3. Архитектура на СУБД

Лекционен курс "Бази от данни"

Три нива на архитектурата

External (Java)	External (COBOL)

class Emp
{
String empNo;
double sal;
}
	01 EMPC.
	02 EMPNO PIC X(6).
	02 DEPTNO PIC X(4).

Conceptual

```
EMPLOYEE
        EMP_NUMBER   CHAR(6)
        DEPT_NUMBER CHAR(4)
        SALARY          NUMERIC(5)
```

Internal

```
STORED_EMP    LENGTH=20
        PREFIX  TYPE=BYTE(6), OFFSET=0
        EMP#    TYPE=BYTE(6), OFFSET=6, INDEX=EMPX
        DEPT#   TYPE=BYTE(4), OFFSET=12
        PAY     TYPE=FULLWORD, OFFSET=16
```

Пример

- На вътрешно ниво всеки служител е представен с тип вътрешен запис, наречен STORED_EMP, съдържащ 4 полета – префикс (контролна информация, указател) и три полета за данните, съответстващи на атрибутите на служителя;
- На концептуално ниво базата данни съдържа информация за обект, наречен EMPLOYEE, имащ следните атрибути: EMP_NUMBER, DEPT_NUMBER, SALARY;
- На външно ниво, използвайки Java, всеки служител е представен с инстанция на клас, съдържащ два атрибута – номер и заплата (отделът е пропуснат);
- Отново на външно ниво, използвайки COBOL, всеки служител е представен със запис, съдържащ номерата на служителя и отдела (заплатата е пропусната)

Архитектурата на СУБД е разделена на три основни нива (ANSI/SPARK Data Model):

- Външно ниво (external level)
  - най-високо ниво на абстракция, най-близо до потребителите;
  - разглежда възможните начини за избирателното представяне на данните за различните потребители, включва различни външни схеми (views), като всяка схема представя само определена част от базата данни, касаеща конкретни потребители, скривайки останалата част от данните;
  - занимава се предимно с индивидуалните гледни точки на потребителите.

На външно ниво всеки потребител разполага с език:

- За програмиста този език би бил някой конвенционален език за програмиране (Cobol, C++, Java)
- За крайния потребител езикът би бил даден специфичен език за заявки – форми, менюта и т.н

Всеки от тези езици ще включва някакъв подезик за работа с данни, който е включен в основния език.

Заключение:

- В общия случай индивидуалният потребител ще се интересува само от някаква част от цялата база данни;
- Още повече, потребителската гледна точка ще бъде абстрактна спрямо физическото представяне на данните; освен това са възможни различни абстрактни гледни точки на едни и същи данни;
- Така външната гледна точка е съдържанието на базата данни от погледа на конкретен потребител.

- Концептуално ниво (conceptual level)
  - 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 е създадена чрез концептуален език за дефиниране на данни (DDL)
  - Тъй като целим независимост на данните, то дефинициите в този език трябва да бъдат само относно съдържанието на информация;
  - Дефинициите не трябва да включват никакви съображения относно структурата на съхранение или техниките за достъп до данните.

Ако концептуалната схема е направена независима от данните по този начин, то и външните схеми, създадени в термините на концептуалната схема, ще бъдат независими от данните.

Заключение

- Концептуалното ниво е връзката между външното и вътрешното;
- Различните гледни точки се трансформират в единна логическа схема - характерна за конкретното СУБД;
- Различаваме различни външни гледни точки - всяка от тях съдържа някакво абстрактно представяне на част от цялата БД;
- Същевременно съществува само една концептуална гледна точка - съдържа абстрактно представяне на цялата БД;
- Повечето потребители не се интересуват от цялата БД, а само от една част от нея.

- Вътрешно ниво (internal level):
  - Представя на ниско ниво цялата база данни;
  - Има вътрешна схема, която описва физическата структура на съхраняване на данните;
  - Най-близо до физическата памет;
  - Решава проблемите, свързани с начините за физическото съхраняване на данните;
  - Представя актуалното разположение на данните върху вторичната памет.

