Увод

Добра работна дефиниция за бази данни е: "База данни е един вид колекция от организирани факти" Тази дефиниция покрива много голяма площ от реалния свят. Всеки от известните носители за съхранение представлява база данни. Проблемът е, че някои носители бази данни не предоставят никакви вградени средства за автоматично сортиране, отпечатване или управление на данните. За да стане възможно това е необходима система за управление на бази данни. Системата за управление на бази данни представлява набор от инструменти за управление на данни в определен формат. Има едно понятие, което всички системи за управление на бази данни включват — машината за бази данни. Машината за бази данни (database engine) е приложението, като например Microsoft Access или Oracle, което обработва данните в базата данни. Тя е сърцето на системата за управление на бази данни. Обикновено тази система е или обектно-ориентирана, или релационна. Обектно-ориентирана система, която добива популярност все още не е налична за повечето настолни PC. Базите данни осигуряват невидимата, но основна, същинска функционалност на съхраняването на данни и тяхната манипулация. По-голямата част са релационни, много използват SQL като основен език, и при повечето достъпът се осъществява чрез ODBC. Релационната база данни е съставена от таблици. Всяка таблица съдържа редове с информация за всеки съхранен в нея обект. Редът съдържа данни за един отделен обект и е съставен от колони с информация, която описва един този обект. Всеки файл на базата данни може да съдържа множество таблици и всяка таблица може да съдържа множество колони. Една или множество таблици могат да бъдат достъпени и чрез изглед (view). Изгледът е логическо подмножество на таблица или комбинация от таблици. Той дефинира различен начин за достъпване, или разглеждане, на данните от таблицата, но самият той не съдържа данни. Позволява да се създават по-нататъшни логически връзки между таблиците, без да създава второ копие на данните.

Важността и необходимостта от създаване на система за управление на бази данни произхожда от факта, че в съвременния свят информацията нараства лавинообразно, като системите за управление на бази данни позволяват тяхното сортиране, анализ, осъвременяване и използване за различни цели.

4.1. Обзор на съществуващи бази данни

Релационният модел на организация на данните възниква в края на 60-те години на

базата на съществуващите системи за управление на бази от данни. В по-голямата си част въпросните системи са възниквали като реализация на оригинални идеи, насочени към решаването на строго специфичен проблем, като впоследствие са били разширявани. В процеса на разширяване обаче първоначалната организация на данните често се е оказвала неефективна, а понякога и напълно безполезна, в резултат на което системата е ставала ненужно сложна и тромава. Релационният модел е първата сериозна крачка по посока на опростяване и универсализация на средствата за управление на базата от данни. Днес той е безспорен лидер при организацията на данните и предлага мощни и същевременно прости решения на широк кръг задачи от икономически, научен и приложен характер. Релационно организираните бази от данни се използват навсякъде, където се изисква съхраняване, актуализация, търсене и сложна обработка на данни, достъпни за един или повече потребители едновременно.

През последното десетилетие обемът на съхраняваните от отделните организации данни нарасна лавинообразно, а общият обем на вложените от компаниите средства с цел модернизация на бизнеса надхвърли един трилион долара. Тези мащабни инвестиции са продиктувани от схващането, че в съвременния бизнес просперират тези фирми, които умеят да анализират, синтезират и ползват натрупаната информация за вземане на правилни и навременни бизнес решения. Ето защо основните усилия днес се насочват към категоризиране и отсяване на ценната информация.

Съвсем спокойно бихме могли да заявим, че днес повечето организации разполагат с достатъчно данни, за да вземат сериозни бизнес решения. За съжаление обаче в по-голямата си част тази огромна по обем информация се пази в релационни бази от данни или в наследени стари системи (мейнфрейми, работни станции и сървъри) и рядко се използва за анализ и вземане на критически бизнес решения. Основният проблем днес се крие не толкова в количеството на данните, колкото в тяхната организация и представяне. Именно във връзка с нарастването на обема на данните за пръв път проличаха някои от сериозните недостатъци на релационния модел.

Класическата структура на релационната база от данни е насочена към извършване на транзакции в реално време OLTP (On Line Transaction Processing) и е оптимизирана за съхраняване на огромни количества данни за транзакции. Във връзка с нарастването обема на съхраняваната информация напоследък започнаха да се налагат друг вид релационни модели — *складовете от данни* (*data warehouse*), чиято структура улеснява анализирането на данните. Докато OLTP системите са предназначени за извличане на информация вътре в системата за управление на релационни бази от данни (RDBMS — Relational Database Management System), складовете от данни предполагат главно външен достъп.

Несъмнено релационният модел е бил, е и ще продължи да бъде най-подходящата организация за съхранение, актуализация и извличане на конкретни данни от огромни по обем бази. Същевременно обаче, дори и организиран като склад от данни, той не позволява бързо и лесно структуриране, анализиране и отсяване на съществената информация. Именно тук на помощ идва оперативната аналитична обработка в реално време OLAP (On-Line Analytical Processing), разглеждана като основно стратегическо направление в развитието на информационните системи. Използването на OLAP съвместно със складове от данни, предлага мощен и гъвкав метод, съчетаващ предимствата на двата подхода.

