Web и базите данни

Голяма част от съществуващите информационни системи са несъвместими помежду си. Това се дължи на времето, машините и технологиите на които всяка една от тях е разработвана. Обменът на данни между такива системи е силно затруднен. Промени в системите или тяхната подмяна е почти немислима поради значителния обем на данните, потребителите и високата цена на такава процедура. Очевидно е, че уеднаквяване на системите не е най-подходящия път, но уеднаквяване т.е. стандартизиране на комуникационните им възможности би ги отворило към информационния свят. В тази задача могат да се дефинират следните необходими предпоставки:

- Връзка между хетерогенни мрежи;
- Независима от времето и разстоянията комуникационна среда;
- Достъп до информацията и приложенията на различни сървъри чрез единствен клиент *(универсално клиентско приложение)*;
- Връзка чрез компютърна мрежа или по модем при възможно най-ниска цена;
- Клиенстата среда трябва да бъде независима от компютърните платформи;
- Осигурен достъп до информацията на сървъри чрез унифициран стандарт, независим от платформите.

Важно предимство на Интернет мрежата е, че тя е хетерогенна. Съвременните информационни и комуникационни технологии и свързаните с тях Интернет услуги в значителна степен отговарят на тези условия.

Класически и универсален клиент-сървър

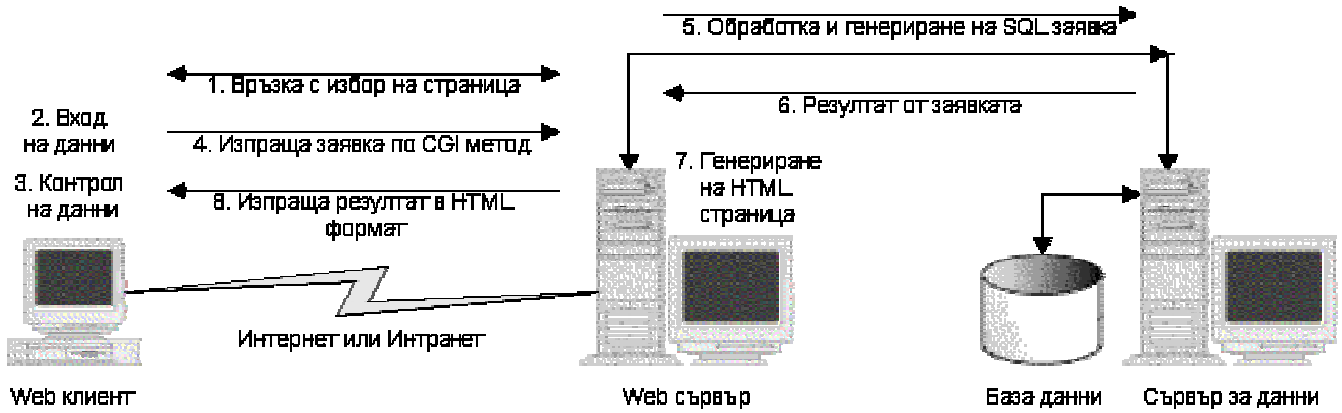
Класическият Клиент-Сървър има относително проста схема на функциониране. Клиентът управлява и извършва по-голямата част от обработките, с помощта на едно или повече приложения. Обработката може да включва:

- управление на графичен интерфейс;
- управление на взаимодействията с потребителя;
- изчисления и локална преработка на информацията;
- визуализация на данните;
- други процедури.

Клиентът изпраща към сървъра заявки под една или под друга форма (SQL, съобщения, ...). Сървърът приема заявките, обработва ги и връща на клиента резултат. Тази схема облекчава работата на сървъра, към който се стичат множество заявки за сметка на клиента, който обработва информация за всеки конкретен, изискан от потребителя случай. От клиента към сървъра тече служебна информация и такава се получава в клиента от сървъра. Управлението на графичния интерфейс и итеракциите с потребителя се извършват в машината на клиента. Конфигурацията на клиентската машина е упростена и поевтинена. Обикновено PC или Macintosh можат да се използват за тази цел.

Технологията клиент-сървър има сериозни, признати във времето предимства. Заедно с това класическия клиент-сървър има и недостатъци. Файловите сървъри са сериозно натоварени. Използваните протоколи за обмен на данни по мрежата са много "приказливи", с което причиняват задръстването й. Сървърът се натоварва от обработки за сметка на клиента. При наличие на по-голям брой клиенти скоростта на обмен със сървъра спада сериозно. Основното предназначение на тази класическа схема е в рамките на една локална мрежа и практически е неприложима за хетерогенни мрежи поради несъвместимост на много нива.

Универсалният Клиент-Сървър е основан на Web технология. Структурата му се състои от две части. Първата част обхваща всичко онова, което се намира между клиента и HTTP сървъра. Тази част е стандартизирана, независима е от платформите, основава се на Интернет услугите, и е в състояние на поддържа мрежи с нисък дебит. Втората част се отнася до всичко онова, което се намира след HTTP сървъра. В тази част се намират елементи и се прилагат технологии специфични за класическата схема на клиент-сървър*(Фигура 1)*.



Фигура 1 - Схема на универсален клиент-сървър

В средата, описана на Фигура 1 всеки отделен елемент има специфична функция. Клиентът управлява потребителския интерфейс и контролира въведените данни, с което се предотвратява излишния трафик по мрежата. Web сървърът *(HTTPD)* свързва данните между клиента и сървъра за данни. В частта сървър за данни се извършват обработки над данните и заявките *(обикновено SQL)* и се осъществява връзка със сървъра на базата от данни за достъп до данните. От страна на сървърите обработките могат да бъдат разпределени между няколко физически машини, а не върху една единствена, с което се облекчава тяхното натоварване и се увеличават ресурсните им способности да обработват значително количество заявки.

Действията на едно приложение в средата на универсален клиент-сървър се състоят от последователност от операции:

1. Установява се връзка между HTTP сървъра и Web клиента и се извлича съответната страница;
2. Данните се въвеждат в HTML формуляри, изобразени от Web браузър;
3. Скриптов език контролира на място въведената информация *(Javascript, VBScript)*;
4. Заявката се изпраща към HTTP сървър по CGI метод;
5. Обработват се данните от заявката в сървъра и се генерират SQL заявки;
6. Достъп до данните посредством сървъра на БД *(SQL сървър, ...)*;
7. Генерира се HTML страница с получените данни. Страниците се генерират от сървър за да бъдат интерпретирани от клиент;
8. Изпращат се резултатите при клиента и се визуализират от браузър.