Заключение

- Трите схеми съдържат само описание на данни;
- Единствено на физическо ниво съществуват данни;
- В описаната архитектура СУБД трябва да трансформира команда от външна схема до заявка към концептуална схема, след което до вътрешна схема;
- Ако командата е за извличане на данни, то данните трябва да бъдат преформатирани така, че да удовлетворят външната гледна точка на потребителя
- Процесите на трансформиране на заявките и резултатите между отделните нива се наричат кореспонденции (mappings)

Кореспонденции (mappings)

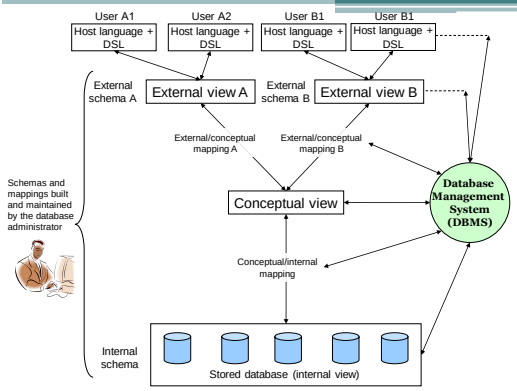
- Кореспонденциите между вътрешното и концептуално ниво (conceptual/internal mapping) определят как данните на концептуално ниво се представят на физическото

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

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.

СУБД

СУБД е софтуерът, който обработва всеки достъп до БД. Най-общо това се извършва по следната схема:

- Потребителите подават заявка за достъп – използвайки някакъв подезик за достъп до данни (напр. SQL);
- СУБД прехваща заявката и я анализира;
- Преглежда външната, концептуалната и вътрешната схема, както и съответните кореспонденции;
- Планира и извършва необходимите операции върху БД, за да изпълни заявката.

Пример:

- Потребител подава заявка за извършване на някаква справка;
- В общия случай компонентите на справката ще се извличат от съответни полета, които са разположени в различни концептуални записи;
- СУБД трябва:
  - да локализира съответните им вътрешни образи;
  - след което ще конфигурира необходимия концептуален запис;
  - едва тогава може да създаде външния запис (справката), с което ще удовлетвори потребителската заявка.

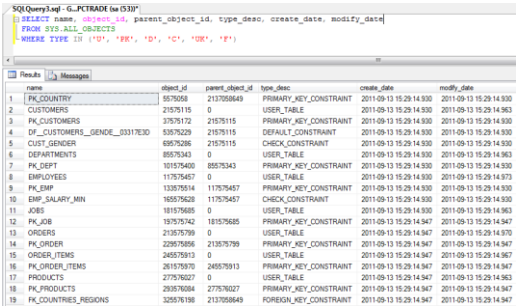
СУБД

Нека проследим функциите на СУБД по-детайлно. Те трябва да поддържат поне следните:

- Управление на речника за данни

СУБД съхранява данни за всички обекти, които са създадени в нея, както и за техните взаимоотношения. Тя използва тези данни, за да намира обектите, които са субект на команди от потребители. По този начин предоставя ниво на абстракция, което прави приложенията независими от структурата на съхранение на данните.

Речник на данните – MSSQL 2008



```
SELECT name, object_id, parent_object_id, type_desc, create_date, modify_date
FROM SYS.ALL_OBJECTS
WHERE TYPE IN ('U', 'PK', 'D', 'C', 'UK', 'F')
```