Какво е OLAP? Може би най-точното определение е даденото от *съвета по OLAP* — консорциум от производители на OLAP-продукти.

OLAP определение: "*Оперативната аналитична обработка в реално време* OLAP е категория от областта на програмното осигуряване, предоставяща възможност на аналитиците, мениджърите и ръководителите да проникнат в дълбочината на данните, използвайки бърз еднообразен оперативен достъп към най-широк спектър от всевъзможни представяния на информацията, получени от първичните данни с цел отразяване на различни аспекти на реалната дейност на предприятието. Функционалността на OLAP се дължи на динамичния многомерен анализ на консолидирани данни за предприятието, насочени към поддръжка на следните аналитични и навигационни видове дейности на крайния потребител:

- Изчисления и моделиране, приложени към измерения и/или техни конкретни елементи, използващи информация за йерархиите;
- Анализ на временните тенденции на различните показатели;
- Извличане на сечения на многомерните представяния с цел визуализация на екрана;
- Преход към по-дълбоко равнище на детайлизация;
- "Ротация" на многомерните представяния."

Макар и сравнително нова технология OLAP успя бързо да набере скорост и днес на пазара могат да се намерят множество конкурентни продукти на компании като Oracle, Informix, IBM, Pilot Software и др. За съжаление обаче за момента липсва общ стандарт за OLAP средствата и всеки разработчик предлага свое собствено решение със специфична терминология и език за описание на заявките към базата. (При релационните бази от данни нещата са твърдо стандартизирани и практически всички

средства поддържат като език за описание на заявки към базата SQL — Structured Query Language). Разглеждането и сравнението на различните подходи излиза извън рамките на настоящата статия и в по-нататъшното ни изложение ще се спрем по-подробно на конкретна реализация, а именно Oracle Express Server.

Oracle Corporation е сериозна компания, нареждаща се сред IT фирмите непосредствено след Microsoft по финансови приходи, с големи възможности за налагане на нови технологии, безспорен лидер на пазара на средства за управление и поддръжка на релационни бази данни. Самият продукт Oracle Express Server вече е излязал от детската си възраст (на пазара се предлага шестата поред версия на продукта, която именно ще бъде разгледана по-долу) и е успял да се утвърди като водещо средство за разработка на OLAP приложения.

Express Server е високопроизводителен многопотребителски OLAP сървър за манипулация с многомерни структури от данни. Всеки елемент на данните се отнася към индивидуална клетка на многомерен куб, чиито оси се наричат *измерения*. Сървърът е силно оптимизиран за бързо изпълнение на произволни нерегламентирани заявки за търсене по произволно измерение, както и за разнообразни преобразувания на изходните данни, зададени чрез формули, в реално време. Обработваната информация може да се съхранява в многомерни структури на OLAP сървъра, което води до минимално време за достъп, практически независещо от вида и обема на информацията; или пък да се зареждат от релационна или друга база от данни в реално време (продуктът се предлага стандартно с богата колекция от native и ODBC драйвери). Оптимални резултати се постигат при хибриден вариант.

Ще разгледаме някои ключови понятия в Oracle Express Server: измерения, променливи, отношения, формули, модели и йерархии, като за всяко понятие ще посочваме съответния му аналог в релационния модел.

Измеренията: Измерението (dimension) е ключово понятие в Express и обикновено представя реални физически обекти, процеси и явления, за които ще се съхранява информация. Аналог на измерението в релационния модел е първичният ключ (primary key). Всяко измерение се характеризира с тип и представлява крайно дискретно едномерно множество от обекти (стойности). Допустими типове са text (низ до 32Kb), ID (низ до 8 символа, оптимизиран за бързо търсене), integer, conjoint (за съвместяване на размерности) и времевите размерности: day, week, month, quarter, year. В Express Server не съществува никакво ограничение относно броя на дефинираните измерения, като броят на стойностите за конкретно измерение не бива да надвишава 268 милиона.

Променливите: Под променлива (variable) в Express се разбира многомерен куб или масив, в който се съхраняват конкретни стойности. Осите на куба се избират измежду вече дефинираните в базата измерения, като броят им не може да надвишава 32. Променливата може да бъде и безразмерна, т.е. да представя единствена стойност, подобно на класическите променливи в математиката. В релационния модел на променливата съответства стълб от таблица, неявяващ се първичен или вторичен ключ.