Предимствата на универсалния клиент-сървър пред класическата схема се извеждат на преден план от способността на един клиент, независимо от неговата платформа, да контактува конкурентно с произволна конфигурация на сървър за БД посредством Web технология или HTTP сървър. Съвременните Web браузъри предоставят лесен за усвояване интерфейс и относително прости средства за програмиране, достъпни за широк кръг от потребители. Промени в структурата на базата данни, или процедурите за обработка на данните при сървъра, или промяна в неговата конфигурация не оказват влияние на клиентската част нещо почти немислимо при класическата схема.

Стандарти във Web

Успехът на Интернет се дължи преди всичко на възприетите от всички актьори в мрежата прости стандарти. Трите W е Интернет услуга, която се основава на взаимодействието на три компонента:

- Протокол HTTP;
- Скриптов език за описание на документи HTML;
- Норма-интерфейс за изпълнение на външни програми в HTTP сървър - CGI.

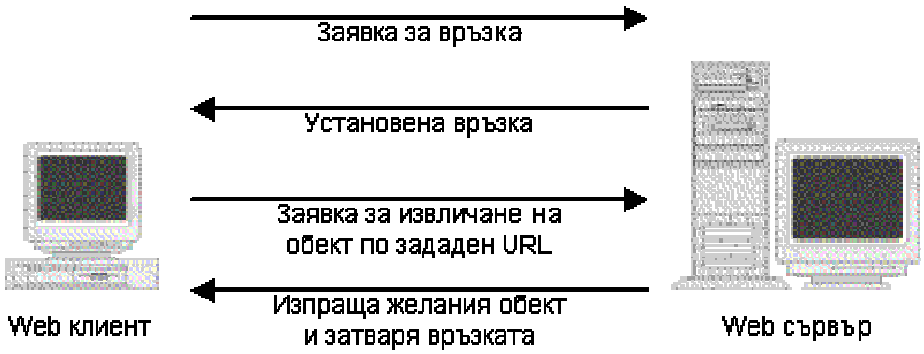
От момента на създаване на тази услуга [1992] компонентите й еволюираха много. Езикът HTML достигна до версия 4.01, въвежда се нов по-разширен стандарт за описание на хипермедийни документи XML. CGI интерфейса има всички шансове да бъде в скоро време заменен от нови сървър технологии, основани на езика Java и технологията CORBA.

Универсален клиент

В контекста на универсалния клиент-сървър, базиран на Web технология, управлението на клиента е основано на взаимодействието на три компонента HTTP, HTML и CGI. Езикът HTML е предназначен за представяне в подходящ вид на документи при клиента. Връзката между клиента и сървъра се осъществява с помощта на протокола HTTP. Поведението на клиента се следи от сървъра чрез скриптов език като CGI интерфейс. В тази проста схема, в последно време, намират място и допълнителни компоненти, които имат за цел да подобрят диалога между клиента и сървъра, да повишат сигурността при обмена на информация, и да въведат динамичен елемент в съдържанието на данните.

Протокол HTTP

HTTP ***(Hyper Text Transfert Protocol)*** е протокол за обмен на документи между сървър и клиент, работещи под управление на Интернет протоколи (TCP/IP). В Интернет мрежа това позволява бързото разпространение на хипермедийни документи. Функционирането на протокола HTTP се свързва с проста схема на "въпрос-отговор". Клиентът излъчва заявка към сървъра, на което той отговаря *(Фигура 2)*.



Фигура 2 - Взаимодействие между Web клиент и Web сървър

Показаното взаимодействие на *Фигура 2* не се отличва от това характерно за други видове Интернет протоколи. Web сървърите изпращат поисканите от клиента файлове. Подобна схема е налице и при взаимодействие между един FTP клиент и съответения FTP сървър. Разликата се състои в по-богатата функционалност на HTTP протокола. Един хипермедиен документ може да съдържа освен текст и асоциирани към него графични, звукови или видео компоненти. За една заявка на клиента от сървъра се изпращат всички елементи от нейния резултат, т.е. всички компоненти влизащи в състава на желания документ. Допълнително, в зависимост от естеството на заявката и средствата за нейната обработка, от сървъра към клиента може да се изпращат и чисто генерирани данни *(CGI интерфейс)*.

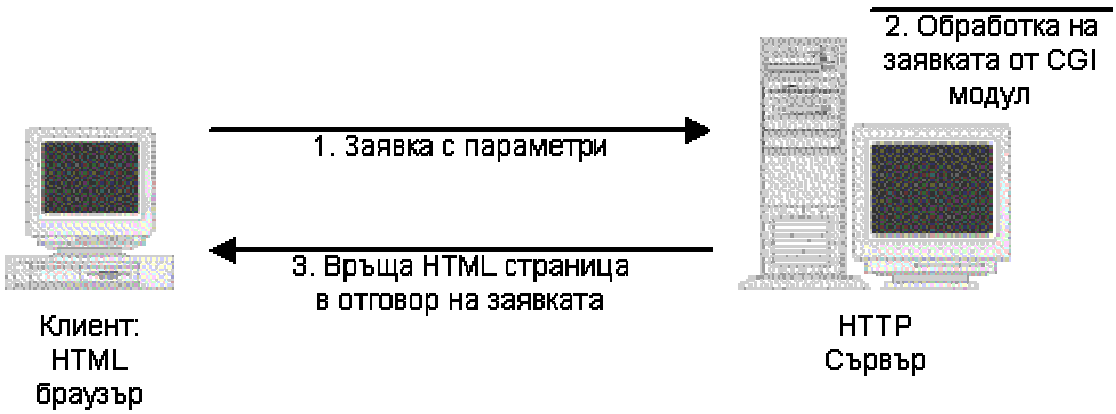
Простата схема на взаимодействия в HTTP протокола направиха Web услугата сред най-предпочитаните и популярни в Интернет. Заедно с това, тази упростена схема създава определени проблеми при по-сложни взаимодействия между клиента и сървъра, каквато е наблюдаваната тенденцията за развитие на Web технологийте. Диалог чрез обмен на файлове изключва възможност за управление на сесии в рамките на протокола, нещо необходимо при разработка на приложения взаимодействащи с бази данни.