Results | Messages

	name	object_id	parent_object_id	type_desc	create_date	modify_date
1	PK_COUNTRY	5575058	2137058649	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
2	CUSTOMERS	21575115	0	USER_TABLE	2011-09-13 15:29:14.930	2011-09-13 15:29:14.963
3	PK_CUSTOMERS	37575172	21575115	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
4	DF__CUSTOMERS__GENDE__03317E3D	53575229	21575115	DEFAULT_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
5	CUST_GENDER	69575286	21575115	CHECK_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
6	DEPARTMENTS	85575343	0	USER_TABLE	2011-09-13 15:29:14.930	2011-09-13 15:29:14.963
7	PK_DEPT	101575400	85575343	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
8	EMPLOYEES	117575457	0	USER_TABLE	2011-09-13 15:29:14.930	2011-09-13 15:29:14.973
9	PK_EMP	133575514	117575457	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
10	EMP_SALARY_MIN	165575628	117575457	CHECK_CONSTRAINT	2011-09-13 15:29:14.930	2011-09-13 15:29:14.930
11	JOBS	181575685	0	USER_TABLE	2011-09-13 15:29:14.930	2011-09-13 15:29:14.963
12	PK_JOB	197575742	181575685	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.947	2011-09-13 15:29:14.947
13	ORDERS	213575799	0	USER_TABLE	2011-09-13 15:29:14.947	2011-09-13 15:29:14.970
14	PK_ORDER	229575856	213575799	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.947	2011-09-13 15:29:14.947
15	ORDER_ITEMS	245575913	0	USER_TABLE	2011-09-13 15:29:14.947	2011-09-13 15:29:14.967
16	PK_ORDER_ITEMS	261575970	245575913	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.947	2011-09-13 15:29:14.947
17	PRODUCTS	277576027	0	USER_TABLE	2011-09-13 15:29:14.947	2011-09-13 15:29:14.963
18	PK_PRODUCTS	293576084	277576027	PRIMARY_KEY_CONSTRAINT	2011-09-13 15:29:14.947	2011-09-13 15:29:14.947
19	FK_COUNTRIES_REGIONS	325576198	2137058649	FOREIGN_KEY_CONSTRAINT	2011-09-13 15:29:14.947	2011-09-13 15:29:14.947

СУБД

- Сигурност и интегритет на данните

СУБД трябва да следи правата на потребителите за достъп до обектите в нея и дали командите на потребителите не нарушават наложени правила за сигурност и цялостност – и евентуално да отхвърля подобни команди.

СУБД

- Контрол на едновременния достъп (конкурентност)

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.

СУБД

- Архивиране и възстановяване

СУБД предоставя възможност за реализиране на стратегия за архивиране и съответно възстановяване от архивни копия.

- Управление на интегритета на данните

СУБД предоставя правила, чрез които може да се контролира интегритета на данните, по този начин намалявайки излишеството.

СУБД

- Езици за достъп до СУБД, API и комуникационни интерфейси

СУБД предоставя достъп чрез език за заявки. Той се състои от следните компоненти:

- ✓ DDL (Data Definition Language) – предоставя оператори, с които се създават и променят структурите, съхраняващи данните;
- ✓ DML (Data Manipulation Language) – предоставя оператори за манипулиране и извличане на данни;
- ✓ DCL (Data Control Language) – предоставя оператори за авторизация на потребителския достъп.

СУБД

- Езици за достъп до СУБД, API и комуникационни интерфейси

Application Programming Interfaces – предоставят достъп до базата данни на езици от високо ниво – Cobol, C, Pascal, VisualBasic, Java и др.

Комуникационните интерфейси реализират достъпа до базата данни от крайните потребители в компютърна мрежа.

СУБД

- Ефективност

Функциите на базата данни трябва да се изпълняват възможно най-ефективно и с минимално заемане на системни ресурси.

Модели СУБД

Съществуват различни модели СУБД – основните са следните:

1. Файлови системи (1960–1980)

Предполагат използването на ***flat files***, съдържащи прост неструктуриран текст.

По дефиниция файлове с comma-separated values (CSV) данни са структурирани, защото те съдържат ";" като разделител на стойностите, но също се приемат за flat file.

Но старите бази данни са използвали flat files с огромни низове, без разделители и без символи за нов ред, в които данните се търсят по позиция във файла, така че сравнени с тях CSV файловете не са съвсем flat, защото все пак имат някаква структура.

Модели СУБД

1. Файлови системи (1960–1980)

Полезни, когато:
- Стойностите са относително прости и малки по размер;
- Стойностите не се променят често;
- Е нужно данните да се редактират с прост текстов редактор.

Не особено полезни, когато:
- Трябва да бъде приложено сложно търсене в данните;
- Данните се променят често;
- Данните не трябва да са лесно достъпни за преглед и манипулация;
- Данните имат йерархичен вид.