На всяка допустима комбинация от стойности на измеренията на дадена променлива съответства точно една клетка от многомерния куб, в която може да бъде записана точно една стойност, като броят на клетките в конкретен многомерен куб не може да надвишава 2^{63}. При дефинирането на променлива освен измеренията, по които ще се измерва, се задава и тип на стойностите, които ще приемат клетките от съответния й куб. Типът е общ за всички клетки на куба и може да бъде: decimal (плаваща запетая), integer (целочислен), shortdecimal, shortinteger, id, text, boolean и date. Забележете, че множествата от допустими типове за измеренията и за променливите се различават.

Отношенията: Отношенията (relations) дефинират релации между измерения от вида един-към-един или един-към-много и са еквивалентни на отношенията между първичен (primary key) и вторичен (foreign key) ключ в релационния модел. Така например бихме могли да дефинираме отношение между отдел на дадено предприятие и град, както и между град и административна област и др. На базата на отношенията в Express могат да се изграждат йерархии, дървовидни и мрежови структури. Отношенията се използват за избор на стойности на измеренията. Така например, въз основа на дефинираното отношение между отдел и град, бихме могли да разглеждаме отделите на дадена фирма само за конкретен град (градове). Други важни приложения на отношенията е изграждането на йерархии и агрегирането на данните.

Йерархиите: В многомерния свят на Express Server измеренията често биват рекурсивни и йерархични. Например бихме могли да създадем текстово измерение KLIENT съдържащо: София, Русе, Търговище, Велико Търново и Ловеч. Прибавяйки към KLIENT стойностите: София-област, Русенска област и Ловешка област, бихме могли в рамките на същото измерение да изградим проста йерархия между град и административна област. В рамките на едно и също измерение бихме могли да изградим повече от една йерархия върху същите данни, както и йерархии между елементи на различни измерения. Няма ограничение за максималния брой нива в йерархиите, както и за максималния брой йерархии в рамките на едно измерение.

Агрегиране на данните: Агрегирането на данните е една от най-ценните възможности, предлагана от OLAP технологията, позволяваща да се извлича най-разнообразна информация въз основа на първичните данни. С помощта на отношения между измерения се изграждат различни йерархични или мрежови структури, описващи различни пътища на консолидация на данните. Така например, използвайки отношенията между отдел и град, бихме могли да разглеждаме сумата от приходите по градове. Използвайки отношенията между град и област пък бихме могли да видим разпределението на съответните печалби по области. При това в процеса на консолидация бихме могли да преминем към нивото административна област директно от най-ниското ниво отдел, без да е необходимо да преминаваме по целия път на агрегиране. Бихме могли да напишем свои собствени функции и формули за агрегиране, или пък да използваме предлаганите от Express специално за целта: ANY, AVERAGE, COUNT, EVERY, LARGEST, NONE, SMALLEST, STDDEV, TALLY, TCONVERT (за манипулация с времеви измерения) и TOTAL (обща стойност).

Времето: Express предлага изключително гъвкавата манипулацията с времето. Между времевите измерения, независимо от типа им day, week, month, quarter или year, съществува неявна връзка (implicit relation) и потребителят е освободен от грижата по явното изграждане на съответно отношение. Възможно е дефиниране на времеви измерения с нестандартна размерност, например 3 дни, 2 седмици и други, като Express автоматично ще създаде съответните неявни връзки с останалите времеви измерения. Началото на година, месец, седмица може да се предефинира за конкретно измерение. Express предлага множество времеви функции, позволяващи намиране на предишни или следващи периоди от време, преобразуване, агрегиране и разбиване на периоди. Така например, разполагайки с данни за движението на продажбите по месеци, бихме могли да ги разбием и да изчислим вероятното им движение по седмици.

Изрази и формули: Изразите в Express представляват комбинация от променливи, константи, обръщения към срезове на кубове (QDR — Qualified Data Reference), функции и оператори, и подобно на променливите се характеризират с измерения и тип. Формулите са еквивалент на представянията (view) в релационния модел. Под формула в Express се разбира специален вид променлива, чиято стойност се изчислява динамично въз основа на някакъв израз в момента на обръщение към нея. В базата се съхранява самият израз, но не и резултатите от неговото пресмятане.

Как Express пресмята стойността на изрази, съдържащи променливи или QDR? Всяко измерение може да се използва от неограничен брой променливи като в такъв случай се поделя между тях. Така например бихме могли да дефинираме времево измерение MESEC, използвано едновременно от едномерните променливи INFLACIJA и LIHWEN_PROCENT и двумерната SREDEN_KURS_MESEC (измервана и по

id-измерението KOD_WALUTA) и винаги да бъдем сигурни, че трите променливи се измерват с едни и същи месеци. При добавяне, премахване или преименуване на стойности на измеренията, това автоматично ще се отрази на всички измервани по тях променливи. Това позволява на Express коректно да изчислява изрази от вида на:

INFLACIJA - LIHWEN_PROCENT

Резултатът от горната операция е едномерен куб, измерван по MESEC. В случай че за някой месец от MESEC една от двете променливи има стойност N/A (т.е. липсва), резултатът за съответния месец ще бъде NA. Класически пример за многомерна променлива е SALES, измервана по месец, продукт и отдел. Ако дефинираме други две променливи EXPENSE и PROFIT със същите измерения, бихме могли да изчислим обема на чистата печалба като разлика на продажбите и разходите така:

PROFIT = SALES - EXPENSE.