Език HTML

HTML *(**H**yper**T**ext **M**arkup **L**angage)* е скриптов език за генериране на преносими документи по Интернет. Преносимостта се осигурява от възприетия стандарт за езика (HTML 3.2, 4.0, 4.01, DHTML) от всички компютърни платформи и производители на Web браузъри *(Виж раздела "Мултимедийни документи")*. Езиковите средства са съставени от мета-дескриптори за описание на различни текстови фрагменти и параграфи, вмъкване в текста на графични, звукови и видео компоненти, както и средства за описание на връзки към други документи. Записани в текстови файлове с разширения *.htm* или *.html* докуметите се предават на клиента, който ги интерпретира и изобразява в полето на браузъра. Чистото съдържание на един HTML файл може да се разглежда с произволен текстов редактор (Notepad в Windows, vi в Unix). Езикът HTML в този смисъл не е средство за изграждане на интерфейс, а само средство за стилово описание на документи. Динамични и диалогови елементи се изграждат с допълнителни средства.

CGI интерфейс

CGI *(**C**ommon **G**ateway **I**nterface)* е програмен интерфейс, позволяващ изпълнението на външни програми от един HTTP сървър. Това е стандарт от ниско програмно ниво, съвместим с почти всички видове сървъри от тип HTTP. Предназначението му е да позволи обработката на клиентски заявки от сървъра. CGI програмите се изпълняват на сървъра с параметри зададени от клиент. Програмният им код се реализира от *shell* скриптове, код на езика PERL, C или C++. Предпочитан в тази област е езикът PERL. Езиковите му средства са максимално адаптирани за нуждите и функциите на този род приложения. Със средствата на CGI програма по заявка на клиента в сървъра се генерира HTML документ, който му се изпраща обратно за визуализация от неговия браузър *(Фигура 3)*.



Фигура 3 - CGI интверфейс

Показаното на *Фигура 3* илюстрира следната схема на функциониране:

1. Браузърът при клиента изпраща заявка към HTTP сървъра в определена точка от йерархията на неговите страници, където се намира изпълнимата програма. Сървърът стартира локално изпълнение на програмата с набор от параметри, получени от клиента заедно със заявката.
2. Резултатът от работата на програмата се трансформира в HTML формат.
3. Резултантната страница се изпраща на браузъра на клиента.

Съществен недостатът в тази технология е, че за всяка заявка сървърът стартира отделно копие на CGI модула. Това може да доведе до сериозен недостик на ресурс при паралелни заявки от множество клиенти. И тук няма механизъм за обработка на сесии, т.е. липсва непосредствена връзка между клиента и сървъра.

Скриптови езици

Статичността на HTML страниците може да бъде разчупена само с допълнителни средства. Разширенията на езика в посока на динамизиране на изразните му средства като DHTML включва използването на скриптови езици *(Виж раздела "Езикът HTML")*. Скриптовият език заема средно положение между мета-езиците като HTML и езиците за програмиране като C, C++ и Java. Предназначението на език като HTML е да създаде стилова представа за документ и връзки към други документи. Класическите езици за програмира дефинират набор от инструкции за изпълнение от компютърната система. Скриптовите езици принадлежат към една междинна категория.

Използваните понастоящен скриптови езици в Интернет са **Javascript**, въведен от *Netscape* и **VB Script**, въведен от *Microsoft*. Правилата и синтаксиса при тези езици са упростени и не толкова строги. Формираният програмен код се вмъква непосредствено в първичен вид в текста на HTML кода за интерпретиране от браузърите. Двата езика са несъвместими и браузърите на двете софтуерни компании поддържат нееднакви множества на реципрочния език, което създава доста главоболия на Web дизайнерите. По Интернет може да се намери информация за съвместимите средства в двата езика. Характерно за *VB Script* е, че притежава прости и лесни за усвояване изразни средства, като подмножество на езика *Visual Basic*, но и възможностите му са по-ограничени в сравнение с тези на *Javascript*. Езикът е обектно-ориентиран, изисква определена предварителна подготовка в програмирането, но е предпочитан от по-напредналите заради по-гъвкавата си структура, изразни средства и възможности.

Изпълними обекти

Изпълними обекти могат да се вграждат в HTML страници. Зареждането им става по подобен на един графичен обект начин и се изпълняват автоматично. Обекти, които се зареждат със страницата и се изпълняват автоматично са два типа:

- Аплети на Java *(SUN)*;
- Active X компоненти *(Microsoft)*.

Използването на тези обекти обогатява статичните страници с по-атрактивни и диалогови елементи. Спомагат за изграждане на функционален потребителски интерфейс от страната на клиента без да се претоварва за това сървъра. Създават условия за боравене с нестандартни за Web технологията обекти и формати.

Мултиплатформения модел е една стара идея, която получи живот с появата и развитието на на езика Java. Един **аплет на Java** може да се изпълнява на различни компютърни платформи без да е необходимо да бъде прекомпилиран, което е задължително за приложения написани на език като C или C++.

Изпълнението на външни програми, заредени по мрежата при клиента, поставя въпроси относно сигурността. При изпълнението на аплет от клиентски браузър този въпрос се решава със средствата на самия език.

Active X не е мултиплатформен модел и е тясно свързан с технологията OLE в среда Microsoft Windows. За разлика от аплетите на Java Active X е приложим само в среда Windows. В зависимост от гледната точка, това може да се третира като сериозен недостатък и същевременно като предимство. Предимство е факта, че всички инструментални средства за развитие под Windows поддържат Active X, с което преминаването към Web технология става по-естествено. Непосредствено може да се интегрира съдържанието на една таблица от Excel в HTML страница, което важи и за други Windows компоненти. Един компонент на Active X, за разлика от аплет на Java, се зарежда еднократно на твърдия диск при клиента и при следващо извикване не се зарежда от сървъра, с което се печели време, но възникват проблеми по отношение на сигурността.

Plug-ins

Допълнителните програмни модули, инсталирани към всеки Web браузър за обработка на определени текстови, графични, звукови, видео и други видове формати се наричат *plug-ins*. Те са в състояние да подпомогнат разнообразяването на HTML страниците, но не се вписват в технологията на универсалния клиент-сървър. Те са специфични за всеки браузър и съответната платформа и не са предназначени да обменят информация със сървъра.

