Система за управление на бази от данни (СУБД)

Определение: Мощен инструмент за създаване и управление на големи обеми от данни ефективно. Този инструмент позволява базата от данни да съществува за дълги периоди от време безопасно.
- ефективно – т.е. да се извършва обработката по-бързо отколкото чрез файловата система, макар че и с нея могат да се управляват големи обеми от данни. Целта е да не се налага потребителят да чака.
- СУБД защитава данните си
- хардуерът и софтуерът се променя/обновява, но СУБД остава във времето

Възможности на СУБД:

- Съхранява устойчиво данните
- Програмен интерфейс – интерактивен и конвенционален
- Управление на транзакциите
  Може да се каже, че СУБД представлява операционна система, а транзакциите са процесите в тази ОС, но има малки разлики като свойствата по-долу, но освен това транзакциите са предсказуеми. Не може да се напише ОС, която ефективно да управлява процесите. Не може да се определи, след колко време ще приключи даден алгоритъм и точно колко ресурси ще отнеме.
  Транзакциите, написани на SQL, могат да се оценят – какви ресурси са необходими и за колко време ще се изпълнят.
    - Изолация – всяка транзакция се изпълнява независимо от изпълнението на другите транзакции
    - Атомарност – всичко или нищо. Ако транзакцията аварира/грешка, нейните резултати се изчистват. Не остават следи от изпълнението ѝ.
    - Устойчивост на резултатите – резултатите се записват завинаги, докато не бъдат променени от друга транзакция, например.

Еволюция на системните бази от данни

Определение: Базата от данни е набор от информация, който съществува за дълъг период от време, често за много години. Също така, базата от данни е набор от данни, които се управляват от СУБД.

Какво се очаква от едно СУБД?

1. Позволява на потребителите да създават нови бази от данни и специфицират тяхната схема (логическата структура на данните), като се използва специализиран език, наречен **език за дефиниция/описание на данни (Data-Definition Language DDL)**. /На него се пишат схемите на базите от данни, за да се създават нови бази от данни./
2. Да могат потребителите да търсят в данните. Да изменят данните, като за това се използва специализиран език за заявки или **език за манипулация на данни (Data-Manipulation Language DML)**. /Този език съдържа в себе си една част, с която само се търси в данните (език за заявки), както и друга част, която се занимава с обновяването на базата от данни. Първата част е по-сложна, а втората - по-проста./
3. Да съхранява големи обеми от данни за дълги периоди от време, като тези данни се съхраняват сигурно/защитено от инциденти или от несакнциониран достъп. Освен това,

достъпът е ефективен до тези дани както за заявки, така и за измемения на базата от данни.

4. Да контролира достъпа до данни при множество едновременни потребители, като работата на един потребител да не оказва влияние върху работата на друг и заедно с това да не се развалят данните. /Пример: В банкова сметка е написано, че един човек има 1000лв. Имаме 2-ма потребители: първият иска да добави 100лв а вторият иска да извади 200лв от тази сметка. Ако няма контрол ще се развалят данните (вж. Тетрадката)/.

Ранни системи за управление на бази от данни

- Системи за резервация на самолетни билети – това са и първите бази от данни. За цяла Европа е имало 1 сървър, който е бил в Париж, и всички терминали са се свързвали с тази голяма машина през телефонна линия. Скоростта на модема е била около 100 бита/сек. В САЩ, в Атланта, е имало друг сървър, който е бил световно обслужващ. Транзакциите са били малки (полет х, а се връщат свободните места).
- Банкови системи – каквато е банковата система, такава е и икономиката.
- Корпоративни записи – информация за заплати, отчетност

Ако торите една земя и влагате все повече и по-качествена тор има една граница на насищане на тази инвестиция, ако продължите да слагате тор, ще изгорите земята. Информационните системи са единствените, в които човек може да влага постоянно и да получава резултати.

Релационен модел на данни

Йерархичен модел на данни – данните са във вид на дървета. Първите, които създават такова СУБД са IBM за проекта „Аполо“.

В IBM, Edgar Codd е разработил релационния модел на данни, който се оказва изключително ефективен. Нарича се релационен, защото се основава на релациите в теория на множествата. Освен стандартните операции, той добавя и две допълнителни.

Данните се представят във вид на таблици вместо дървета или мрежи. Всички редове, освен първия, се наричат кортеж – tuple (row). Първият ред съдържа наименование на колоната – attribute. Атрибутите носят някаква семантика за смисъла на колоната, а в самата колона данните са еднотипни.

Когато се появява въпросът, как става търсенето, се създава езика sequel (SQL).
Създава се проектът Gamma. Gamma – 0 започва с език, който е релационна алгебра. След това се създават езици от по-високо ниво: Gamma – 1, …, **Gamma – n.** Gamma-n езикът се нарича **SeQueL.**

DB2 е комерсиалното СУБД до ден днешен, след System-R.

Релационният модел на данни е в основата на всичко компютърни системи, които ползваме по един или друг начин в нашето ежедневие.

SQL езикът е сравнително прост. Първо казваме, откъде четем FROM, след това задаваме условието WHERE. Така ще намерят всички записи с това условие, но ще се изведе само избраното SELECT.

Еволюция на системите бази от данни

Едното управление е надолу – все по-малки системи. Появяват се мини машини. Фирмата Digital разработва компютъра PDP11 за управление в реално време на атомни електроцентрали. Оказва се, че този миникомпютър може да се използва и за управление на информацията в един отдел. Oracle са едни от първите, които са пуснали СУБД на персонални компютри.