Не е необходимо променливите да имат еднаква структура, за да могат да се прилагат аритметични операции над тях. Така например, ако дефинираме променливи PRICE, измервана по месец и продукт, и UNITS, измервана по месец, продукт и отдел, то бихме могли да изчислим продажбите по формулата:

SALES = PRICE * UNITS

Въпреки че PRICE има едно измерение по-малко, Express автоматично ще изравни измеренията и ще върне верен резултат.

Езикът: Тъй като SQL се оказва непригоден за манипулация над многомерни обекти, в Express Server е реализиран собствен гъвкав процедурен език — Express, поддържащ конструкции за цикъл, за условно и безусловно предаване на управлението, за създаване на процедури и функции на ниво база и др. Същевременно се поддържат практически всички стандартни SQL команди, необходими за достъп, зареждане и динамична манипулация с релационни бази данни. Едно от интересните решения в езика

се отнася до избора на данните, с които се работи. Докато в SQL изборът се задава в WHERE-клауза като част от командата SELECT, в Express изборът е изнесен в отделна команда LIMIT, манипулираща с достъпните стойности на измеренията. Така, ако издадем командите:

> LIMIT MESEC TO JAN97 TO FEB98

> LIMIT DEN TO MESEC

> LIMIT GRAD TO София Русе Ловеч

зададените ограничения ще се отразят върху всички променливи, измервани по DEN, MESEC и GRAD. Това позволява на Express да намали необходимото време за достъп до данните — след като веднъж сме направили своя избор, до издаване на нов LIMIT работим само с избраните данни. Разбира се, в езика е предвидена и възможност за обръщение към специфични данни, без да се налага издаване на команди LIMIT и без изменяне на текущия статус на измеренията. Така например, ако искаме да видим продажбите за декември 2003 и януари 2004г. за софийския отдел на фирмата, бихме могли да издадем командата

> REPORT SALES(MESEC DEC03 JAN04, OTDEL ‘София’)

Горната команда използва QDR (Qualified Data Reference), за да посочи нужните стойности на измеренията.

Моделите: Моделите (models) са едно от най-мощните разширения на езика. Моделът е специален вид подпрограма, опериращa над едно или повече измерения и представлява последователност от взаимнозависими присвоявания, действащи върху променливи или стойности на измерения. Така например, в модел MYMODEL, базиран на GRAD, бихме могли да напишем следното уравнение:

Русенска област = Русе + Търговище

В резултат на изпълнението на MYMODEL с параметър променлива SALES, измервана по MESEC, PRODUCT и GRAD, данните за Русенска област ще бъдат изчислени като сума на съответните данни за Русе и Търговище и записани в SALES. Моделите не са свързани с конкретна променлива и не зависят от броя на измеренията на подадения параметър. Могат да работят върху повече от едно измерение, както и да се наследяват един друг. С тяхна помощ без особени усилия могат да се създават сложни what-if анализи, описващи прогнози за развитието на предприятието, зависещи от наличните данни и от подадени от потребителя параметри.

Връзка с други бази от данни: Express Server се доставя стандартно с богата колекция от ODBC и native драйвери за достъп до релационни и други бази от данни, електронни таблици, текстови файлове и други. Достъпът до данните се улеснява значително при използването на Express Spreadsheet Add-In, Express Administrator и Relational Access Manager.

Express Spreadsheet Add-In осигурява интерфейс за достъп на електронни таблици като Excel до Express бази данни.

Express Administrator е средство за администриране на Express бази от данни. С негова помощ могат да се създават нови и редактират съществуващи Express бази, да се дефинират, редактират и премахват измерения, променливи, формули, модели, програми на ниво база, отношения, множества от стойности (valuesets), да се създават и поддържат йерархии в рамките на едно или няколко измерения и др. Express Administrator предлага изключително лек начин за импортиране и достъп до данни от други бази. За всяка импортирана колона на таблица или представяне (view) от релационната база се посочва съответен обект-приемник (измерение, променлива или отношение) от Express базата. Express Administrator генерира автоматично програма на езика Express, извършваща съответното импортиране, след което я стартира. Програмата се запазва в базата, което позволява нейното модифициране и многократно изпълнение с цел актуализация на данните. Разбира се, опитният потребител би могъл изобщо да се откаже от автоматичното генериране и да състави своя собствена програма, използваща вградените в езика Express SQL конструкции.