Универсален Клиент-Сървър

Управлението от страна на сървъра при универсалния клиент-сървър модел не се обхваща от определени стандарти, какъвто е случаят при клиента. Тук нещата са специфични за всяко приложение, но те могат да се групират в две основни техники за осъществяване на достъп до информацията на база данни по Web технология.

Първата се базира на генерирането на статични HTML страници. Втората се основава на създаването на динамични бази от данни, достъпни непосредствено чрез Web. Трети вариант е възможен при комбиниране на първите два.

Генериране на статични HTML страници от база данни

Тази техника във всичките си стъпки е статична:

1. Обслужване на базата данни в режим off-line.
2. Периодично генериране на HTML страници като сечение на БД.
3. Прехвърляне на страниците във Web сървър.
4. Консултиране на страниците от сървъра.

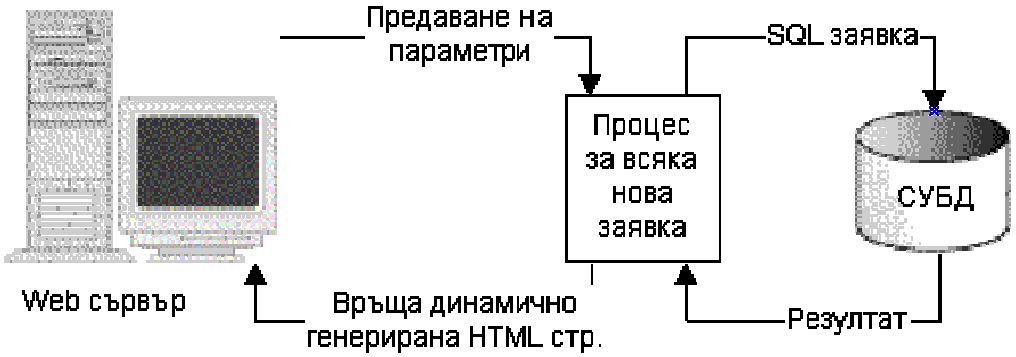
БД и Web сървъра се обслужват и работят независимо един от друг и могат да бъдат разположени на различни машини. Генерирането на статични страници може да се извършва от външна програма или от самото СУБД. Голяма част от съвременните системи за управление на бази от данни предоставят тази възможност. Прехвърлянето на генерираните страници в сървър е процес на чисто копиране, например по FTP протокол, или по протоколите на локална мрежа *(LAN)*. Клиентът консултира информацията от Web сайта при нормални условия.

Реализацията на този модел е относително лесен за осъществяване, прост за поддръжка и не изисква допълнителни инвестиции. Неудобствата, свързани с неговото използване не са малко. Информацията във Web сървъра не се обновява автоматично при промяна на данните в базата данни.

Генериране на динамични HTML страници от база данни

Реалните условия на динамично променяща се информация в една БД налагат и съответните гъвкави методи за консултиране на данните в нея по Web технология. При тази техника са разработени пет основни архитектури.

1. **CGI** интерфейса е станал своеобразен стандарт за всички производители на Web сървъри *(Фигура 4)*.



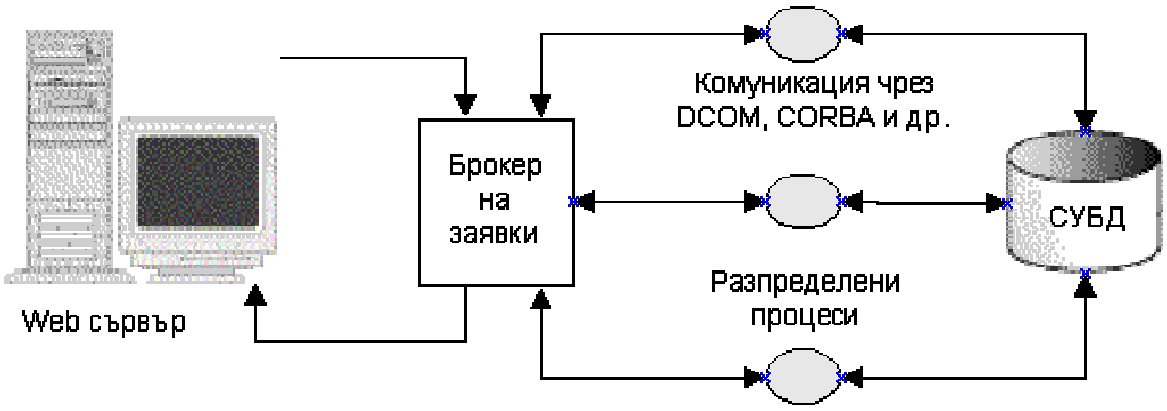
Фигура 4 - Взаимодействия при CGI архитектура

Относително простият за реализация и поддържане CGI модел предизвиква често задръстване на сървъра при много едновременни заявки от клиентски машини. За всяка, получена в сървъра заявка CGI модула стартира отделен процес с отделна SQL заявка към СУБД. Тази архитектура е многоплатформена, CGI модулите се реализират на избран от разработчика език. При възникване на грешка отпада една заявка, с което не се предизвиква срив в системата.

2. **API** *(**A**pplication **P**rogram **I**nterface)* е архитектура създадена като алтернатива на CGI от Netscape и Microsoft със сходна схема на функциониране. С API се въвежда един естествен диалог между Web сървъра и съответните приложения за достъп до БД. С помощта на предварително компилиран DLL *("Процес" от Фигура 4)* в общ блок от паметта се

извършва обработка на параметрите за заявката, получава се резултата от заявката и динамично се генерира HTML страница. Приложението е многонишково, но се реализира по различен начин. Netscape въвежда NSAPI, а Microsoft ISAPI, които са несъвместими помежду си. Грешките се обработват трудно поради общо използваната памет, което може да доведе до срив в системата.