Другото направление е нагоре – все по-големи системи, в които да се работи и с третичната памет. Задачата на СУБД-то се насочва не само към вторичната, но и към третичната памет. Налагат се и паралелните изчисления за реализирането на достъпа до данни в реално време.

Към еволюцията попада и развитието на многослойните архитектури. Появяват се по-мощни файлови сървъри. Към него чрез локална мрежа се връзват персонални компютри, които имат по-слаби устройства за съхраняване. Използвало се е за различни отдели, като на 1 сървър са се закачали около 16 PC-та. Тогава се стига до идеята за появата на клиент-сървър архитектурата. Изпълнението на заявките се случва само върху файловия сървър, където е и съхранението на данните.

Важен елемент са мултимедийните данни. Те генерират големи обеми от данни. СУБД-тата поддържат големите мултимедийни обеми от данни.

Интеграцията на информацията е важен, съществен елемент. Имаме наследени бази от данни, които са просъществували от порядъка на 50-60 години са в йерархичен модел на данни. Когато се направи една база от данни се разработват куп приложения, които работят с нея, и обикновено тя не може да бъде унищожена. Става въпрос за големите фирми. В България обикновено няма наследени бази от данни. Големите фирми не унищожават предишните бази от данни legacy databases, но възниква проблемът за интеграция на информацията (от различни бази от данни да бъдат представени единно базите от данни). Това става в складовете за данни data warehouses. Там се събират всички данни. Анализът на данните се извършва със специализирани средства data mining.

Структура на СУБД

User/application – те пускат някакви заявки. Те могат да са за търсене в базата от данни, могат да са и някакви обновявания на базата от данни. По-общо казано, те пускат заявки на SQL.

Query compiler – извършва синтактичен анализ, дали е вярно написана заявката,

семантичен анализ, който се състои в това дали има такава таблица, като е казано, дали в тази таблица наистина им такива колони, колоните дали наистина са от типа, използван в заявката и подобни проверки. Използват се метаданни (описание на данните, данни, които описват други данни). Ако нещо не е наред, връща на потребителя отговор, че нещата не са ОК. Ако е добре сегенерира query plan. Понякога този план може да се оптимизира, когато заявката ще се

използва много пъти. Процесът на оптимизация може да се окаже по-тежък от самото изпълнение на заявката, но ако ще има много заявки се прави.

Планът на заявките представлява релационна алгебра, разписана във вид на разпознато дърво.

След това заявките се интерпретират. Интерпретират се чрез индекси, файлове и записи**.** Дискът и ОП са организирани в блокове. Файлът представлява набор от блокове с фиксирана дължина, независимо дали е индексен файл или файл с данни. В блоковете се разполагат записите. Файловете могат да бъдат балансирани дървета, hash таблици. **Execution engine** се справя с всички тези файлове и записи.

Index/file/record manager -Реализация на различни методи на достъп към такива файлови записи.

СУБД ефективно използва буфери - **Buffer manager.** Четенето от дисак се извършва с този мениджър на буферите, не се чете пряко.

Storage manager – чете и пише данни от диска

Всяка една заявка е транзакция, явно или неявно зададена от потребителя. С пускането на заявките се отива и към **Transaction manager –** трябва да гарантира, че ресуристе, които изисква транзакцията, са налични. Мениджърът на транзакциите се обръща към модула за контрол на конкурентността, който проверява дали исканите ресурси са свободни. Той проверява, кои ресурси са свободни и кои са заключени.(**Concurrency control).** От друга страна има още един модул, който се нарича Журнал и възстановяване **Logging and recovery.** Всяко изменение на БД се регистрира в така наречените журнални страници. Това ни гарантира, че ако транзакцията аварира, ще знаем как да възстановим обратно съдържанието на базата от данни. Ще ликвидира нейното изпълнение, използвайки тези страници, които са организирани също в блокове и се зареждат в буферите. Остана само **DDL Compiler.** Администраторът на базата от данни е човекът, който изменя дадена база от данни. Промяната на схемата на логическата структура на базата от данни се прави рядко, но се прави от много подготвени хора. На потребителите не се дават такива права. Това са двете съществени роли в базите от данни.

ACID Properties of Transactions

Буквата A идва от атомарност, всичко или нищо при изпълнението на транзакциите.

C означава consistency, т.е. цялостност на БД. Тя се реализира с т. Нар. Ограничения за цялостност. Една сметка, например, не може да бъде с отрицателен баланс. Транзакциите се очаква да запазват тези ограничения за цялостност. Структурно могат да се задават ограничения с помощта на структурите в модела. Една таблица, например, може да се намира много към един с друга. Структурно можем да представим ограничения за цялостност от реалния свят.

I – означава изолация isolation. Означава, че транзакциите се изпълняват все едно че нма други транзакции, които да се изпълняват по същото време

D durability устойчивост на резултатите. Като завърши транзакция резултатите се записват в БД устойчиво докато друга транзакция не ги промени

Outline of DB System Studies

Какво ще изучаваме:

Проектиране на бази от данни Design of databases – предимно на лекции. Как се разработват полезни бази от данни? Какви видове информация се включват в базата от данни? Как информацията се структурира? Какви предположения за типовете данни и стойностите на тези елементи данни могат да бъдат направени? И как тези елементи се свързват помежду си?

Програмиране на базата от данни Database programming – на упражнения. Как да изразяваме заявки, операции с базата от данни? Как да използваме и други такива способности на БД, като описание на транзакции, ограничения за цялосност? Как могат да се извикват тези функции в базата от данни през СУБД?

Реализация на система база от данни Database system implementation – Как се обработват заявки, транзакции? Как се организира паметта за ефективен достъп? Няма да го учим това.