Relational Access Manager е специализирано средство за достъп до релационни бази

данни, електронни таблици, текстови файлове и др., осигуряващо на Express приложенията достъп до складове от данни (Data Warehouse), без изрично да изисква съхранение на данните в база данни Express. Relational Access Manager съдържа три компоненти: Relational Access Administrator, описващ структурата на Express данните и начина на достъп до RDBMS; Build Module, създаващ нови или актуализиращ съществуващи Express бази; и Runtime Module, осигуряващ на Express приложенията средства за достъп до RDBMS по време на изпълнение.

Клиентът: Oracle Corp. предлага като отделен продукт високопроизводителна визуална среда за разработка на Express приложения — *Oracle Express Objects*, чийто език Express Basic е синтактично съвместим с Visual Basic. Достатъчно е да изтеглим с мишката произволна променлива, измерение или формула от списъка на достъпните в базата обекти (прозорецът Database Browser) върху прозореца на приложението, за да създадем съответна таблица или графика. Многомерните обекти се визуализират като двумерни. За целта две от измеренията се избират за базови, а стойностите на останалите се появяват в прозореца на приложението като падащи списъци. Над таблицата (графиката) се появява ивица на измеренията (dimension bar), позволяваща на крайния потребител по време на изпълнение на приложението да избира показваните стойности на измеренията, както и да променя реда, в който измеренията се визуализират върху ивицата, чрез просто провлачване с мишката, в частност задавайки нови базови измерения. Изобщо крайният потребител има пълна свобода на действие и по подразбиране може да променя по време на изпълнение на приложението почти всичко — цвят, размери и местоположение на обектите и т.н., стига това да не е забранено изрично.

Oracle Express Analyzer е инструментално средство за анализ, предназначено за крайния потребител. Продуктът изпълнява всички приложения, създадени с Oracle Express Objects и позволява бързо и лесно публикуване на произволни таблици и графики в OLAP Web сайт, разширявайки възможностите за анализиране на произволни данни, разположени в или достижими чрез Express Server.

Освен изброените средства, Oracle Corp. предлага три готови приложения за достъп, изчисления и съвместно използване на информация относно продукти, пазари, канали на разпространение, сценарии, времеви периоди, бюджет (фактически и/или прогнозен) и др.: *Oracle Sales Analyzer*, *Oracle Financial Analyzer* и *Oracle Financial Controller*.

Express и Web: С помощта на *Oracle Express Web Publisher*, add-in за Oracle Express Objects и Oracle Express Analyzer в Web могат да се експортират под формата на

таблици и графики всевъзможни OLAP анализи, базирани на данни, достъпни чрез Express Server. OLAP web сайтът е изцяло обектноориентиран — поддържат се обекти като WebSite, WebBriefing, WebPage, WebGraph (експортира се като Java аплет или VRML — Virtual Reality Modeling Language обект) и WebTable.

В заключение бихме могли да кажем, че OLAP технологията е мощно оръжие в жестоката борба за оцеляване в съвременния бизнес, вдъхващо нов живот на купищата информация в старите релационни бази от данни. Чрез подходящо структуриране, категоризиране и отсяване на ценната информация в реално време, OLAP позволява на ръководителите да вземат оптимални бизнес решения, предоставяйки им възможността да погледнат на процесите и явленията, свързани с развитието на конкретното предприятие, в дълбочина, без да се налага да се отказват от традиционния релационен модел на организация на конкретните данни. С появата на продуктите от фамилията Oracle Express за пръв път стана икономически целесъобразно и административно възможно внедряването на OLAP технологията практически във всяко предприятие — разработчикът ползва евтината Windows NT платформа, а данните достигат до потребителя с помощта на обикновен Web-браузър.

MySQL е система за управление на релационна база данни (БД), която използва Structured Query Language (SQL) - най-популярният език за добавяне, прочитане и обработка на информация в базите данни днес. Системата е с отворен код и използването й се подчинява на лицензa GPL. Първата версия на MySQL се появи през януари 1998 година. Може да се използва с голяма група програмни езици - C, C++, Eiffel, Java , Perl, PHP, Python и Tcl и има версии за Linux , UNIX и Windows .
Системата за управление на релационна база данни (RDBMS) позволява създаването, администрирането и работата с релационни бази данни. Тези БД представляват съвкупности от информационни единици, организирани формално в таблици. Достъпът и промяната на данните се извършва без да е необходимо реорганизирането на таблиците или каквото и да било друго в БД. Едно от най-важните предимства на релационните БД е лекотата с която се създават, четат и изтриват записите, както и лесната разширяемост.

Специалистите в бранша обясняват, че релационната БД се състои от таблици, в които данните са подредени по колони(категории). Всеки ред съдържа уникално смислено съчетание от данни (стойности на колоните). Така таблицата всъщност свързва категориите, тя е "релация" между тях. Данните от различни таблици също могат да са свързани -- така имаме отново релация, но този път между таблиците. По този начин таблиците и връзките между тях спояват данните в едно логическо цяло, и именно то е (релационна) базата данни.. Ако последните няколко изречения не са ви ясни, няма проблеми. За да не ви се смеят "знаещите", трябва само да запомните, че MySQL не е

база данни, а система за управление на база данни. И то не на каква да е БД, а на релационна. Толкова с теорията.