3. **ASP** *(**A**ctive **S**erver **P**age)* е архитектура разработена от Microsoft. В основата на ASP е заложена идеата за интерпретиране на код в сървър изпратен от клиент. В статичните HTML страници се вмъкват допълнителни елементи, които изпратени в сървъра ще бъдат интерпретирани от съответен интерпретатор. Подобна идея е реализирана при PHP/FI, където в HTML кода се вмъкват инструкции, написани на скриптов език, подмножество на езика PERL, и интерпретирани във Web сървър от вида Apache с активиран модул за интерпретация. В този случай HTML страниците са с разширение *.phtml*. Предшественик на тази архитектурна схема е технологията SSI, с която се постига диалог с клиента почти в реално време. Схемата на функциониране на тази технология е проста и достъпна за развитие. "Процесът" от *Фигура 4* е интерпетираща част в сървъра, където се обработва скрипт кода от клиентската страница, извлича се SQL код, получения резултат от БД се конвертира в HTML формат преди да бъде изпратен обратно в браузъра на клиента. Съществен недостатък в тази схема е задължителната връзка със сървър за интепретиране на код, което заема ресурс и е процес по-бавен от изпълнението на компилирана програма.
4. **WRB** *(**W**eb **R**equest **B**roker)* или посредник на заявки е архитектура въведена от Oracle. Посредникът играе ролята на арбитър при управлението на разпределени обекти по Web. Заявка от клиент се приема от посредника, който активира процес в отделен блок памет. Всеки процес управлява своя опашка, където се изпълнява заявката и се получават резултати. Брокерът на заявки може да управлява множество процеси едновременно чрез оптимизиране на опашките и заетата памет.



Фигура 5 - Брокер на заявки

Тази архитектура позволява на базата на протокола TCP/IP процесите да бъдат разпределени между няколко машини. Архитектурата WRB е изключително стабилна при работа и надежна по отношение на резултатите. Обемът на обработваната информация не влияе на скоростта на изпълнение на заявките. При възникване на грешка, брокерът на заявки е в състояние автоматично да рестартира обработката. Единствен недостатък тук може да се посочи много високата цена за реализация и необходимостта от специлно подготвен персонал за съпровождане и развитие на системата *(Фигура 5)*. Изискванията към Web сървъра са специфични и обикновено се удовлетворяват от фирмени разработки като Oracle Web Server.

5. **Архитектура "реле"** е метод, основан на най-нови разработки в информационните технологии. Обектно-ориентираният модел, езикът Java, и разработките на SUN в областта

на CORBA стандарта намират приложение в реализацията на тази архитектура. Фирмата Microsoft използва DCOM метод в тази идеа. Сървърът HTTP инициира начален диалог между обекти-клиент и обекти-сървър. Всяка клиентска заявка извиква съответен метод от списък с приложения в сървъра. Експериментират се различни разработки в усилията за реализация на тази архитектура. За комуникация между отдалечени обекти в среда Java се използва метод RMI *(Remote method Invocation)*, Java/CORBA технологията прилага IIOP *(Internet Inter-ORB Protocol)*. Прилагането на тази архитектура предполага функционална пълнота на приложенията, но реализацията е сложна.

Средства за достъп до БД

До средата на 1996 година средствата за решаване на достъпа до данни се основават на стандартизиран интерфейс между HTTP сървър и СУБД *(Система за Управление на База Данни)*. В тези средства се включват методите CGI, ISAPI, NSAPI, с чиято помощ трудно се постига необходимата интерактивност между клиента и сървъра. Нарастналите нужди на корпоративните и университетските компютърни мрежи, както и безусловният успех на глобалната мрежа Интернет подтикна компаниите Sun и Microsoft да търсят по-перспективно решение на проблема. Така се появяват две нови технологии JDBC *(Java Data Base Connectivity)* предложена от Sun и ADO *(Active X Data Objects)* предложена от Microsoft.

JDBC и ADO са компоненти за достъп до данни на две конкурентни технологии - Java и Active X. Двете технологии са предназначени за изграждане на Web базирани клиент-сървър приложения с използване на протокола HTTP. Така става възможно поддържане на постоянна връзка между клиента и сървъра, което е предпоставка за изграждане на приложения за управление на транзакции, нещо характерно за съвременните релационни бази данни. Двете технологии са аналозите на стандартния клиент-сървър, базиран на ODBC. Общите точки между двете технологии е тяхната обектна ориентация, капсулирането на API функциите, предоставени от производителите на СУБД, относителната им независимост от ограниченията на различните Web архитектури. Различията между двете технологии са в използваните езици и в платформите, за които са приложими.

Достъп до база данни с ADO и ASP с VBScript

Основна задача на този пример е да се покаже, как може да се изпълни **SQL** заявка в **ASP** *(Active Server Page)* страница и да се върне резултат в **HTML** таблица. Това изисква използването на **ADO** *(ActiveX Data Objects)* за установяване на връзка с източник на данни *(Connection Data Source)* и манипулирането им с методите на обекта *(Recordset)*.

Усвояването на тази технология предполага известни познания по *ASP*, *SQL*, *ODBC*, обектно-ориентирано програмиране и бази данни *(MS Access 2000)*.

Източник на данни

За целите на примера се изпозва готова база данни *(dbstud.mdb)*, реализирана на *MS Access 2000*, съдържаща една таблица *(Екран 6)* с данни за студенти и техните оценки *(Students)*.

students : Table

	FacNo	Name	Subject	Score
	1	Петър	Химия	5.50
	2	Марина	Информатика	6.00
	3	Георги	Математика	4.50
	4	Гергана	Физика	5.00
	5	Стефан	Информатика	6.00
▶	6	Иван	Електротехника	4.00