Модели СУБД

2. Електронни таблици

Представят данни в редове и колони. Позволяват писане на формули, правене на графики, лесен импорт и експорт в текстов и други формати.

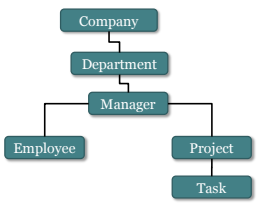
Полезни, когато:
- Данните са удобни за представяне по естествен начин в табличен вид;
- Трябва да се визуализират графики;
- Потребителите умеят да работят с тях и искат да експериментират с данните в собствени локални копия.

Не особено полезни, когато:
- Трябва да се представят сложни взаимоотношения между стойности в различни таблици;
- Трябва да се изпълнят сложни изчисления;
- Е нужно валидиране на данните;
- Трябва да бъдат изпълнени сложни заявки;
- Трябва да се актуализират големи обеми от данни.

Модели СУБД

3. Йерархичен модел (1970–1990)

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.

Модели СУБД

3. Йерархичен модел (1970–1990)

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

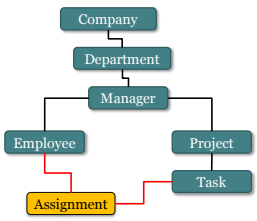
Полезен, когато:
- Данните са йерархични;
- Трябва да се изпълняват операции, които се възползват по-добре от йерархичната структура.

Не особено полезен, когато:
- Данните не са йерархични по природа и се налагат трансформации, за да бъдат представени като такива;
- Трябва да се правят сложни изчисления или търсения, които не използват йерархичната структура;
- Е нужно сложно валидиране на данните;
- Трябва да се актуализират големи обеми от данни.

Модели СУБД

4. Мрежов модел (1970–1990)

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 няколко служители.

Модели СУБД

4. Мрежов модел (1970–1990)

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

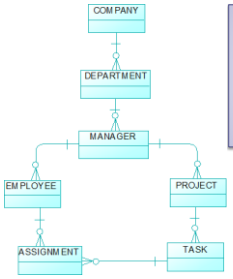
Полезен, когато:
- Структурата на данните е мрежова или йерархична;
- Е нужно изчисление, специфично за тези структури – намиране на най-кратък път между възли и др.;
- Не са необходими изпълнения на сложни заявки към данните.

Не особено полезен, когато:
- Данните не са в мрежов вид;
- Е нужна валидация;
- Трябва да бъдат изпълнявани заявки;
- Трябва множество потребители често да променят данни без да си пречат взаимно.

Модели СУБД

5. Релационен модел (1980–досега)

Представя базата данни като колекция от таблици.



Усъвършенства ограниченията на йерархичната структура без да я отхвърля. Данните могат да бъдат достъпени 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.

Модели СУБД

5. Релационен модел (1980–досега)

Представя базата данни като колекция от таблици.

Полезен, когато:

- Са нужни изпълнения на сложни заявки със съединения на данни от различни таблици;
- Е нужно валидиране на данните;
- Е нужна гъвкавост при създаване на нови заявки, които не са били планувани по време на дизайна на системата.

Не особено полезен, когато:

- Трябва да се представят данни в йерархичен или вид на мрежа, защото ефективността при обработката им може да е по-ниска в сравнение с други модели бази от данни, които са предназначени за такива структури.

Модели СУБД

6. Обектен модел (1990–досега)

Дефинира базата данни в термините на обекти, техни характеристики и операции. Класовете на обектите са организирани в йерархии или ациклични графове.

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 (със заменящи ги таблица) като ги заменя с колекции, включени в класовете.
Назначението на задачите на служители и мениджъри е решено с използването на колекции в тези два класа.

Модели СУБД

6. Обектен модел (1990–досега)

Полезен, когато:
- Програмната среда и архитектурата изискват работа с обекти;
- Не са нужни сложни заявки към данните, които биха забавили работата им.

Не особено полезен, когато:
- Трябва да се изпълняват сложни заявки, които биха били по-ефективни в релационна база;
- Не използвате обектно-ориентиран език;
- Е нужна валидация, която обектната база не предоставя.