За да стане по-ясен текстът от тук нататък, ще кажем, че базата данни се състои от таблици, всяка от които има колони (указващи каква информация трябва да се съхрани там) и редове (представляващи записите). Всяко парче от данните се поставя в полагащото му се сечение между редовете и колоните - т.е. в съответното поле.

MySQL предлага поддръжка на различни типове данни, но най-вероятно поне на първо време ще са ви необходими четири или пет от тях. Ето описанията на основните, с "x" в скобите е обозначено число, указващо дължината на записа, а квадратни скоби са показани параметри, които не са задължителни.

CHAR (x) - низове с точно определена дължина. Възможната стойност за x е от 1 до 255. Пример :user_id CHAR(10) - очаква се името на модела да е точно 10 символа.

VARCHAR (x) - низове с максимална дължина x. Отново възможните стойности са между 1 и 255. Пример : user_id VARCHAR(10).
INT (x) [Unsigned] - използва се за указване на цели числа между -2147483648 и 2147483647. Ако се използва Unsigned, тогава валидните числа са между 0 и 4294967295. Пример : user_phone INT.

FLOAT [(M,D)] - указва малко дробно число, като M регулира общия брой цифри, от които може да се състои числото, а D ограничава колко от тях може да са зад десетичната запетая.

Ако числото е с повече цифри след запетаята, се прави закръгляне. Пример : suma FLOAT (4,2) - означава, че ако поставите в това поле числа като 2,31 или 22,56, а също и 1, 34 или 1,2, те ще бъдат съхранени правилно. Но ако се опитате да сложите нещо като 32,567, на практика в полето ще се запише 32,57.

DATE - използва се за съхранение на дати, както говори и името. Форматът по

подразбиране е "ГГГГ-ММ-ДД". Пример : born DATE.

TEXT - служи за съхраняване на по-големи низове - от 255 до 65535 символа. При търсене в низовете, съхранявани в такива полета, MySQL няма да прави разлика между малкии големи букви.

BLOB - също като TEXT, с разликата, че търсенето тук взима предвид малките и големи букви.

PRIMARY KEY (първичен ключ на таблицата) е онова поле, което системата използва за да разграничава различните записи. Не може да има два различни реда с една и съща тойност на полето, определено за първичен ключ. Всяка таблица има свой PRIMARY KEY.
UNIQUE също ще ви гарантира уникалността на информацията.

След като релационните бази от данни станаха популярни, възникна нуждата от стандартен език за операции с данни. Отговорът бе SQL (Structured Query Language, Език за структурирани заявки). Постепенно SQL прерасна в многофункционален език за работа с бази данни с управляващи конструкции за: създаване, промяна и изтриване на данни; дефиниране на данни (таблици, колони); защита на достъпа до елементи от бази данни чрез работа с групи и индивидуални потребители; операции за управление на данните като създаване на архивни копия, блоково копиране и актуализация; и, най-важното, обработка на транзакции.

Общата форма на SQL се нарича ANSI-SQL, но всеки производител на СУБД има собствена реализация на SQL. В SQL сървъра на Microsoft, който е една от клиентско/сървърните релационни СУБД, е реализиран “диалект” на SQL с име Transact/SQL, докато SQL на Oracle се нарича PL/SQL. Но фактически всички комерсиални (релационни) продукти за бази данни разбират SQL, въпреки че повечето също имат и собствени специални диалекти. Това означава, че за по-голямата част заявките за базата данни, разработени за използване с точно определена база данни, са преносими от един продукт или инструмент до друг. SQL кодът, написан от програмист използващ настолни (desktop) бази данни, като Microsoft Access, могат да бъдат използвани на система, с инсталирана корпоративна (enterprise) база данни, като Oracle, без съществени промени. Освен това SQL се използва не по-малко от езици като Java, C++ и други поради възможностите, които предоставя в областта на работата с данни и управлението на бази данни.

PL/SQL: Един от най-често използваните диалекти на SQL е PL/SQL. Той се използва в управлението и работата на/с бази данни от Oracle корпорацията и е процедурен език, с разширения към SQL. Основната му цел е да комбинира език за бази данни с процедурен програмен език. Основната част в PL/SQL се нарича блок, който от своя страна се дели на три части: декларативна, изпълнима и за построяване на изключения. Понеже PL/SQL позволява смесването на SQL декларации с процедурни конструкции, е възможно PL/SQL блоковете и подпрограмите да се групират със SQL частите, преди изпращането им до Oracle сървъра за изпълнението им. Без PL/SQL, Oracle трябва да създава процес за всяка SQL декларация поотеделно, в мрежова среда, като това може да повлияе на потока на трафика и да забави неимоверно времето за отговор. PL/SQL блоковете позволяват единствена компилация и съхраняване във форма за изпълнение за подобряване времето на отговор PL/SQL програма, която е запазена в базата данни във форма за компилация, и чието извикване става чрез името ѝ, се нарича запазена процедура (stored procedure). PL/SQL запазена процедура, която се стартира безусловно, в резултат от изпълнението на INSERT, UPDATE или DELETE част, се нарича тригер (trigger).