Record: 6 of 6

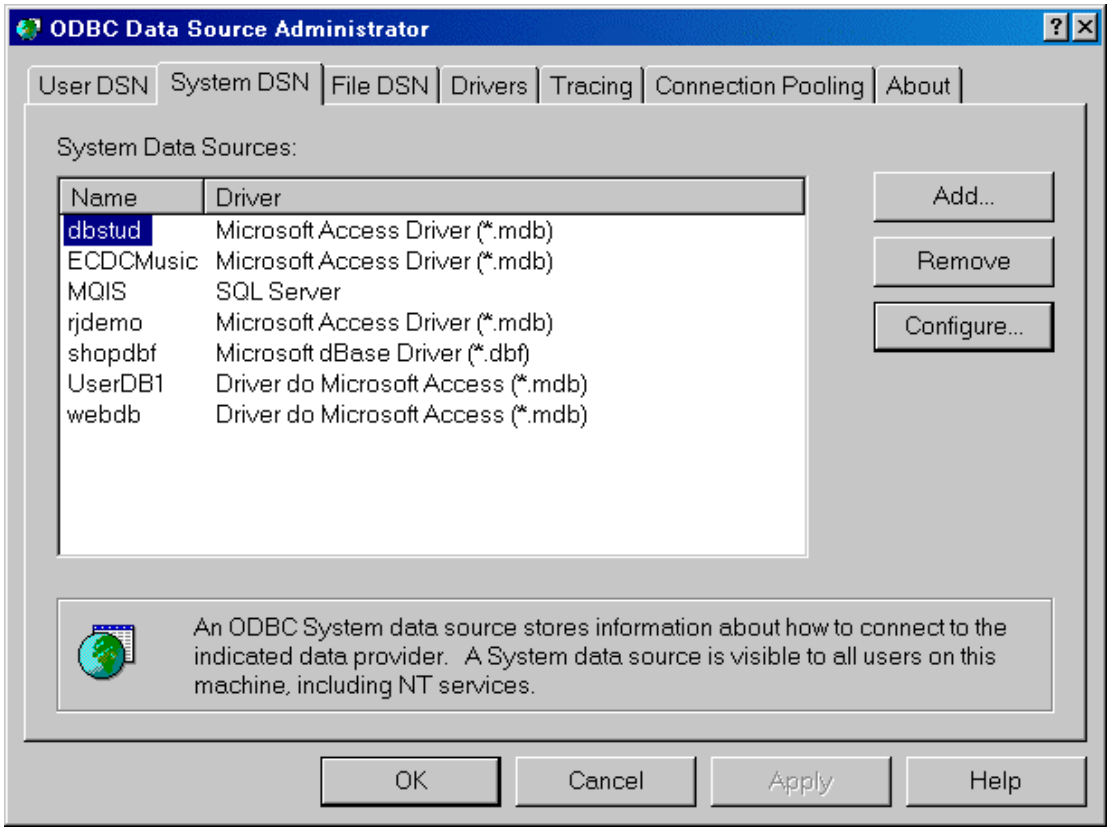
Екран 6 - Примерна база данни с информация за студенти

Достъп до базата данни се осъществява посредством **ODBC** *(Open Data Base Connectivity)*. За целта трябва да бъде конфигуриран драйвер за достъп до данните на база данни на *MS Access 2000* и да се укаже месторазположението ѝ на диска. От *Start/Settings/Control Panel* се избира *ODBC Data Sources* за установяване на параметрите за настройка *(Фигура 7)*. Последното важи за Windows 98 и NT 4.0. При Windows 2000 има още едно ниво *Administrative tools*, от където става достъпен *Data Sources (ODBC) (Фигура 8)*.

Фигура 7 - Windows 98/NT

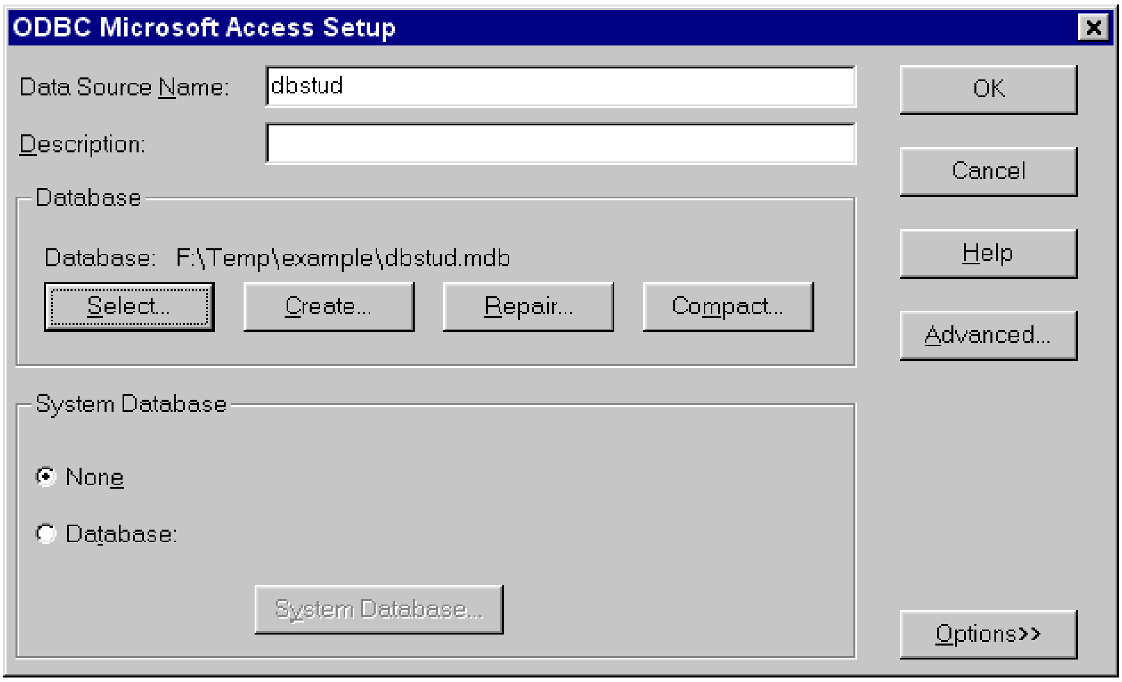
Фигура 8 - Windows 2000

В секцията *System DSN* се добавя *Microsoft Acces Driver* и се виждат вече инсталираните драйвери: *(Екран 9)*



Екран 9 - Драйвери и символни имена на БД

След избор на бутон *Finish* се появява *Екран 10*. Тук се задава името на източника на данните и месторазположението на базата данни върху диска.



Екран 10 - Параметризиране на ODBC за конкретна БД

Инсталираният драйвер се появява в списъка на *System DSN*. Избор на бутон *OK* завършва процедурата.

От *Екран 10* е видно, че източника на данни *(Data Source Name)* е с име *dbstud*, а базата данни *dbstud.mdb* се намира на *f:\temp\example*. Приключване на тази процедура става след избор на бутон *OK*. Така се изпълнява задължителната предварителна настройка на достъпа до БД.

Изграждането на *ASP* страница с възможност за достъп до информацията на тази БД преминава през няколко етапа.

Свързване към източника на данни

Свързването към източника на данни става с помощта на обект за връзка *(Connection object)*. За създаването на този обект се използва CreateObject метод на Server обекта.

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

Свързването на обекта за връзка *(cnDB)* с източника на данни *(dbstud)* става чрез метод Open.

