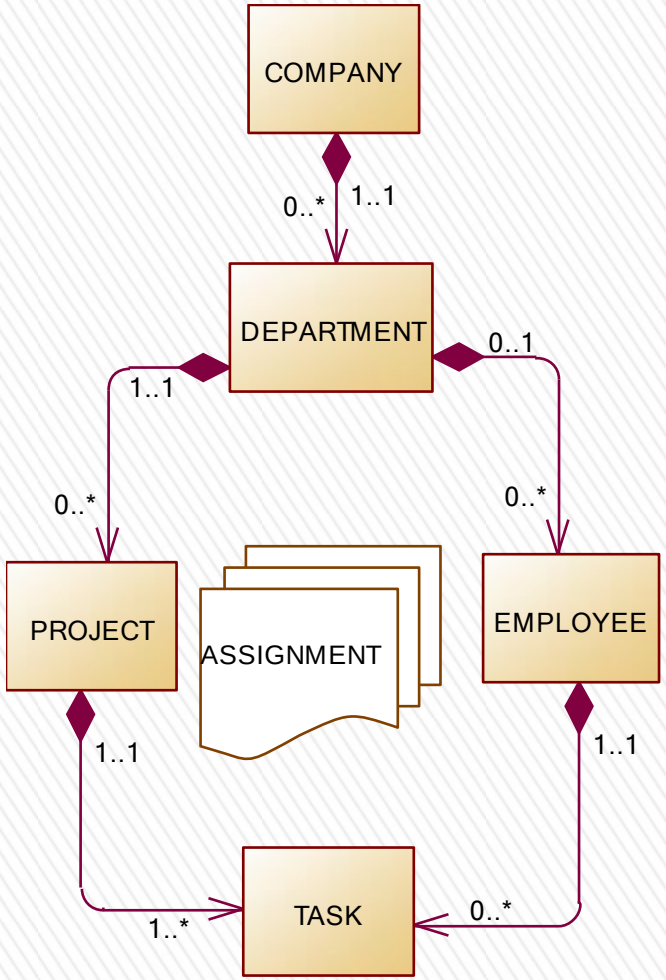


[DB2, Oracle, MS SQL Server, MySQL]

Усъвършенства ограниченията на йерархичната структура без да я отхвърля. Данните могат да бъдат достъпвани директно без да се достъпват първо родителските обекти.
Напр., не е нужно за да бъде намерен служител да бъдат намерени първо неговата фирма, отдел и т.н.

Предложен през 1970 от E.F. Codd (IBM). Смята се за основополагащо откритие, а неговата простота поражда истинска революция в света на базите данни.

Модели СУБД - еволюция



» Обектен модел (90‘s)

Дефинира базата данни в термините на обекти, техни характеристики и операции.

По-неефективен от релационния в случаите, когато трябва да се извличат данни за повече от една инстанция, но добавя обектно-ориентирани възможности – наследяване, представяне на комплексни елементи в йерархии и др.
Решава проблемите със сложните представяния на M:N взаимоотношения като ги заменя с колекции, включени в класовете. Задачите от проектите и назначението им на служители е решено с използването на колекции от задачи в тези два класа.

[Versant, Objectivity/DB, DB2 UDB, Oracle 12c]

Модели СУБД - еволюция

- Обектно-релационен модел (90‘s)

Релационна база данни, поддържаща и обектно-ориентирани структури и механизми.

Разширение на релационния модел с цел да покрие изискванията за съхранение на комплексни данни – капсулирани обекти с атрибути и методи. Предимството е, че потенциално големи обекти могат да бъдат съхранявани в поле от таблица.

Много от съвременните водещи бази от данни могат да бъдат класифицирани като такива.

[Versant, Objectivity/DB, DB2 UDB, Oracle 12c]

Модели СУБД - еволюция

» XML

XML (eXtensible Markup Language) – език за съхранение на данни в йерархичен вид, удобен за пренос на данни.

```
<Employees>
  <Person>
    <FirstName>Силвия</FirstName>
    <LastName>Николова</LastName>
    <DateOfBirth>1981-01-11</DateOfBirth>
  </Person>
  <Person>
    <FirstName>Иван</FirstName>
    <LastName>Петров</LastName>
    <DateOfBirth>1992-08-24</DateOfBirth>
  </Person>
</Employees>
```

Модели СУБД - еволюция

» Big Data, NoSQL

Данните от Интернет пространството като история на плащания, клиентски предпочитания, тенденция на разглеждане и поведение, от социални мрежи като Facebook, Twitter, LinkedIn са огромно количество и нарастват ежедневно.

В допълнение данни от мобилни устройства, сензори (GPS, RFID, метеорологични и др.) генерират също огромни количества данни в различни формати – текст, изображения, звук и видео.

Нуждата от ефективно съхранение и управление на тези данни поражда нещото, наречено „Big Data“, включващо и обработка на тези данни за извличане на знания.

Модели СУБД - еволюция

» Big Data, NoSQL

Характеристики на Big Data:

- Volume – съхраняват огромни масиви от данни;
- Velocity – бърза обработка за извличане на знания;
- Variety – различни формати на данните.

Проблеми на релационния модел/подход:

- Не винаги е възможно/удобно да се съхраняват неструктурирани данни от социални медии и сензори в релационна структура с редове и колони;
- Добавянето на милиони редове от структурирани и неструктурирани данни ще доведе до сериозно оскъпяване на хардуера;
- Анализът на неструктурирани данни изисква различен от познатия OLAP модел за релационните бази от данни, където данните са структурирани.

Модели СУБД - еволюция

» Big Data, NoSQL

Някои от популярните Big Data технологии:

> Hadoop – Java базирана **разпределена** система за съхранение и обработка на данни. Основните й модули са Hadoop Distributed File System (HDFS – служи за ефективно съхранение на данни) и MapReduce;

> MapReduce – API за паралелна аналитична обработка в разпределена система;

> NoSQL бази от данни – бази от данни (могат да са разпределени), съхраняващи ефективно структурирани и неструктурирани данни.

Модели СУБД - еволюция

» NoSQL

Използваме NoSQL база данни всеки път, когато:

- > Търсим продукти в Amazon;
- > Използваме Facebook или Google Maps;
- > Гледаме видео в YouTube, и др.

Това са ново поколение бази от данни, които имат за цел да посрещнат предизвикателствата на Big Data и имат някои от следните характеристики:

- > Не са базирани на релационен модел (от там и името NoSQL);
- > Използват разпределени архитектури;
- > Предоставят висока мащабируемост (scalability), наличност (availability) и толеранс към възникнали грешки (fault tolerance - продължават да работят при срив на някой от компонентите);
- > Поддържат големи обеми от разпределени данни;
- > Приспособени са за ефективност, а не за консистентност на данните.

Модели СУБД - еволюция

» NoSQL

Няма определен стандарт за модел на NoSQL бази данни. Сред тях са:

- Документни – съхраняват данните в т.нар. документ или полуструктуриран вид, идентифициран чрез уникален ключ. Документът може да е с формат XML, JSON и други (MongoDB);
- Графове (graph store) – използват се за съхранение на мрежи от данни, като социални мрежи (Neo4J, Giraph);
- Column store – оптимизирани за заявки върху големи множества от данни, съхранявайки данните по колони, вместо по редове – така всички данни от дадена колона са съхранени последователно, след това от другата колона и т.н. (Cassandra, HBase);
- Key-value store – най-простият тип, всеки елемент се съхранява като комбинация от ключ и стойност (Riak, Berkeley DB, Oracle NoSQL).