Transact-SQL: Transact-SQL е процедурен език, използван и при Microsoft SQL Server и при Sybase SQL Server системите. Това е програмен език, снабден с всички способности, който драматично разширява силата на SQL. Докато SQL осигурява общ синтаксис за разработване на заявки, все още съществува необходимостта от протокол, който може да приема този стандартен синтаксис, да го превежда на естествен за процедурни извиквания език и в действителност да извършва заявката. Open Database Connectivity (ODBC) стандартно осигурява абстрактен слой между приложния интерфейс и базата данни, който ефективно скрива разликите и странностите на всяка специфична база данни. ODBC и SQL имат поддръжката на всяка основна софтуерна компания в света, включително Microsoft, Sun и IBM.

ODBC: ODBC, съкращение от Open DataBase Connectivity (конективност между отворени бази данни) (може да мислите за това като за универсален конектор) е един от най-популярните интерфейси за бази данни на PC-съвместимите компютри и бавно завоюва позиции и на другите платформи. Ако искаме да дефинираме с едно изречение какво представлява ODBC, то ще е: “ODBC предоставя функции за операции над бази данни от език за програмиране включително добавяне, модифициране и изтриване на данни, намиране на различни параметри на базите данни, таблиците, виртуалните таблици и индексите.” Всяко приложение, поддържащо ODBC, се състои от пет логически слоя:

- приложение,

- интерфейс към ODBC,
- диспечер на драйверите,
- драйвер и източник на данни.

Слоят "приложение" осъществява потребителския интерфейс и изчислителната част на програмата и е написан на език като Java, Visual Basic или C++. Приложението използва функциите за работа с ODBC от интерфейса за ODBC. Диспечерът на драйверите е част от ODBC на Microsoft. Той управлява наличните драйвери зарежда необходимия драйвер, насочва извикванията към него, предоставя информация за драйвера на приложението. Тъй като една програма може да взаимодейства с база данни, диспечерът на драйверите се грижи всяка СУБД да получава съответстващите й извиквания и данните от базите да бъдат безпроблемно предавани на приложенията Драйверът е компонент, който съответства на конкретната база данни, например драйвер за Access, за SQL Server или Oracle. В интерфейса към ODBC има набор от функции като SQL заявки, управление на свързването, запитвания за информация относно бази данни и др. Изпълнението им се осъществява именно от драйвера. За някои бази данни той трябва да симулира изпълнението на функции, които не се поддържат пряко от съответната СУБД. Всъщност работата на драйвера се състои в изпращането на заявки към базата. получаването на данни и препращането им към приложението. За бази данни в локални мрежи или Internet драйверът също така управлява мрежовия обмен. В контекста на ODBC източникът на данни може да бъде СУБД или просто набор от файлове върху твърд диск.

Как се свързва програма към база от данни?

DSN съдържа общ набор от информация, която всички бази данни трябва да предоставят, затова е необходимо създаването на DSN име, преди свързването на дадена програма към база данни. Ето каква информация се извлича от DSN името:

- Местоположенията на базата данни;
- Типът на драйвера на базата данни;
- Информация за потребителско име и парола;

Има два основни типа DSN имена:

- Потребителско (User) DSN името е достъпно само за акаунта, който го е създал.
- Системно (System) DSN име - достъпно е за който и да е акаунт.

JDBC: Стандартният достъп до релационни бази от данни е важен и за програмите на Java, тъй като те по природа не са монолитни, съдържащи всичките си данни приложения. Те са модулни и се нуждаят от външни бази от данни, от които четат, обработват и записват обратно данни, които после да бъдат използвани от други аплети. Монолитните приложения могат да си позволят собствени схеми за съхраняване на данните, но пресичащите границите между операционни системи и платформи аплети трябва да използват отворени стандартни схеми за достъп. JDBC (Java Database Connectivity, Връзка на Java с бази данни) на Java Enterprise е първият междуплатформен и поддържащ много СУБД програмен интерфейс за използване на бази от данни от програми на Java. От гледна точка на разработчика JDBC е първият стандартизиран метод за интегриране на Java с базите от данни. Други приложни програмни интерфейси на Enterprise са RMI, средствата за сериализация, Java IDL (Interface Definition Language, Език за дефиниране на интерфейси) за комуникация с CORBA и др. В проектирането на JDBC са използвани основни абстракции и методи от ODBC. Идеята за базиране на JDBC върху ODBC идва от това, че ODBC е популярен сред независимите разпространители на софтуер, както и сред потребителите и реализирането и използването на JDBC ще бъде по-лесно за хора, работили с ODBC. Освен това Sun и Intersolv разработват програми за връзка JDBC-ODBC, за да се използват многобройните съществуващи ODBC драйвери. С JDBC и тези програми ще можете да се работи с фактически всяка бази данни от средата на програми и аплети на Java.