Изпълнение на SQL заявка

Използването на *SQL* оператори в *ASP* страница се подчинява на синтаксиса на *ASP*. Оператора се присвоява на променлива *strQuery*, която се използва като параметър при изпълнение на *SQL зявката* от типа SELECT. Полученият резултат от изпълнението на метода Execute е обект от типа *Recordset* и съдържа върнатите данни.

```
 – –
```

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□
```

Възможно е използването на заявки от типа UPDATE, INSERT или DELETE. Тяхното изпълнение се извършва непосредствено чрез *Execute* метода.

Извличане на информация

При изпълнение на заявка от типа *SELECT* се получава еднопосочен резултат от тип обект *Recordset*. Извличане на данните се извършва в циклична структура *(Do While)* при гранично условие край на файл *(Eof)*.

```
□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□табулиращи□инструкции
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□елемент□на□структура□□□□
□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□
□□□□
```

Първоначално *Recordset* обекта сочи първия ред данни, върнати след изпълнение на *SQL* заявка. Методът *MoveNext* служи за преминаване на следващ ред. След преминаване зад последния ред се включва условието *Eof*, с което цикълът се прекратява.

Табулиране на резултата

Резултатите се представят в HTML таблица, с форматиране на заглавед ред и следващите редове със съдържанието на резултата от заявката.

```
□□□□□□□
□□□□□□
□□□□□□□□Фак□□□□□□□□□□□□□Име□□□□□□□□□Предмет□□□□□□□□□Оценка□□□□□
□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□
□□□□
□□□□□□□□□□□□□□□□□□□
□□□□□□□□
□□□□
□□□□□□□□□
```

Сред *HTML* елементите, задаващи параметрите на таблица, се вмъкват *ASP* инструкции, с което се получава желаното табулиране на данните.

Затваряне на връзката

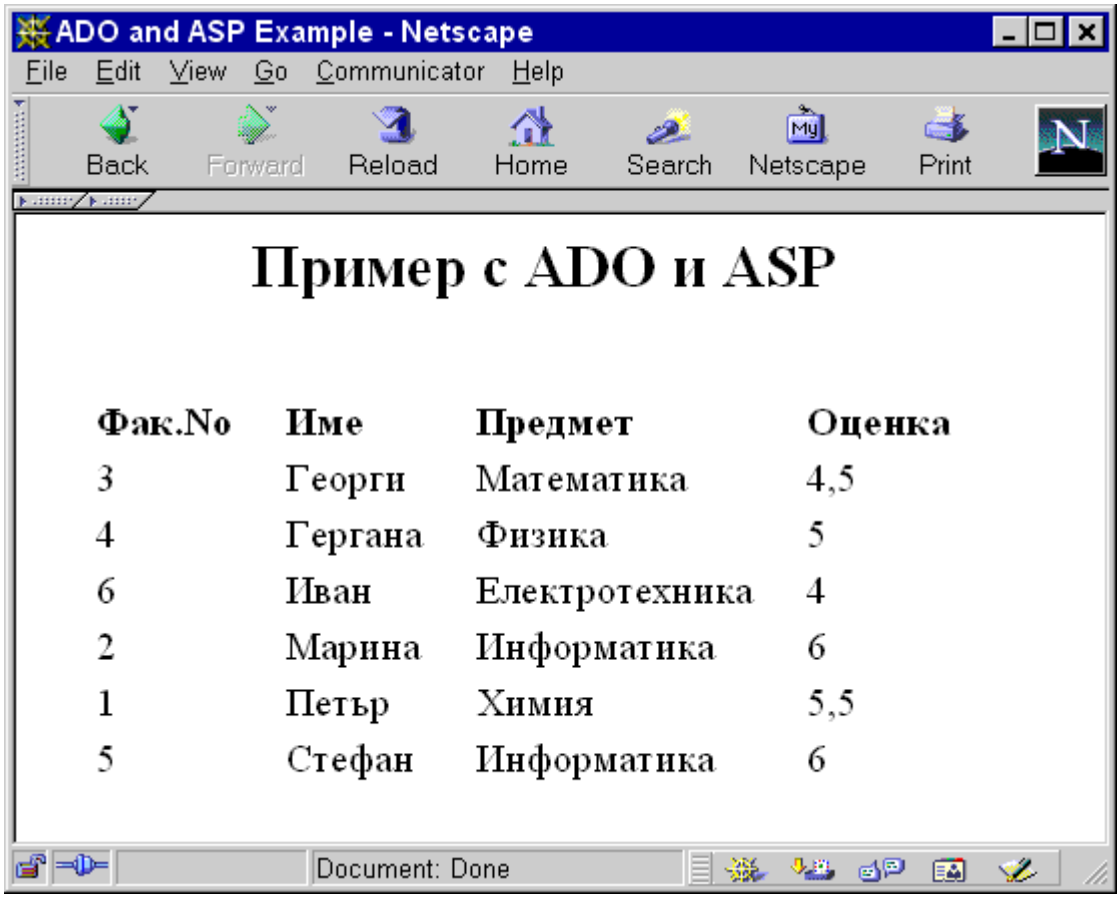
Финалната операция след изпълнение на заявката и получаване на резултатите е освобождаване и затваряне на всички използвани обекти.