Модели СУБД

7. Обектно-релационен модел (1990–досега)

Релационна база данни, поддържаща и обектно-ориентирани структури и механизми.
Включени са възможностите на обектният модел в релационна система. Много релационни системи позволяват съхранение на двочни обекти с ограничени възможности за кодиране на методи.
Предимството е, че потенциално големи обекти могат да бъдат съхранявани в поле от таблица.

Полезен, когато:
- Програмната среда и архитектурата изискват работа с обекти;
- Трябва да се изпълняват сложни релационни заявки;
- Е нужна валидация в релационен стил.

Не особено полезен, когато:
- Не използваме обектно-ориентиран език.

Модели СУБД

8. 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

8. XML

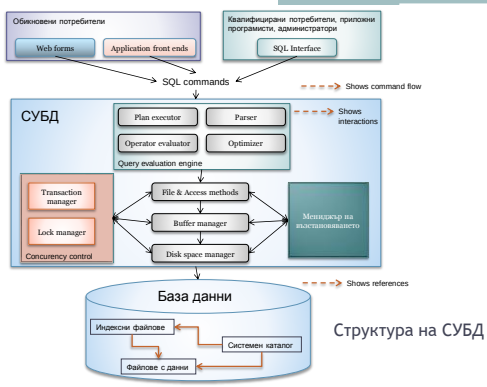
Полезен, когато:
- Данните са йерархични;
- Имаме на разположение XML инструменти, предлагащи нужните ни функции;
- Е нужно валидиране на файл спрямо XML схема;
- Са нужни импорт/експорт на данни за системи, разбиращи XML.

Не особено полезен, когато:
- Се използват не йерархични данни;
- Е нужна по-сложна валидация от предоставяната от XML схемата;
- Трябва да се изпълняват по-скоро релационни, отколкото йерархични заявки;
- Базата данни е много голяма и презаписването на цял файл за промяна на малка част от данните е твърде тежка операция;
- Е нужно различни потребители да променят данните без да си пречат взаимно.

Модели СУБД

9. Други

- NoSQL – обхваща множество от различни технологии за бази данни, разработени в отговор на нуждата от по-големи обеми информация, които да бъдат съхранявани:
  - Документно-ориентирани – създадени да работят с документно-ориентирани приложения, позволяващи на потребителите да отварят „документ“, който може да е писмо, клип и др.
  - Graph хранилища – използвани за съхранение на данни от структури от тип граф, като напр. социални връзки.
  - Key-value хранилища – на-простия вид NoSQL. Всяка единица се съхранява с ключ и стойност.
- Deductive – даващи възможност за дедукции, базирани на правила и факти, съдържащи се в базата данни;
- Dimensional - (multi-dimensional database) представя различни аспекти от данните като измерения.
- И др.



Структура на СУБД

Релационни системи

Повечето от БД, разработени в последните години, са релационни - релационният модел е една от най-съществените разработки в цялата история на БД.

Какво означава една система да е релационна?

Тук ще дадем неформален и първоначален отговор на този въпрос.

Накратко, една релационна система е такава система, в която:

- данните се съхраняват в двумерни таблици, наречени в теорията релации
- операторите, с които потребителят разполага, генерират нови таблици от съществуващите

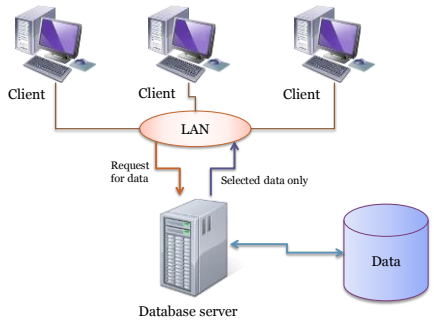
Системите за релационни бази данни използват математическата теория на множествата, за да организират ефективно данните.

Клиент/Сървър архитектура

От една по-висока (по-абстрактна) гледна точка СУБД може да се разглежда като архитектура, състояща се от две части:

- сървър (също backend);
- множество от клиенти.

Двуслойна архитектура



Многослойна архитектура