Какво представлява JDBC: JDBC е набор класове и методи за взаимодействие на програми на Java с източници на данни. Той е базиран на X/Open SQL Call Level Interface (CLI), както и ODBC, за да се приеме лесно от разработчици и потребители. ODBC е интерфейс на C към СУБД, поради което не е толкова лесно да бъде прехвърлен на Java, затова JDBC следва концепциите, заложени в ODBC, но

едновременно с това е програмен интерфейс, напълно съвместим и еднотипен с останалите програмни интерфейси за Java и на места предлага по-прости за използване решения от ODBC. Най-общо архитектурата на едно приложение, използващо JDBC може да бъде видяно на фигурата, по-долу:

Какво всъщност осигурява JDBC: Според някои автори, "JDBC предоставя стандартна библиотека за достъп до релационни бази данни. Използвайки JDBC API интерфейса, можете да осъществявате достъп до множество различни SQL бази данни чрез един и същ Java синтаксис. Важно е да се отбележи, че въпреки че JDBC стандартизира механизма за свързване с бази от данни, синтаксисът за изпращане на заявки и за изпълнение на транзакции, както и структурата на данните, представляващи резултатите, JDBC не се опитва да стандартизира SQL синтаксиса. Така че можете да използвате произволни SQL разширения. Но тъй като повечето заявки следват стандартния SQL синтаксис, употребата на JDBC позволява да промените хостове на бази от данни, портове и дори производителя с минимални промени във вашия код".

Настолни бази данни: Настолните бази данни (desktop databases) обхващат тези, които могат да работят на едни от най-известните операционни системи: Dos, Windows или Macintosh. По принцип те не се справят добре при почече от пет или десет потребителя едновременно, но не са скъпи и са значително по-опростени в сравнение с инструментите от корпоративно (enterprise) ниво. Тази фамилия от инструменти е най-подходяща за малки Интранет приложения в работна група или на ниво-отдел, с нисък трафик на интернет сайтове, и като разработване на платформи за по-сложни приложения. Таблицата по-долу разглежда някои от най-популярните настолни бази данни (desktop databases), включително и дали те използват SQL и дали съществува ODBC драйвер.

Платформа

SQL

ODBC

Опции

Access (Microsoft)

Win

Да

Да

MS SQL Server

FoxPro (Microsoft)

Win, Mac, DOS

Да

Да

Unix ODBC drivers

FileMakerPro (FileMaker)

Win, Mac

Не

Не

FileMaker Server

Excel (Micorosoft)

Win, Mac

Не

Да

Конвертира до Access

ASCII/Text file

Win, Mac, DOS

Не

Да

Вмъкване в базата данни

Microsoft Excel и ASCII файловете не са бази данни сами по себе си, но често се използват като

Корпоративни бази данни: Корпоративните бази данни (enterprise databases), са значително по-мощни, сложни и обикновено скъпи. Те могат да поддържат хиляди, дори стотици хиляди потребителя едновременно. Голяма част от тях имат сложни собствени разширения към SQL, за да представят съхраняване и разработване на огромен брой приложения. Те са основно релационни бази данни. Таблица 2 разглежда главните "играчи" на арената на корпоративните Web бази данни

Платформа

Бележки

SQL Server (Microsoft)

Win (NT)

Butler SQL

Macintosh

Oracle (Oracle)

Unix, Windows NT

Обектно-релационна

Sybase (Sybase)

Unix (Linux)

Обектно-релационна

Informix(Informix)

Unix

Обектно-релационна

DB2(IBM)

Unix (AIX, Linux)

Обектно-релационна

MSQL(Huges)

Unix (vsichki)

Безплатна

MySQL

Unix (vsichki)

Безплатна

Всеки от тези комерсиални продукти е база данни, но в действителност е група от бази данни и инструменти за разработване на приложения, повечето включващи сървъри за приложения в Web (Web application server), за постигане на пълни решения за разработване на приложения с бази данни в Web. Изключенията от случая са двете безплатно разпространявани, опростени, но мощни текст-базирани бази данни за Unix, които поддържат SQL стандарта. Те не са от класата на другите приложения, ако разглеждаме гъвкавостта и силата им, но са често употребявани, защото струват хиляди долари по-малко от комерсиалните продукти. Днес бизнес информацията се съхранява чрез някоя от водещите системи за бази данни на компании като IBM, Oracle, Informix или Sybase. Всъщност всички те са про релационни бази данни от корпоративно ниво, както вече стана ясно по-горе.