```
 □□
□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□
□□□□□
```

Изпълнение

Подготовката на едно самостоятелно изпълнение на този пример изисква инсталиран WEB сървър *IIS4* или *PWS*. Създаване на база данни *dbstud.mdb* на *MS Access 2000*, конфигуриране на *ODBC*. Необходимият за изпълнението *ASP* файл *(dbstud.asp)* може да се създаде с текстов редактор *(Notepad)* и да се запише в зоната на сървъра, например в директория □□□□□□□□□□□□□□□□□□. Директорията трябва да бъде с разрешени права за *"execute"* и *"scripts"*.

Резултатът от успешно изпълнение на Уеб приложение за връзка с база данни е показан на *Екран 11*.



Екран 11 - Резултат от SQL заявка

Примерът може да бъде проигран дистанционно след избор на посочената връзка тук. Съдържанието на използвания ASP файл може да се види тук.

Достъп до база данни с ADO и ASP с JScript

JScript е реализация на *Microsoft* на езиковата спецификация *ECMA 262*, популярна най-вече като **Javascript**. Реализацията покрива напълно изискванията на спецификацията, като заедно с това са

налице и някои допълнителни опции, характерни за техническите характеристики и възможностите на *Microsoft Internet Explorer 4.x | 5.x* като ***Web*** навигатор.

JScript е обектно-ориентиран, скриптов език, интерпретиран както от страна на ***Web*** клиента, така и от страна на ***Web*** сървър. Приложението му е сходно с това на **VBScript**, но на практика е предпочитан поради по-голямата си близост с **JavaScript**. Сходството е пълно на ниво езикови характеристики и средства за програмиране. Различията настъпват при използване на някои обекти и техните методи.

Пример за извеждане на справка от БД *dbstud.mdb* по описаната вече техника чрез **ADO** и **ASP**, но с използване на **JScript** в качеството му на скриптов език за интерпретиране от страна на Web сървъра.

Съдържание на *dbstudjs.asp*:

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□
□□□□□□
□
□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□
□□
□□□

□□□□□□□
□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□Справка□от□БД□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□Студенте□□□□□□□□□□□□□□□
```

```
□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Фак□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Име□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Предмет□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Оценка□□□□□□□□□□□□□□□□
□□□□□□□

□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□
□□□□□□□□
□□□□□□□□
□□□□□□□
```

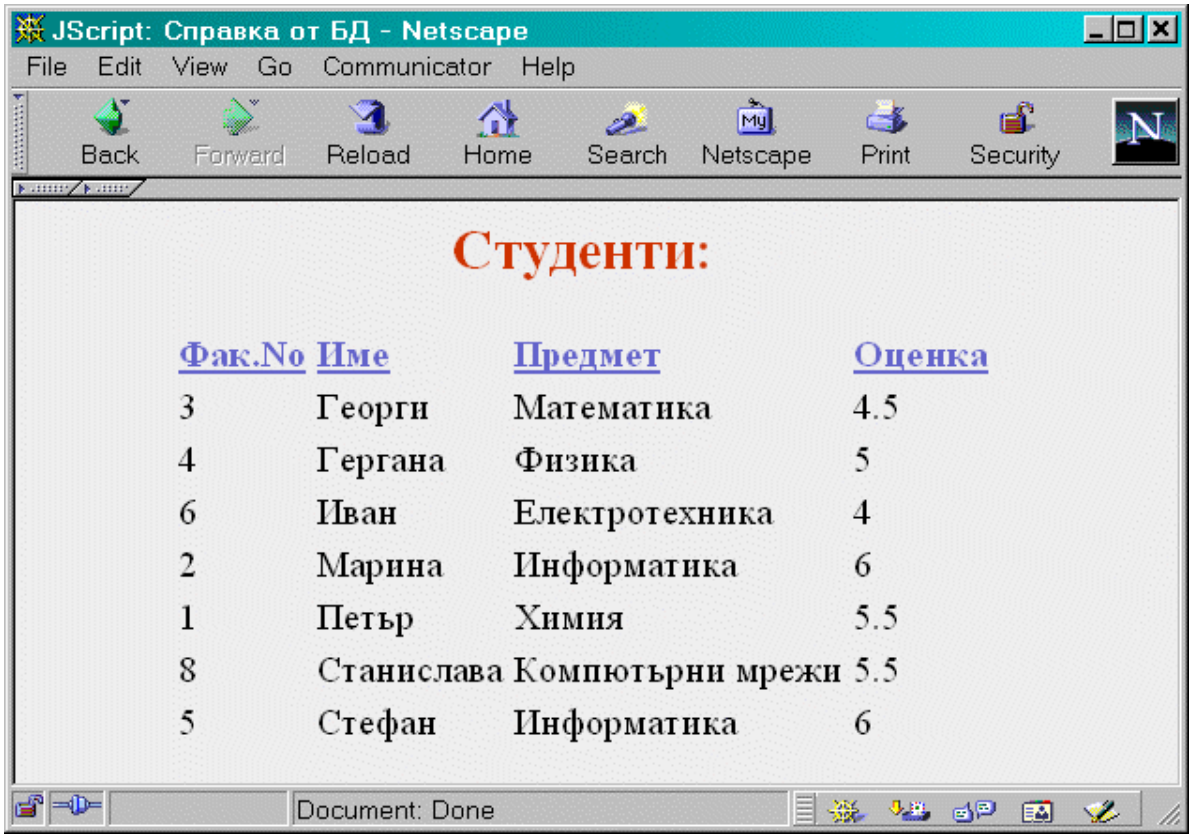
Съдържание на HTML страница изпратена на клиента след интерпретация от сървъра:

```
□□□□□□□□
□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□Справка□от□БД□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□Студенти□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Фак□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Име□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Предмет□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Оценка□□□□□□□□□□□□□□□□□
□□□□□

□□□□□
□□□□□□□□□□□□□□Георги□□□□□□□□□Математика□□□□□□□□□□□□□□□□□□
□□□□□□
□□□□□
□□□□□□□□□□□□□□Гергана□□□□□□□□□Физика□□□□□□□□□□□□□□□□
□□□□□□
□□□□□
□□□□□□□□□□□□□□Иван□□□□□□□□□Електротехника□□□□□□□□□□□□□□□□
□□□□□□
□□□□□
□□□□□□□□□□□□□□Марина□□□□□□□□□Информатика□□□□□□□□□□□□□□□□
□□□□□□
□□□□□
□□□□□□□□□□□□□□Петър□□□□□□□□□Химия□□□□□□□□□□□□□□□□□□
□□□□□□
□□□□□
□□□□□□□□□□□□□□Станислава□□□□□□□□□Компютърни□мрежи□□□□□□□□□□□□□□□□□□
□□□□□□
□□□□□
□□□□□□□□□□□□□□Стефан□□□□□□□□□Информатика□□□□□□□□□□□□□□□□
□□□□□□
□
□□□□□□□□□
□□□□□□□□□
□□□□□□□
```

Резултат при изпълнение:



Фак.No	Име	Предмет	Оценка
3	Георги	Математика	4.5
4	Гергана	Физика	5
6	Иван	Електротехника	4
2	Марина	Информатика	6
1	Петър	Химия	5.5
8	Станислава	Компютърни мрежи	5.5
5	Стефан	Информатика	6

Екран 12 - Справка от БД с JScript

Непосредствено изпълнение на примера от сървъра